An efficient acyclic contact planner for multiped robots

Steve Tonneau, Andrea del Prete, Julien Pettré, Chonhyon Park, Dinesh Manocha, Nicolas Mansard

Submitted to The Int. Jour. of Robotics Research

An efficient acyclic contact planner for multiped robots

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

Abstract

We present a framework capable of producing contact plans describing complex multiped motions (including humanoid): standing up, climbing stairs using a handrail, crossing rubble and getting out of a car. Our framework answers a need demonstrated at the Darpa Robotics Challenge, where the lack of an automatic acyclic contact planner was recognized a major issue.

Our novel key idea is the reachability condition. Informally, it verifies that the root configuration of a robot is "close, but not too close" from obstacles: close to allow contact creation, not too close to avoid collision. With this approximation of the space of admissible root configurations we decompose the hard contact planning problem into simpler sub-problems: first, to plan a guide path for the root without considering the whole-body configuration; then, to generate a discrete sequence of whole-body configurations in static equilibrium along this path. The reachability condition turns the high-dimensional computation of the guide into a collision checking problem, solved in less than a few seconds. Then a deterministic contact selection algorithm tackles the combinatorial issue of generating of a discrete sequence along the guide path. Several innovations make it computationally efficient: a criterion for verifying static equilibrium, and a set of heuristics used to enforce desirable properties on the configuration.

Our approach results from the pragmatic choice of favoring efficiency over exhaustiveness, justified empirically: in a few seconds, with satisfying success rates, we generate complex contact plans for various scenarios and robots, namely HRP-2, HyQ, and a dexterous hand.

Keywords

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

1 Introduction

We consider the problem of planning an acyclic sequence of contacts describing the motion of a multiped robot in a constraining environment. Acyclic contact planning is a particular class of motion planning in which the robot must be in contact with the environment at every configuration to maintain the static equilibrium.

Most multipedal locomotion systems focus on cyclic walking gaits (Kajita et al. 2003). However, executing this behavior on constraining environments is dangerous, if not impossible. In an analysis of their participation to the Darpa Robotics Challenge, Atkeson et al. noted: "Except for egress, no robots in the DRC Finals used the stair railings or any form of bracing. Even drunk people are smart enough to use nearby supports. Full body locomotion (hand holds, bracing, leaning against a wall or obstacles) should be easier than our current high performance minimum contact locomotion approaches."

Indeed, the current approach to locomotion planning is to avoid obstacles as much as possible, instead of using them to facilitate locomotion. The reason for this, as the authors state, is that "More contacts make tasks mechanically easier, but algorithmically more complicated for planning, and the transitions are difficult to both plan and control[...]. We have seen very few robot planners that are capable of generating this behavior." The difficulty of addressing such a problem comes both in practice from the proximity to obstacles (which makes tedious the sampling of collision-free configurations) and in theory from the foliation of

[1]LAAS-CNRS / Université Paul Sabatier, 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

the configuration space, where zero-measure manifolds intersect in a combinatorial manner (Siméon et al. 2004).

Previous contributions that embrace the combinatorial provide complete approaches, not applicable in practice because they require hours of computations (Bretl 2006). More recent successes use a local solution, resulting in more reasonable computation times (still far from real time), at the cost of dynamically inconsistent behaviors (Mordatch et al. 2012). Neither global nor local methods present satisfying performances, because planning simultaneously the robot trajectory and the contacts that allow its execution is a very hard problem.

As suggested by Bouyarmane et al. (2009), we believe that these two problems can be treated sequentially within a global planner. We go further and claim that this can be done at a much smaller cost, provided we can formulate a computationally-efficient condition for *equilibrium feasibility* of the root configuration (i.e. there exists a joint configuration such that the robot is in static equilibrium). This paper presents a geometrical approximation of this condition, and a concrete implementation of such a decoupled planner. This approximation results from a trade-off between a necessary and a sufficient condition for *equilibrium feasibility*, allowing us to find solutions extremely rapidly, while preserving a high success rate in the demonstrated scenarios.

In the remainder of this introduction, we discuss further the current state of the art. This allows us to situate more precisely our contributions.

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 intrinsically not able to adapt to new situations or discover complex behaviors in unforeseen contexts.

The issue of planning acyclic contacts was first completely described by Bretl et al. in their seminal paper (Bretl 2006). The issue requires the simultaneous handling of two problems, $\mathcal{P}_1$: planning a relevant guide path for the root of the robot in $SE(3)$; and $\mathcal{P}_2$: planning a discrete sequence of acyclic equilibrium configurations along the path. A third nontrivial problem, $\mathcal{P}_3$, not addressed in this work, 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. This 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 particular situations, typically 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 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. A complete example of the potential offered by such approaches was proposed (Mordatch et al. 2012) and successfully applied to a real robot (Mordatch et al. 2015). To keep reasonable computation times, the method uses a simplified dynamic model for the avatar. Still, the computation time is far from real-time (about 1 minute of computation for a sequence of 20 contacts). A similar approach has been considered for manipulation by Gabiccini et al. (2015). Deits and Tedrake propose to solve contact planning globally as a mixed-integer problem, but only cyclic, bipedal locomotion is considered. Dai et al. extend the work of Posa et al. to discover the contact sequence for landing motions, but need to specify the contacts manually for more complex interactions. In addition to the practical limits of the current implementations, a major drawback of these optimization-based approaches is that they only offer local convergence when applied to acyclic contact planning.

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 is shown by Escande et al. who manually set up a rough root guide path (i.e. an ad-hoc solution to $\mathcal{P}_1$). $\mathcal{P}_2$ is addressed as the combinatorial computation of a feasible contact sequence in the neighborhood of the guide. A solution can 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 focus 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 into the contact sub-manifold using a 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 positions in the path. Chung and Khatib also propose 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. This approach is efficient in the demonstrated scenarios. In highly-constraining cases such as the car egress scenario that we address here, we believe that including collision constraints in the planning is a requirement.

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

As far as robotics applications are concerned, none of the existing planners is able to solve the problem with *interactive* performances. 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, Carpentier et al. are able to achieve this with *interactive* performances on a real robot. 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 Rationale

This paper presents a pragmatic approach to break the complexity of multi-contact planning. With that objective in mind, we believe that the separation between the generation of the root guide path and the contact sequence is the most promising direction (Escande et al. 2008). However, this direction raises two theoretical questions that remain to be solved, or even to be properly formulated.

Regarding $\mathcal{P}_1$, the guide path must guarantee the existence of a contact sequence to actuate it. We call this property *equilibrium feasibility*. This property has not been studied yet; the only rigorous way to validate a waypoint in the path is to explicitly compute the contact locations and forces, which is computationally not reasonable (Bouyarmane et al. 2009), unless the scenario is limited to cyclic, quasi-flat cases (Zucker et al. 2010).

Regarding $\mathcal{P}_2$, there are infinitely many combinations of possible contact sequences for a given root path. The selection of one particular contact sequence with interesting properties (minimum number of contact changes, robustness, efficiency or naturalness) has been studied for cyclic cases (Hauser et al. 2006), but has not been efficiently applied to constraining environments (Bouyarmane and Kheddar and Escande et al. mostly randomly picked one contact sequence, possibly leading to very tedious transitions).

The key idea of the paper is the *reachability condition*, a computationally-efficient approximation of the *equilibrium feasibility*. We assume that most of the times the *reachability condition* implies the *equilibrium feasibility*. This assumption is not always verified, but we provide empirical evidence that it is in the considered class of problems, the *cluttered contact planning problems*. Such problems can be solved with a contact sequence where in every configuration at least one contact is *quasi-flat* (Del Prete et al. 2016). A contact is *quasi-flat* if the friction cone contains the direction opposite to gravity.

To address $\mathcal{P}_1$, we first use the *reachability condition* to plan the guide path. This step is fast because the planning happens in a low-dimensional space, and does not require explicit contact computations.

Then we address $\mathcal{P}_2$ by extending the path into a sequence of whole-body configurations in static equilibrium. This second step requires the explicit computation of contact configurations.

