A Continuous Approach to Legged Locomotion Planning

Nicolas Perrin*, Christian Ott†, Johannes Englsberger†, Olivier Stasse‡, Florent Lamiraux‡

Abstract

Legged locomotion planning can often be reduced to the search of finite sequences of contacts between the robot and the ground. We address this problem, using an original, continuous approach. We first demonstrate how this intrinsically discrete problem can be made continuous, and then present two applications: an experiment of vision-based reactive footstep planning with the DLR-Biped robot, and a simulation of non-gaited locomotion planning with a hexapod robot.

1 Introduction

The general problem of locomotion planning for legged robots takes place in a stratified configuration space. When a contact with the ground is made or released, the system moves from a submanifold of the configuration space to another one, and the equations of motion change in a discrete manner. As pointed out in [Goodwine, 1997], this discontinuous nature is one of the most important characteristic of legged robots (and other "stratified systems").

In an attempt to solve the locomotion planning problem in a computationally efficient way, simplified models are often used, and they should also have a discontinuous nature, at least to some extent. Completely continuous models have been used in previous work (see [Yoshida et al., 2008], [Kanoun et al., 2011], [Dalibard et al., 2011]), but they always fail to capture some important features of the legged robots, such as for example their ability to step over obstacles.

In [Goodwine, 1997], small-time local controllability is obtained for some stratified systems thanks to an extension of the Chow-Rashevskii theorem ([Bullo and Lewis, 2004], [Sastry, 1999]). As a result, some classic algorithms such as [Lafferriere and Sussmann, 1991] can be applied to locomotion planning for legged robots (see also [Harmati and Kiss, 2001]), but it essentially consists of converting initial sliding motions into feasible walking motions. Again, this means that the stepping over abilities of the robot are not well captured.

Continuous models are however interesting because they make possible the use of conventional and efficient motion planning algorithms that suppose smooth configuration spaces (e.g. PRM [Kavraki et al., 1996], RRT [LaValle and Kuffner, 2000]). Up to now, the state-of-the-art solution to obtain fast algorithms that take into account the discontinuous nature of the problem is to *choose footsteps before computing motions*. Depending on the context and on the robot, a heuristic is defined and used to search for finite sequences of footsteps. Once a sequence of footsteps has been constructed, the next phase is to find a feasible continuous motion of the robot that follows this sequence. This kind of approach amounts to dealing with the discrete and continuous natures of the problem separately. It has been used extensively for humanoid robot navigation planning ([Kuffner et al., 2001], [Bourgeot et al., 2002], [Chestnutt et al., 2003], [Chestnutt et al., 2005], [Gutmann et al., 2005]), for locomotion plan-

*Department of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego, 30, 16163 Genova, Italy

†Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Wessling, Germany

‡CNRS/LAAS, Université de Toulouse UPS, INSA, INP, ISAE, Toulouse, France

ning for a hexapod [Hauser et al., 2006], or for more general multi-contact planning problems [Bouyarmane and Kheddar, 2011] (in that case, instead of footsteps, sequences of *stances* are searched for). To search for finite sequences of footsteps, as conventional motion planning algorithms cannot be applied, graph search algorithms such as A* are used instead.

In this paper, we present an original approach that enables us to take into account the discrete aspects of the problem while doing the motion planning in a smooth configuration space. Our work shares similarities with [Boissonnat et al., 2000] where a connection is established between the free space of a spider robot and the free space of a half-disk robot moving by translation and rotation amidst obstacles. In this latter work however, the goal was to solve exactly the motion planning problem for the spider. Our goal is different since we aim at efficiently using sampling-based motion planning algorithms through the definition of weak collision-freeness. Our work is also more general than [Boissonnat et al., 2000].

In Section 2, we define the problem of "flea motion planning" and show an efficient algorithm which serves as an example to illustrate our approach. In Section 3, we demonstrate a general theorem that can be used to transform discrete motion planning problems into continuous ones. In Section 4 and Section 5, we apply this theorem to perform vision-based reactive footstep planning with the DLR-Biped robot, and give also a thorough description of our software architecture. The planning is done on a discretized heightmap of the environment. This implementation gives strong connections between our work and [Eldershaw and Yim, 2001] where the complexity of the task of planning steps for a legged robot is also reduced by using first a continuous high-level planner. The main advantage of our work over [Eldershaw and Yim, 2001] is the theoretical soundness of our equivalence between the existence of high-level and "foot-level" paths. In Section 6, we show that the same approach can be easily adapted to locomotion planning for a hexapod robot. Finally, in Section 7, we conclude and discuss about future work.

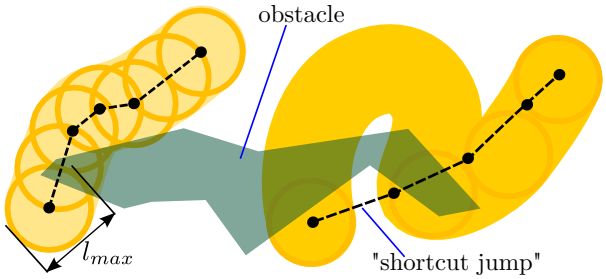

Figure 1: The "flea motion planning problem". On the left: from a collision-free sequence of flea jumps to a continuous "weakly collision-free" path for the disk. On the right: converting a continuous weakly collision-free path of the disk into a sequence of flea jumps, using a greedy algorithm.

2 An efficient algorithm for flea motion planning

In this section we consider the simple example of flea motion planning, which has been introduced in [Perrin et al., 2012] to illustrate the method used in [Perrin et al., 2011] and [Perrin et al., 2012] to convert footstep planning into classical continuous motion planning.

The flea is represented by a point in a 2D environment; $\mathcal{C} = \mathbb{R}^2$ is the configuration space. There are obstacles in this 2D environment such that the free space $\mathcal{F}$ is an open set. The Boolean-valued function $\kappa : \mathcal{C} \to \{\texttt{true}, \texttt{false}\}$ of collision-freeness in $\mathcal{C}$ is defined for any $s \in \mathcal{C}$ by $\kappa(s) \Leftrightarrow s \in \mathcal{F}$. The flea can make jumps in any direction and of any length strictly less than $l_{max} > 0$. The goal is to find a sequence of jumps from a location $(x_A, y_A) \in \mathcal{F}$ to a location $(x_B, y_B) \in \mathcal{F}$ such that every intermediate configuration of the flea is in $\mathcal{F}$. The discontinuous nature of this flea motion planning problem is similar to the discontinuous nature of the problem of finding sequences of footsteps for legged robots.

To solve this problem efficiently, we first prove an equivalence between the discontinuous motion of the flea, and the continuous motion of an open disk, but with a new notion of collision-freeness. So, let us assume that a sequence of jumps has been found,

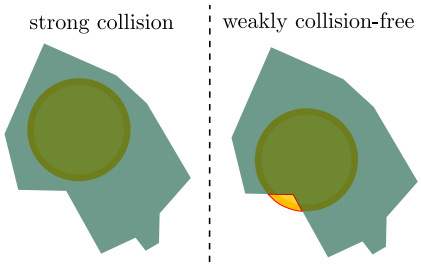

Figure 2: Weak collision-freeness.

and that it corresponds to the sequence of configurations $p_1 = (x_1, y_1), p_2, p_3, \ldots, p_n = (x_n, y_n)$, with $(