This sequential approach is the key to the efficiency of our method, though it can result in failures because our hypothesis is not always true. However, we demonstrate empirically the validity of the approach: the high success rate, combined with the low computation times, allow us to plan (and re-plan upon failure) multi-contact sequences at *interactive* rates. Similarly, in this paper we do not prove that all the contact sequences produced by our planner can be interpolated ($\mathcal{P}_3$), although we have been able to do this for some of the demonstrated sequences (Carpentier et al. 2016).

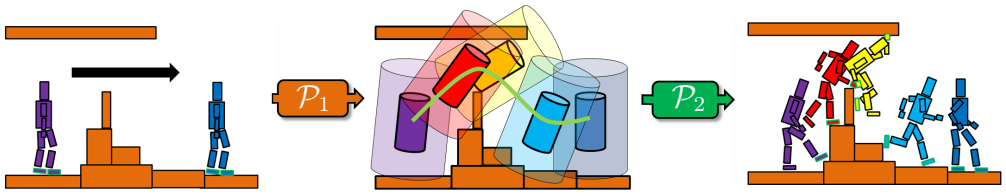

Figure 1. Overview of our two-stage framework. Given a path request between start and goal positions (left image), $\mathcal{P}_1$ is the problem of computing a guide path in the space of *equilibrium feasible* root configurations. We achieve this by defining a geometric condition, the *reachability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

1.3 Paper contribution and organization

We propose contributions to both problems $\mathcal{P}_1$ and $\mathcal{P}_2$.

- A low-dimensional, efficient sampling-based approach to plan guide paths.
- A very efficient and general implementation of an acyclic contact planner, the first one compatible with *interactive* applications.
- Four heuristics for contact generation. They bias the planner towards configurations that are in robust static equilibrium, or more efficient with respect to the task.
- Statistical tests that empirically demonstrate the validity of our approach in the considered scenarios.

In Section 2, we present an overview of our method. Section 3 and Section 4 present respectively our answer to $\mathcal{P}_1$ and $\mathcal{P}_2$. In Section 5, we present heuristics used for the selection of a contact configuration. Finally, in Section 6 we propose a complete experimental validation of the planner with three very different kinematic chains (the HRP-2 and HyQ robots, and a three-finger manipulator) in various scenarios.

Comparison with our previous work

The present paper is an extension of a conference paper to appear in the proceedings of the ISRR'15 conference (Tonneau et al. 2015). The conference paper focuses on the theoretical formulation of the problem, and presents results obtained with virtual avatars. This extension completes this work and brings new contributions, resulting from the implementation of our approach on real-world robots and problems. In particular, a strong contribution is the introduction of a robust equilibrium criterion, designed to ensure the equilibrium of the robot despite bounded errors in the contact forces, coupled with heuristics for relevant contact selection. These two contributions were required to apply our method to real-robot models, namely HRP-2 and HyQ (Semini et al. 2011). In the present paper, a complete algorithm for the contact planner is given, and an access to the source code of our implementation is provided. Furthermore, our experiments are supported by an in-depth analysis of our performances, as well as a discussion on the influence of the parameters to the method.

2 Overview

Figure 1 illustrates our workflow. $\mathcal{P}_1$ and $\mathcal{P}_2$ are addressed in a sequential fashion: we first plan a root guide path with the *reachability condition*, before extending it into a sequence of configurations in static equilibrium. In this Section we describe this workflow and give some notations used throughout the paper. Some key concepts are introduced, whose definitions can be found in the glossary at the end of this paper.

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

We first consider the problem of planning a root guide path. Ideally we would like to sample root configurations that are *equilibrium feasible*. A root configuration is *equilibrium feasible* if and only if there exists at least one joint configuration such that:

C1 : the robot is not in collision with the environment,

C2 : the robot is in static equilibrium.

Verifying C2 is computationally expensive because it requires the projection of the whole-body configuration into a contact posture, as well as the resolution of a linear program to compute the contact forces (Del Prete et al. 2016). For this reason, we are interested in finding an approximation of the *equilibrium feasibility* that is computationally efficient. A requirement for C2 is the condition:

C3 : at least one end-effector is in contact with the environment.

C1 and C3 define *contact feasibility*, which is easier to verify than *equilibrium feasibility* because it does

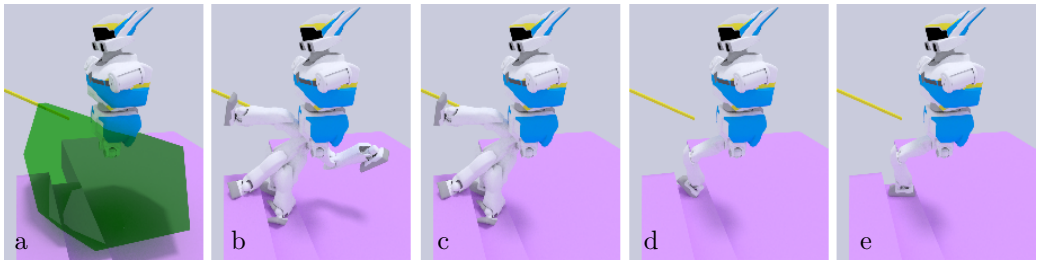

Figure 2. Generation of a contact configuration for the right leg of HRP-2. (a): Selection of reachable obstacles. (b): Entries of the limb samples database (with $N = 4$). (c): With a proximity query on the octree database, configurations too far from obstacles are eliminated. (d): The best candidate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

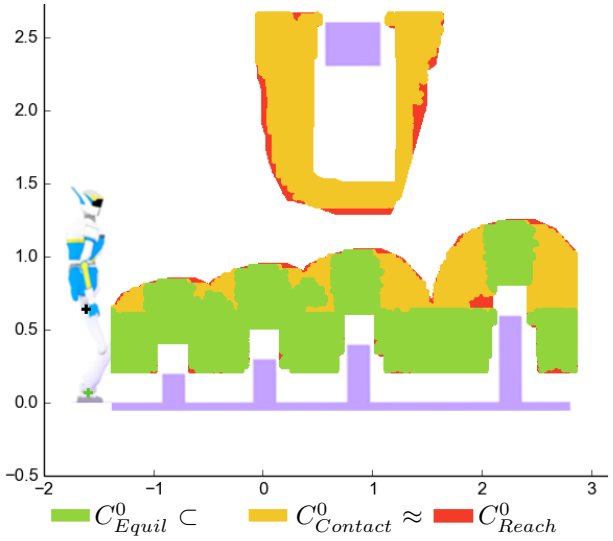

Figure 3. Illustration of several root configurations sets used in this paper in a 2D scene. Obstacles are violet, and units are in meters. To show the sets in a 2D representation, all the rotational joints of HRP-2 are locked in the shown configuration, such that a torso configuration is only described 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 dramatic divergence between C^0_{Equil} and $C^0_{Contact}$ when the problem is not *cluttered*.

not require the computation of contact forces. We make the assumption that for the *cluttered contact planning problem*, most of the times *contact feasible* root configurations are also *equilibrium feasible*. For this reason we neglect C2 during the resolution of $\mathcal{P}_1$, and we consider it only when solving $\mathcal{P}_2$. However, even verifying *contact feasibility* is computationally too expensive because it requires the projection of the whole-body configuration into a contact posture.

An intuitive description of *contact feasible* configurations is "close, but not too close": close, because a contact surface must be partially included in the reachable workspace of the robot (represented for the right leg in Figure 2–a); not too close, because the robot must avoid collision. We approximate *contact feasibility* with the *reachability condition*, a geometrical criterion based on this intuitive description. More precisely, a root configuration is *reachable* if the root scaled by a factor $s \geq 1$ is not in collision, while the reachable workspace is in collision with the environment. Verifying the *reachability condition* is very efficient in practice. In Section 3 we detail how we define and use the *reachability condition* to compute a guide path with the Reachability-Based RRT (Figure 1—$\mathcal{P}_1$).

Figure 3 gives an insight into the difference between the three conditions, depicting C^0_{Equil}, $C^0_{Contact}$ and C^0_{Reach}, which are the sets of *equilibrium feasible*, *contact feasible* and *reachable* root configurations, respectively.

In the remainder of this paper, we use the terms *contact feasible* 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–$\mathcal{P}_2$). With a fixed root position, the dimensionality of the contact generation problem is reduced to the number of degrees of freedom of the considered limb. We select from a database of precomputed limb configurations (independent from the environment) the ones resulting in the end-effector being close to the environment. Limiting our selection to this set of precomputed configurations allow us to drastically

Figure 4. 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.

reduce the computation times. Then we select one of these configurations based on user-defined heuristics (Figure 2), which we present in Section 5.

From a given start configuration, the planner proceeds in an iterative fashion along the discretized path: given a new root configuration, an inverse-kinematics solver is used to maintain the contacts that were previously existing. Possibly, some of these contacts cannot be maintained (because of joint limits or collisions). Then the contacts are broken. Conversely, new contacts are created to ensure the static equilibrium of the robot. The algorithm is designed so that only one contact can be created or broken between two successive configurations. While it does not provide guarantees that the interpolation between two configurations is achievable, it appears empirically as a reasonable heuristic. Details are presented in Section 4.

2.3 Notation conventions 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, that we divide into different parts: 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 $\mathbf{q} \in \mathbb{R}^{r+n}$. We define some relevant projections of $\mathbf{q}$:

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

The volume encompassing R^0 is denoted W^0 (Figure 4). The reachable workspace of a limb R^k is denoted W^k:

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

where $\mathbf{p}^k$ denotes the end-effector position of R^k (translation only) for $\mathbf{q}^0 = \mathbf{0}$ being the null displacement, and C^k_{jl} 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.

3 Computing a guide path in C^0_{reach} ($\mathcal{P}_1$)

We consider the issue of computing an *equilibrium feasible* guide path $\mathbf{q}^0(t) : [0,1] \longrightarrow C^0_{Equil}$ for the root of a multiped robot, connecting user-defined start and goal configurations.

Again, we assume that, for *cluttered* problems, most of the times *contact feasibility* implies *equilibrium feasibility*. Under this assumption, our goal is to find a *contact feasible* guide path.

To generate such a path efficiently, ideally we need to only sample *contact feasible* root configurations. This requires exhibiting a necessary and sufficient condition for *contact feasibility*. By default, verifying *contact feasibility* implies a constructive demonstration by exhibiting a valid $\mathbf{q}^{\overline{0}}$. This is the approach chosen by Bouyarmane et al. (2009), which is too computationally expensive. To formulate a cheaper condition, we may turn our attention towards either an only-necessary or an only-sufficient condition.

Only-necessary conditions are appealing because they preserve the completeness of the search, while reducing the search space: they provide an outer approximation of $C^0_{Contact}$. On the other hand, only-sufficient conditions provide the guarantee that any configuration that satisfies them is indeed *contact feasible*: they provide an inner approximation of $C^0_{Contact}$.

In practice, the only-necessary and only-sufficient conditions that we can provide are trivial and give rather inaccurate approximations of $C^0_{Contact}$. Therefore, we propose a compromise between them: the *reachability condition*, which is computationally efficient and provides a rather accurate approximation of $C^0_{Contact}$.

3.1 Conditions for contact feasibility

Contact feasibility, a necessary condition: For a contact to be possible, a volume $O_i \in O$ necessarily intersects with the reachable workspace $W(\mathbf{q}^0)$ (Figure 2–1).

Furthermore, if $\mathbf{q}^0$ is *contact feasible*, then the torso of the robot $W^0(\mathbf{q}^0)$ is necessarily not colliding with the environment O.

Therefore we can define an outer approximation $\mathcal{C}^0_{Nec} \supset C^0_{Contact}$ defined as:

$$\mathcal{C}^0_{Nec} = \{\mathbf{q}^0 : W(\mathbf{q}^0) \cap O \neq \emptyset \text{ and } W^0(\mathbf{q}^0) \cap O = \emptyset\} \tag{2}$$

Contact feasibility, a sufficient condition: A trivial sufficient condition for *contact feasibility* can be constructed as a variation of $\mathcal{C}^0_{Nec}$, by replacing W^0 with a bounding volume B^{Suf} encompassing the whole robot in a given pose, except for the effector surfaces to be in contact. We denote by $\mathcal{C}^0_{Suf} \subset C^0_{Contact}$ the set of root configurations corresponding to this sufficient condition:

$$\mathcal{C}^0_{Suf} = \{\mathbf{q}^0 : W(\mathbf{q}^0) \cap O \neq \emptyset \text{ and } B^{Suf}(\mathbf{q}^0) \cap O = \emptyset\} \tag{3}$$

3.2 Contact feasibility: a compromise reachability condition

The sufficient condition is not interesting in practice since it leads the solver to lose too many interesting paths. The necessary condition is not perfect either, since the first stage of the planner would stop on a guide that is not *contact feasible*. An ideal shape B (with $W^0 \subset B \subset B^{Suf}$) that leads to a necessary and sufficient condition may exist—even if it seems intuitively very unlikely in general.

However, using a shape between W^0 and B^{Suf} leads to a trade-off between a necessary and a sufficient condition. 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

$$C^0_s = \{\mathbf{q}^0 : W(\mathbf{q}^0) \cap O \neq \emptyset \text{ and } W^0_s(\mathbf{q}^0) \cap O = \emptyset\} \tag{4}$$

If $s = 1$, then $W^0_s = W^0$, such that $C^0_1 = \mathcal{C}^0_{Nec}$. We thus consider that $s \geq 1$, since smaller values would only worsen the approximation. By increasing s, the condition can become sufficient. The parametrization of s then defines a trade-off between these two interesting extremes. We can choose s by hand, or automatically as explained in Section 6.2.1. The chosen value $s = s^*$ defines the *reachability condition*, therefore we write $C^0_{reach} = C^0_{s^*}$.

In Appendix A, we give a generic method to compute the volumes appearing in the definition of C^0_{reach}, with the example of HRP-2.

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

Any sampling-based motion planner can be used to plan a path in C^0_{reach}. Indeed, contrary to $C_{Contact}$, C^0_{reach} has a nonzero measure in the configuration space C. Therefore a standard uniform sampling approach can work, in spite of a high rejection rate. Thus, the only significant change regarding a classical planner is to replace the collision checking with the *reachability condition* when verifying the drawn configurations and associated local paths.

However, to improve the sampling efficiency we bias the sampling process to generate near-obstacle configurations, similarly to Amato et al. (2000). Our current implementation of these modifications is based on the Bi-RRT planner (LaValle and Kuffner 1999) provided by the HPP software.

Thanks to these modifications, the problem of planning a *contact feasible* path is reduced to a geometric collision-checking problem, of low dimension (6 for HyQ, 8 for HRP-2 that has 2 joints in the torso). By doing this we can solve the problem with a sampling-based approach, in *interactive* computation times.

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

Our planner computes guide paths in C^0_{reach}, an approximation of C^0_{Equil}. As an input of this stage, we however assume an *equilibrium feasible* root guide path $\mathbf{q}^0(t) : [0, 1] \longrightarrow C^0_{Equil}$. If this is not the case, our planner will fail rapidly, thus allowing replanning, as discussed in Section 6.3. We now consider the second problem of computing a discrete sequence of equilibrium configurations $\mathbf{Q}^0$ along $\mathbf{q}^0(t)$.

In this Section we first describe a single contact-generation process, that is how to generate a contact configuration for a limb, given a root location. Then, we propose an iterative algorithm to generate a discrete sequence of contact configurations in static equilibrium.

Our criterion to assert efficiently the static equilibrium of the system is described in Section 5.

4.1 Definition of a contact generator

Given a configuration $\mathbf{q}^{\overline{k}}$ of the root and all the limbs but R^k, we look for a limb configuration $\mathbf{q}^k$ such that R^k is in contact, and not colliding (neither with parts of the robot nor with the environment). While exhibiting analytically a $\mathbf{q}^k$ does not seem tractable, we can iteratively try to generate one as follows:

1. Generate randomly a collision-free limb configuration;
2. Project the end-effector onto the closest surface with inverse kinematics;
3. If a valid solution is found, stop. Otherwise repeat from step 1.

We trade the completeness of this approach for a more efficient solution introduced in our previous work (Tonneau et al. 2014), which we describe here to be exhaustive.

We first define $C^{\epsilon}_{Contact} \supset C_{Contact}$ as the set of configurations almost in contact. This means that the minimum distance between an effector and a surface of the environment is less than a small value ϵ. We then apply the following steps:

1. Generate off-line N valid sample limb configurations $\mathbf{q}^k_i, 0 \leq i < N$;
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, retrieve from the octree the list of samples $S \subset C^{\epsilon}_{Contact}$ close to contact (Figure 2 (b) and (c));
4. According to a user-defined heuristic h, sort the elements of S;
5. Select the first configuration of S. Project the configuration onto contact with inverse kinematics. If S is empty, stop (failure case) (Figure 2 (d) and (e));
6. If a valid solution is found, stop (success case). Otherwise remove the element from S and go back to step 5.

This approach has two main advantages. First, this allows us to select a large number of candidates in a single proximity request. Having several candidates is interesting, because it allows to compare them using a user-selected heuristic h, thus obtaining a locally-optimal candidate. Furthermore, the fact that the candidates are already close to contact increases the odds that the inverse kinematics will converge to a valid solution in a small number of iterations. Regarding convergence, it is immediate to verify that as N grows, the probability of finding a solution if it exists converges to 1. N is a parameter that allows us to specify the trade-off between exhaustiveness and efficiency. The reader is invited to refer to our previous work for an extensive discussion on the optimal value of N (Tonneau et al. 2014).

4.2 Extension of the guide path

Using the contact generator, we define an iterative algorithm to generate the contact sequence along the guide path. Although we address acyclic contact sequences, the algorithm is deterministic in the order in which the contacts are created, allowing it to break the combinatorial. The complete algorithm can be found in Appendix B. In the remainder of the section we provide an intuition of it.

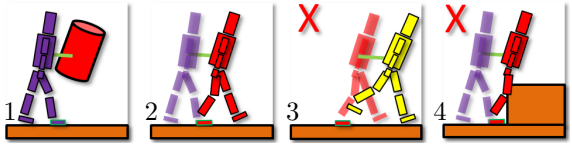

Figure 5. Contacts are maintained if joint limits and collisions constraints are respected (2). They are broken otherwise(3,4).

As an input, we consider the guide path $\mathbf{q}^0(t)$, 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 $\mathbf{q}^0_{j-1}$ respectively correspond to the start and goal configurations. We want to extend the configurations of $\mathbf{Q}^0$ in such a way that continuity is preserved regarding the contact transitions. To do so, we define an algorithm that, given the current root configuration, and the previous full body configuration, computes a full body configuration in C_{Equil} such that contacts are maintained if possible. The first full body configuration of the sequence is given by the initial state of the robot.

4.2.1 Maintaining a contact in the sequence: Figure 5 illustrates the contact-persistence strategy. If possible, a limb in contact at step $i-1$ remains in contact at step i. The contact is broken if an inverse-kinematics solver fails to find a collision-free limb configuration that satisfies joint limits. The solver is directly provided by the HPP software.

If the solver fails, the contact is broken and a collision-free configuration is assigned to the limb. If two or more contacts are broken in a single step, one or more intermediate steps are added. For these steps the root configuration is the same as for the previous step, with the difference that one faulty contact is repositioned, in the hope that it will not be broken at the next step.

4.2.2 Creating contacts: Contacts are created using three rules:

1. Only one contact creation can happen between two consecutive steps. This increases the odds that the interpolation between the two steps be feasible;

2. A contact is validated if and only if the resulting configuration is in static equilibrium;
3. We use a FIFO approach: we always try first to create a contact with the limb that has been contact-free the longest. If the contact creation was not successful for a limb, the limb is pushed on top of the queue, and will only be tried again after the others.

These three rules provide a deterministic generation of contacts along the discretized guide path. For a given *equilibrium feasible* path, our current implementation does not guarantee that the planner will succeed, because of the deterministic approach used to break the combinatorial. However, in practice the planner is successful in the large majority of cases, as discussed in Section 6.3.

5 Heuristics for contact selection

5.1 A heuristic 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.

5.1.1 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);
- $\mathbf{m} \in \mathbb{R}$ 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 $\mathbf{p_i}$;
  - $\mathbf{n}_i, \mathbf{t}_{i1}, \mathbf{t}_{i2}$ form a local Cartesian coordinate system centered at $\mathbf{p_i}$. $\mathbf{n}_i$ is aligned with the contact surface normal, and the $\mathbf{t}_i$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:

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

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:

$$\exists \boldsymbol{\beta} \in \mathbb{R}^{4e}, \boldsymbol{\beta} \geq 0 \text{ and } \mathbf{f} = \mathbf{V}\boldsymbol{\beta}, \tag{5}$$

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:

$$\underbrace{\begin{bmatrix} \mathbf{I}_3 & \cdots & \mathbf{I}_3 \\ \hat{\mathbf{p}}_1 & \cdots & \hat{\mathbf{p}}_e \end{bmatrix}}_{\mathbf{G}} \mathbf{V}\boldsymbol{\beta}, = \underbrace{\begin{bmatrix} \mathbf{0}_{3\times3} \\ m\hat{\mathbf{g}} \end{bmatrix}}_{\mathbf{D}} \mathbf{c} + \underbrace{\begin{bmatrix} -m\mathbf{g} \\ \mathbf{0} \end{bmatrix}}_{\mathbf{d}} \tag{6}$$

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 (5) and (6), it means that the configuration is in static equilibrium. The problem can then be formulated as an LP:

$$\begin{aligned} \text{find} \quad & \boldsymbol{\beta} \in \mathbb{R}^{4e} \\ \text{subject to} \quad & \mathbf{G}\boldsymbol{\beta} = \mathbf{D}\mathbf{c} + \mathbf{d} \\ & \boldsymbol{\beta} \geq 0 \end{aligned} \tag{7}$$

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

$$\begin{aligned} \text{find} \quad & \boldsymbol{\beta} \in \mathbb{R}^{4e}, b_0 \in \mathbb{R} \\ \text{minimize} \quad & -b_0 \\ \text{subject to} \quad & \mathbf{G}\boldsymbol{\beta} = \mathbf{D}\mathbf{c} + \mathbf{d} \\ & \boldsymbol{\beta} \geq b_0\mathbf{1} \end{aligned} \tag{8}$$

We observe that if b_0 is positive then (7) 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 (8), we solve an equivalent problem of smaller

dimension that we get by taking the dual of (8) and eliminating the Lagrange multipliers associated to the inequality constraints:

$$\begin{aligned} \text{find} \quad & \boldsymbol{\nu} \in \mathbb{R}^6 \\ \text{maximize} \quad & -(\mathbf{D}\mathbf{c}+\mathbf{d})^T\nu \\ \text{subject to} \quad & \mathbf{G}^T\boldsymbol{\nu} \geq 0 \\ & \mathbf{1}^T\mathbf{G}^T\boldsymbol{\nu} = 1 \end{aligned} \tag{9}$$

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 (9) is indeed the optimal b_0.

5.2 Manipulability-based heuristics for contact selection

This Section proposes heuristics to select a contact that optimizes desiered 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. In this section, we derive three such heuristics from the work on manipulability by Yoshikawa (1985), that we present first.

5.2.1 The force and velocity ellipsoids: For a limb configuration $\mathbf{q}^k$, the Jacobian matrix $\mathbf{J}^k(\mathbf{q}^k)$ defines the relation:

$$\dot{\mathbf{p}}^k = \mathbf{J}^k(\mathbf{q}^k)\dot{\mathbf{q}}^k \tag{10}$$

For clarity in the rest of the section we omit the k indices and write $\mathbf{J}^k(\mathbf{q}^k)$ as $\mathbf{J}$.

As a linear approximation of a forward-kinematics function, $\mathbf{J}$ describes how small variations from the configuration $\mathbf{q}$ affect the position vector $\mathbf{p}$.

Now we consider the unit ball in the configuration space C defined by the set of joint velocities for which the norm is at most 1:

$$||\dot{\mathbf{q}}||^2 \leq 1 \tag{11}$$

We assume that $\mathbf{J}$ is full rank (we are not interested in singular configurations, which we discard). From (10) we can thus obtain the following equality (Appendix C):

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

We can use (12) to map the ball into an ellipsoid in the Euclidian space $\mathbb{R}^m$:

$$\dot{\mathbf{p}}^T(\mathbf{J}\mathbf{J}^T)^{-1}\dot{\mathbf{p}} \leq 1 \tag{13}$$

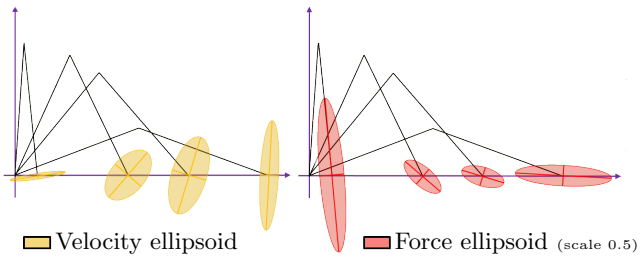

Figure 6. Examples of velocity and force ellipsoids for a manipulator composed of 2 dofs and 2 segments. Only the horizontal and vertical speeds are shown (not the rotation speeds), since it would require being able to draw in four dimensions.

This ellipsoid is called the manipulability ellipsoid, or velocity ellipsoid, introduced by Yoshikawa (1985). It describes the set of end-effector velocities that can be reached under the constraint (11) for the current configuration. The longer the axis of the ellipsoid is, the faster the end-effector can move along the direction of the axis. Figure 6 - left shows the velocity ellipsoid for different configurations of a manipulator with two degrees of freedom.

Similarly to the velocity ellipsoid, Yoshikawa also defines the force ellipsoid. Considering: a force vector $\mathbf{f}$ expressed in the task space $\mathbb{R}^m$; the equivalent joint torque vector $\boldsymbol{\tau}$; we can define the mechanical work in both spaces:

$$\dot{\mathbf{q}}^T\boldsymbol{\tau} = \dot{\mathbf{p}}^T\mathbf{f}$$

Exploiting (10), we can easily see that the set of achievable forces in $\mathbb{R}^m$ subject to the constraint:

$$||\boldsymbol{\tau}||^2 \leq 1$$

is the so-called force ellipsoid (Figure 6 - right):

$$\mathbf{f}^T(\mathbf{J}\mathbf{J}^T)\mathbf{f} \leq 1 \tag{14}$$

5.2.2 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 $\mathbf{m}$, the length of the ellipsoid is given by the force-transmission ratio (Chiu 1987):

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

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

$$h_{EFORT}(\mathbf{q}, \mathbf{m}) = [\mathbf{m}^T(\mathbf{J}\mathbf{J}^T)\mathbf{m}]^{-\frac{1}{2}}(\mu \mathbf{n}^T \mathbf{m}) \qquad (15)$$

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:

$$h_{vel}(\mathbf{q}, \mathbf{m}) = [\mathbf{m}^T(\mathbf{J}\mathbf{J}^T)^{-1}\mathbf{m}]^{-\frac{1}{2}}(\mu \mathbf{n}^T \mathbf{m}) \qquad (16)$$

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:

$$h_w(\mathbf{q}) = \sqrt{det(\mathbf{J}\mathbf{J}^T)} \qquad (17)$$

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.

h_{EFORT}, h_{vel} and h_w define three kinematic heuristics, fast to compute (indeed, thanks to our sampling-based approach, the Jacobian and inverse products can be precomputed off-line), that allow the planner to select the best candidates according to user-defined criterion.

6 Results

In this Section we present some of the results obtained with our planner. The complete sequences computed are shown 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. For each scenario we indicate the chosen heuristics. We also provide a performance analysis, which shows that the planner is compatible with *interactive* applications, and present the success rates obtained in each scenario. Moreover, we demonstrate the interest of our robustness criterion in the different computed poses. Finally, a last example suggests possible applications to dexterous manipulation.

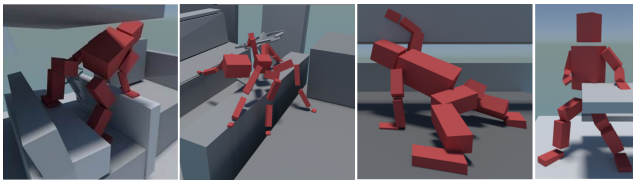

Figure 7. Virtual avatars in various scenarios demonstrated in our conference paper.

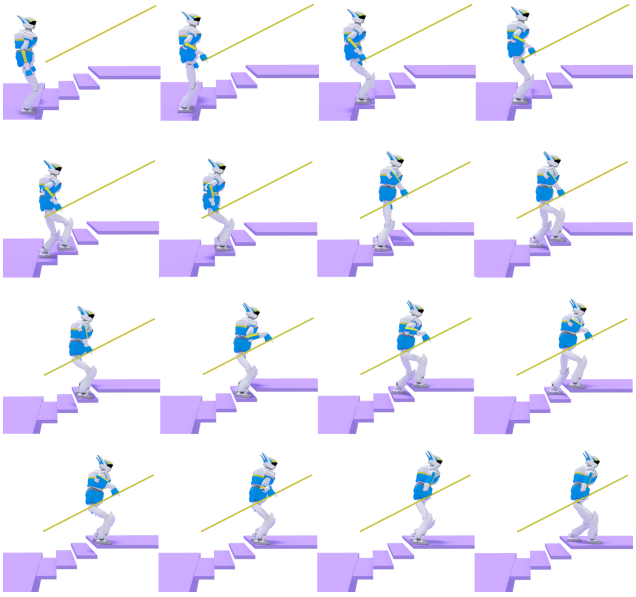

Figure 8. HRP-2 in the steep stair climbing scenario.

In our previous work (Tonneau et al. 2015) additional results are demonstrated with various virtual avatars (Figure 7). In this extension we choose to focus on actual robots. We invite the interested reader to watch the ISRR video (http://youtu.be/LmLAHgGQJGA), and to refer to the previous paper for a discussion on these results.

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 an input of the scenario. 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 parameters chosen: the heuristics, and the constraints on the reachable workspaces (for instance in all the scenarios, the reachable workspaces of the legs of HRP-2 are always required to intersect with the environment).

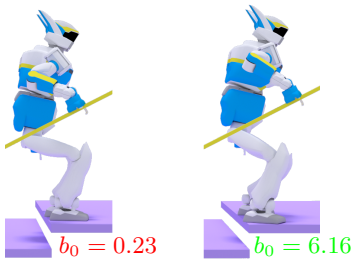

Figure 9. Evaluation of the robustness b_0 of two contact configurations. Although in equilibrium, the left configuration is on the verge of slipping.

6.1.1 HRP-2 – Steep staircase (Figure 8): The goal is to climb three 15-cm high steps. This height requires HRP-2 to use a ramp to perform the task.
Contacts involved: Feet and right arm.
Heuristics: The manipulability h_w is chosen for the feet; h_{EFORT} is chosen for the right arm. Regarding equilibrium, the video demonstrates two sequences computed for two different threshold values of b_0: 0 and 2 (Figure 8).
Observations: This scenario illustrates best the importance of the equilibrium-robustness criterion. With a robust approach, more states are required to reach the last step (15 rather than 13 in average). However, when the last step is reached by both feet, in the nonrobust case the contacts are extremely close to the cone limits (Figure 9).

The geometry of the environment is easily addressed by our planner, and the contact planning is several times faster than real time in this scenario.

Again, the interpolation motion between the contact steps is out of the scope of this paper. However it should be noted that the computed plan in this scenario has been executed successfully on the robot (`http://youtu.be/YjL-DBQgXwk#t=0m28s`).

6.1.2 HRP-2 – Standing up (Figure 10): From a bent configuration, a standing-up motion is computed in a constraining environment. The resulting motion involves 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.
Observations: The scenario illustrates well the acyclic aspect of the planning. For instance, in the four first frames of Figure 10, we can see that the right foot is moved twice, with the left foot in between, before the configuration allows HRP-2 to move its hand. Because the contacts are tried in a FIFO manner, the fact that the output contact sequence is acyclic shows that a cyclic approach (with a finite state machine for instance) is not sufficient for the computed path. The reason for this is not reachability, but equilibrium. The planning is slower than for the stair scenario (because the contact generation fails more), though it remains compatible with *interactive* performances.

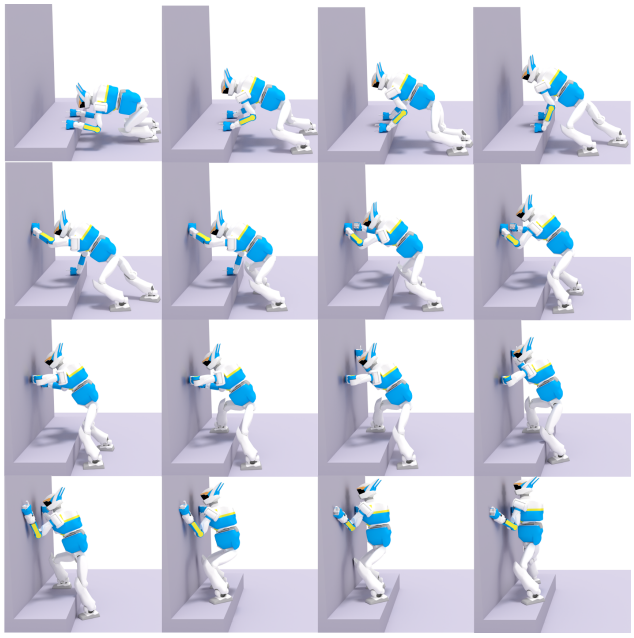

Figure 10. HRP-2 in the standing scenario.

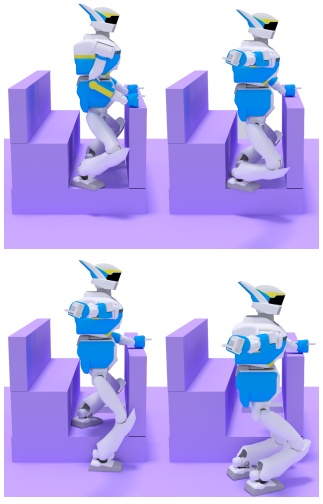

Figure 11. Selected frames from the car egress scenario.

6.1.3 HRP-2 – Car egress (Figure 11): This scenario is inspired from the Darpa challenge car egress scenario (`http://cpc.cx/edH`). HRP-2 has to find the contact sequence that allows it to step out of a car.
Contacts involved: All (both feet and hands).
Heuristics: h_w.

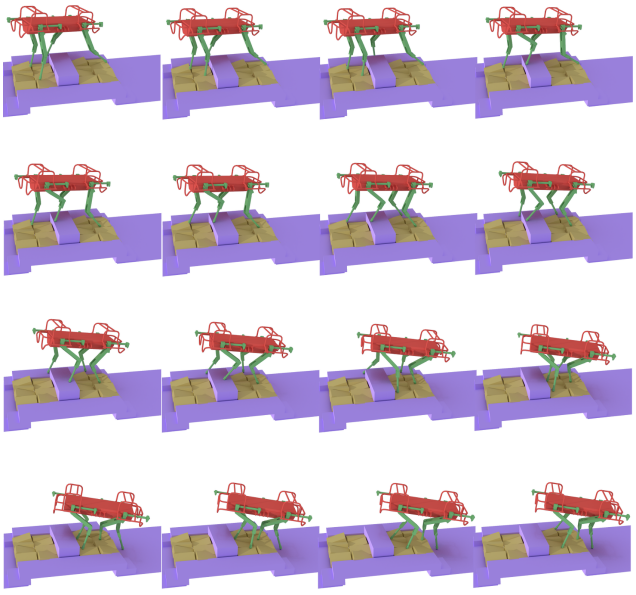

Figure 12. Robust crossing of rubbles by HyQ ($b_0 > 20$).

Observations: The difficulty of this scenario lies in the strong reduction of the reachable workspace induced by the extreme proximity of all obstacles. The planner is able to find a sequence, that consists in many steps. The proximity of the obstacles invalidate a large number of contact candidates because of collisions. To avoid breaking more than one contact between each step, the motion has to be decomposed into a large number of steps (61 in average). While this scenario is the slowest to solve, the planner still computes a solution interactively.

6.1.4 HyQ – Darpa-style rubble (Figure 12) The quadruped robot is given the task to 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.

Observations: In this context, setting up a really important minimum value for b_0 is possible due to the high stability of the HyQ robot, and results in more contact switches, in exchange for safety. The guide path-planning in this scenario takes a few seconds in average, more than in any other scenarios. This is explained by the necessity of discovering a safe way to "climb down" the rubble. In this part of the planning, the constraint that the 4 reachable workspaces of all legs must collide with the environment at all times is hard to respect, but enforces the equilibrium of

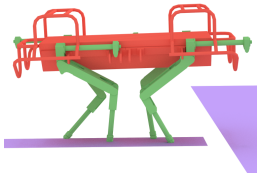

Figure 13. HyQ crossing a narrow bridge.

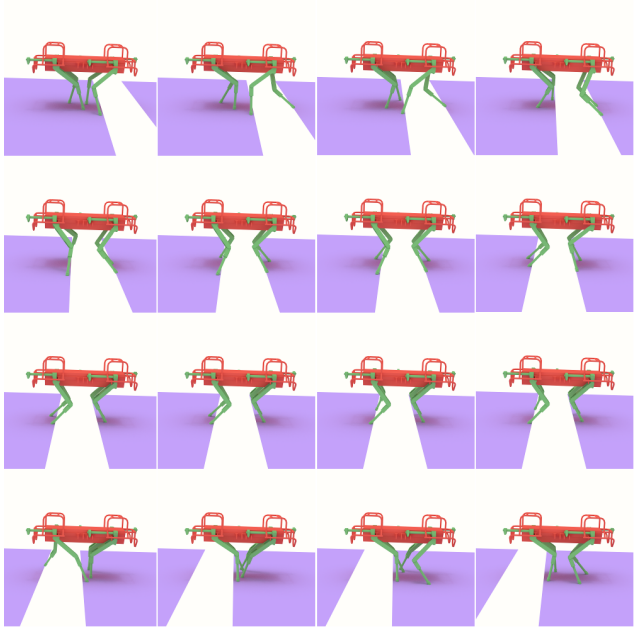

Figure 14. Crossing a hole contact sequence for HyQ ($b_0 > 4$).

HyQ. Again, the computation times remain however *interactive*.

6.1.5 HyQ – Obstacle race (Figure 13 and 14): In this long scene, HyQ is first required to cross a 55-cm large hole; then, to cross 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.

Observations: Despite the apparent simplicity of the scene, this scenario is a hard case for a contact planner. While finding a guide path above the hole is easy for the guide planner, finding a sequence of contacts that allows for equilibrium is not trivial. Second, the narrow bridge is hard both for the planner and the contact generator: to make sure that equilibrium is preserved along the traversal, the bridge must be approached with the appropriate angle. The difficulty is illustrated in Figure 14, where several feet rearrangements are required to cross the hole (although the video shows

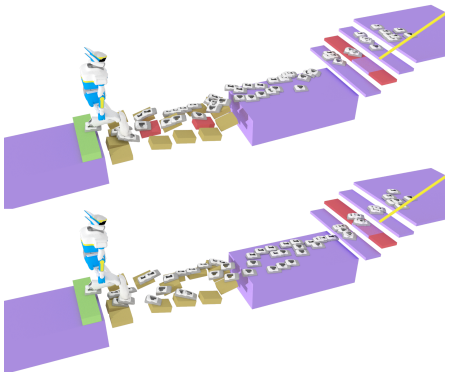

Figure 15. HRP-2 in the re-planning scenario. After the red step stones are removed, a new sequence of contacts is re-planned. Hand contacts are not presented here for readability.

this best). The planner however succeeds in finding a feasible sequence in the end, again with *interactive* computation times.

6.1.6 HRP-2 – Path re-planning (Figure 15): 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.
Observations: This scenario is designed to illustrate concretely the computation times of the planner. In the video, the footsteps indicating the contact sequence appear at the average speed of their computation (including the guide-path planning).

6.1.7 3-fingered hand – Manipulation of a pen (Figure 16): 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 16). 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 Influence of the parameters

This subsection discusses the several factors that influence the outcome of our planner: the root scaling factor s, the choice of heuristics for contact generation, the desired value of the robustness parameter b_0, and lastly, the discretization of the guide path.

Value of s	False positive	False negative
1	24%	0 %
1.1	12%	4%
1.15	7%	6%
1.2	3%	7.5%
1.25	2%	8.3%
1.5	1%	9.5%

Table 1. Percentage of true and false positive in the reachability condition, depending on the scaling value of W^0.

6.2.1 Choosing the scaling factor s: To find a convenient value s^* of s, we proceeded as follows. For several values of s, we generated 10000 configurations. We then computed the rates of false positives (i.e. the configuration satisfies the reachability condition, but does not belong to C^0_{Equil}) and false negatives (i.e. the configuration does not satisfy the reachability condition, but belongs to C^0_{Equil}).

The obtained results for HRP-2 are summarized 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). As it can be expected, the scaling results in a high increase of the false negatives, while the false positives decrease. For HRP-2 we decided to set $s^* = 1.2$, that results in less than 3 % of false positives, and was also effective for the car egress scenario.

6.2.2 On the choice of heuristics: In our conference paper, the computed motions were only generated using the EFORT heuristic. EFORT is designed for tasks requiring to exert important forces (such as pushing / pulling / climbing). Regarding 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 on the real robot turned out to be harder (Carpentier et al. 2016). 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, that results in less contact repositioning.

6.2.3 On the robustness equilibrium criterion: Robustness is really important when considering practical applications on the robot, to account for the various uncertainties that result from environment and state estimation. However, maximizing the robustness criterion is often not optimal, because the resulting configurations may be too conservative, thus not favoring the motion. In our experiments, we choose not to maximize the robustness, but to empirically set a robustness threshold value under which a configuration is not considered to be in static equilibrium (If no candidate

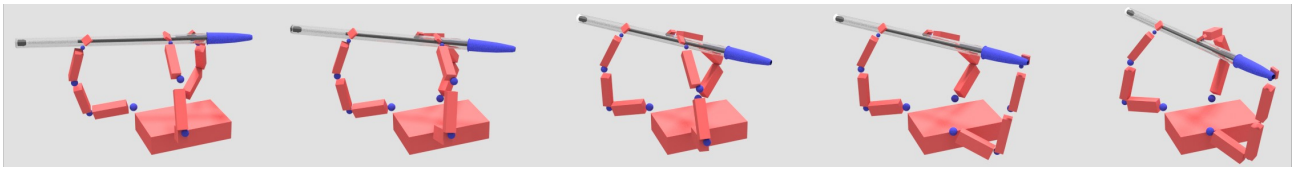

Figure 16. Contact sequence found for a pen manipulation in a zero gravity environment.

reaches the threshold, instead of failing, the algorithm can eventually return the "more robust" configuration found). Currently for HRP-2, a threshold value of 2 subjectively gives the best results in the considered scenarios, while 10 seems to be a good choice for HyQ. Both values do not significantly slow down the planning times.

6.2.4 Discretization of the guide path: Currently, to discretize the path, the user defines a fixed step size. The step size 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. For instance, in most scenarios the torso of HRP-2 moves about 15 cm between two postures, but only 3 cm for the car egress scenario, where we had to lower the step size because of the many contact breaks induced by the extremely constraining geometry. For future work, we would like to automatically adapt the size of the discretization step to the complexity of the planning, for instance by reducing the step size when failure occurs.

6.3 Performance analysis

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

6.3.1 Computation times: Table 2 summarizes the performance measurements obtained, in terms of computation times.

Considering the repartition of the computation time, for HRP-2, most of the time is 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. Therefore a large number of projections fail because of joint-limit violations.

For HyQ, there is a more uniform repartition of the times for the rubble scenario. For the obstacle scenario, we observe a large increase of the time spent performing collision checking and inverse kinematics. This is explained by the complexity of crossing the bridge: the robot has to stand high and create contacts close to each other because the bridge is narrow. In many cases this results in collisions, thus a large number of contacts have to be evaluated.

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.

6.3.2 Success rates: Table 3 summarizes the success rates obtained for each scenario. We observe that, as expected, our planner does not succeed systematically, because of the approximations made in our formulation. The extreme situation of the car egress scenario provides the less satisfying results in terms of kinematic failure. This is not surprising considering the narrowness of the scene. From a pragmatic point of view, regarding the computation times, we claim that our approach provides a satisfying compromise between completeness and efficiency. Indeed, the advantage of the framework is that when the contact generation fails, it does so rapidly, which allows us to rapidly re-plan a new contact sequence with a reasonable chance of success. The most efficient (and immediate) approach is probably to launch in parallel several instances of the planner for a given problem (our current implementation is single threaded) and to apply any successful result.

6.4 Comparison with previous work

Comparing our method with others is not trivial. Indeed, we focus on the contact planning aspect of the problem, and do not address the interpolation in this paper, nor provide a strong guarantee that our sequences can be interpolated. Most papers in the literature only provide the overall computation time, without specifying how much is spent in each phase. Moreover, there exists no off-the-shelf implementation of the related methods. Due to their complexity, we cannot afford the development resources required to reimplement them.

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 (61)	145 – **1055** – 15063	64 – **85** – 144	394 – **735** – 1422	3947 – **13069** – 36262	6919 – **21416** – 67031	**351**
Rubble (71)	151 – **15220** – 125867	242 – **511** – 3480	233 – **505** – 4564	180 – **414** – 3518	1548 – **18058** – 13126	**254**
Race (112)	267 – **966** – 2257	266 – **449** – 3956	824 – **1061** – 2130	666 – **874** – 1613	2797 – **4187** – 11292	**37**

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	Equilibrium feasibility	Kinematic failure	Equilibrium failure
Steep stairs	100%	99.5%	0.1%	0.4%
Standing up	68%	87.8%	6.1%	6.1%
Car egress	39%	77.0%	21.0%	2.0%
Rubble	74%	97.9%	0.1%	2.0%
Obstacle race	58.0%	95.7%	1.8%	2.5%

Table 3. Success rates for each scenario, rounded to the first decimal. The first value indicates the percentage of successful complete contact plannings for 1000 tests; The second value indicates the percentage of equilibrium feasible root configurations: considering each limb individually, indicates the percentage of root configurations of the guide path that led to a feasible contact. Kinematic failure is the percentage of contact generations that failed because no collision-free candidate was found. Equilibrium failure is the percentage of contact generations that failed because no candidate respected the static equilibrium condition.

Scenario	Method	Guide Path	Contact sequence	Interpolation
Stair 20 cm	Hauser et al.	5.42 min		
	Ours + Carpentier et al.	**5 ms**	< **18 ms**	< 2s
	Mordatch et al.	2 to 10 min		
Stair 30 cm	Hauser et al.	4.08 min		
	Ours	**5 ms**	< **18 ms**	X
	Mordatch et al.	2 to 10 min		
Stair 40 cm	Hauser et al.	10.08 min		
	Ours	**5 s**	**3 s**	X
	Mordatch et al.	2 to 10 min		
Table (car) egress	Bouyarmane et al.; Escande et al.	10 min	3.5 hours	
	Ours	**529 ms**	< **22 s**	X

Table 4. Comparison between the computation times obtained by our method and previous ones, for each phase of the planning. Blue cells indicate merged times (because no precise time information is available). The X symbol indicates scenarios where we did not address the interpolation of the contact sequence.

Furthermore, the interpolation itself is not completely solved. For instance in Escande et al. (2008), the end-effector trajectories are computed automatically with splines, but are manually corrected if they enter in collision. In (Mordatch et al. 2012), internal joint collisions can occur in the resulting motion. Recent methods suffer from similar issues (Carpentier et al. 2016).

Nonetheless, we indicate in Table 4 what is actually computed by each method and in how much time. We compare motions obtained in previous works with HRP-2 (or a humanoid avatar) with ours. The two scenarios we selected consist in: climbing one step,

of height 20, 30 or 40 centimeters; getting out of a table, that we consider of a complexity similar to the car egress scenario (or inferior since the feet locations remain on the ground in the former case). For scenarios that apply, we also indicate the interpolation time that we obtained by feeding the plan to our interpolation method presented in Carpentier et al. (2016).

Despite the difficulty of comparing the methods, we believe the results presented in Table 4 indicate that our planner holds the promise of planning complete trajectories significantly faster than previous works.

7 Discussion and future work

In this paper we consider the *cluttered* 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_{reach}, an approximation of the space of *equilibrium feasible* root configurations. Thanks to the computationally efficient verification of the *reachability condition*, 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 take into account criteria to optimize user-defined properties.

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 arbitrary multiped robots in *cluttered* problems. The only prerequisite is the specification of the volumes W^0. **Regarding performance**, our framework is really efficient in addressing both $\mathcal{P}_1$ and $\mathcal{P}_2$. This results in *interactive* computation times. **Regarding the quality of the paths**, the *reachability condition* allows us to compute *equilibrium feasible* paths in all the presented scenarios, with low rejection 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 replanning in case of failure. Furthermore because of this approximation, the guide search is not complete. The choice is deliberate, because we are convinced 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.

Another limitation of the method is that it currently only applies to *cluttered* problems. However, it should first be noted that many problems belong to this class: for instance all the problems at the DRC were *cluttered*. Our method is thus already useful for many cases. One way of extending its range of application, that 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 candidates surfaces, we can use them to verify the equilibrium criterion. This would give us a necessary condition for *equilibrium feasibility*.

Regarding the interpolation between the contact sequences ($\mathcal{P}_3$), we have already obtained some success for some of the computed sequences (Carpentier et al. 2016), but additional work is required on the planning side to obtain a seamless workflow. To achieve this, we are currently working on the notion of transition certificate, i.e. formulating conditions that guarantee that the interpolation between two contact configurations is dynamically feasible. A last limitation of our method is that only static equilibrium configurations are considered for contact planning. We aim at performing kinodynamic planning to overcome this limitation. 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 last issues is essential to bridge the gap between the planning and control aspects of multiped 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.

Glossary

$C_{Contact}$ Set of collision free full body configurations such that at least one end-effector is in contact with the environment. 7, 17

$C^0_{Contact}$ Set of *contact feasible* root configurations $\mathbf{q}^0$, that verify $\exists \mathbf{q}^{\bar{0}}, \mathbf{q}^0 \oplus \mathbf{q}^{\bar{0}} \in C_{contact}$. 6, 7, 17, 18

C_{Equil} Set of collision free full body configurations that are in static equilibrium. 8, 17

C^0_{Equil} Set of *equilibrium feasible* root configurations $\mathbf{q}^0$, that verify $\exists \overline{\mathbf{q}^0}, \mathbf{q}^0 \oplus \overline{\mathbf{q}^0} \in C_{Equil}$. 6, 7, 14, 17

C^0_{reach} Set of root configurations $\mathbf{q}^0$ that verify the *reachability condition*. 7, 17

cluttered A class of contact planning problems that admit as a solution a sequence of contact configurations for which at least one contact occurs with a *quasi-flat* surface. 3–6, 16, 17

contact feasible Refers to a root configuration that can be extended into a collision free contact configuration. 4–7, 17, 18

equilibrium feasible Refers to a root configuration that can be extended into a collision free configuration in static equilibrium. 2–7, 9, 17

interactive We say that a planning method is interactive when the computation time for one step is lesser than the time to execute it. We arbitrarily approximate this time to one second. 3, 4, 7, 11–13, 15, 17

quasi-flat refers to a contact surface for which the friction cone includes the direction opposite to the gravity. 3, 17

reachability condition A root configuration $\mathbf{q}^0$ that verifies :
$W(\mathbf{q}^0) \cap O \neq \emptyset$ and $W^0_{s^*}(\mathbf{q}^0) \cap O = \emptyset$, where s^* is a user-defined value. 7, 17

References

Amato NM, Bayazit OB, Dale LK, Jones C and Vallejo D (2000) Choosing good distance metrics and local planners for probabilistic roadmap methods. *IEEE T. Robotics and Automation* 16(4): 442–447.

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.

Ben-Israel A and Greville TN (2003) *Generalized inverses: theory and applications*, volume 15. Springer Science & Business Media.

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.

Bouyarmane K and Kheddar A (2011) Multi-Contact Stances Planning for Multiple Agents. In: *Proc. of IEEE Int. Conf. on Robot. and Auto (ICRA)*. Shanghai, China.

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.

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, 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.

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.

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*.

Kajita S, Kanehiro F, Kaneko K, Fujiwara K, Harada K, Yokoi K and Hirukawa H (2003) Biped Walking Pattern Generation by using Preview Control of Zero-Moment Point. In: *Proc. of IEEE Int. Conf. Robot. and Auto (ICRA)*. Taipei, Taiwan.

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.

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.

Semini C, Tsagarakis NG, Guglielmino E, Focchi M, Cannella F and Caldwell DG (2011) Design of hyq - a hydraulically and electrically actuated quadruped robot. *IMechE Part I: Journal of Systems and Control Engineering* 225(6): 831–849.

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.

Zucker M, Bagnell JA, Atkeson CG and Kuffner J (2010) An optimization approach to rough terrain locomotion. In: *Proc. of IEEE Int. Conf. on Robot. and Auto (ICRA)*. Anchorage, Alaska, pp. 3589–3595.

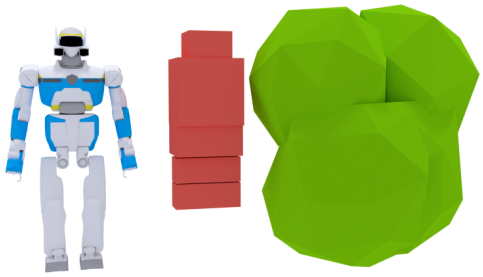

Figure 17. The W volumes computed for HRP-2. The red shapes are W^0. The green shapes represent the W^k.

Figure 18. Different approximations of the range of motion of the right arm 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.

A Generating the W volumes for HRP-2

In this Appendix, we detail our method to generate the volumes W used in RB-RRT, with the example of HRP-2.

We consider the robot HRP-2, and proceed to the decomposition of its kinematic chain 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 17.

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);
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 18 illustrate the obtained W^k for HRP-2. Regarding the procedure, we can see that step 2 is conservative (Figure 18–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 feasible*, as illustrated in Figure 3, 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 18–middle) instead of a minimum encompassing shape (Figure 18–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 3.

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 17 – 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 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($s0$,$\mathbf{Q}^0$, 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: $s =$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.

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** $NULL$

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.

GENERATECONTACT($\mathbf{q}^0, k$) is a call to the contact generator presented in Section 4.1. 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.

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
```

C Derivation of the manipulability ellipsoid

Again, we assume that $\mathbf{J}$ is full rank. We discard the k indices, and write the pseudo-inverse of $\mathbf{J}$ as $\mathbf{J}^\dagger$.

$$\begin{aligned} \dot{\mathbf{p}} &= \mathbf{J}\dot{\mathbf{q}} \\ \dot{\mathbf{q}} &= \mathbf{J}^\dagger \dot{\mathbf{p}} \\ \dot{\mathbf{q}}^T &= \dot{\mathbf{p}}^T \mathbf{J}^{\dagger T} \\ \dot{\mathbf{q}}^T \dot{\mathbf{q}} &= \dot{\mathbf{p}}^T \mathbf{J}^{\dagger T} \mathbf{J}^\dagger \dot{\mathbf{p}} \end{aligned}$$

Then, the equality $\mathbf{J}^{\dagger T}\mathbf{J}^\dagger = (\mathbf{J}\mathbf{J}^T)^{-1}$ follows from the SVD decomposition of each term (Ben-Israel and Greville 2003).

D Source code of our planner

Our planner is implemented using the Humanoid Path Planner (HPP) software. 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. Contrary to the mature HPP, at the time of submitting this article, our planner is not officially released, mostly for lack of a complete documentation and a tutorial.

However a reader familiar with HPP should be able to use the library. Our goal is to make our code useful for the community, and we are working on releasing a stable version of our planner as soon as possible.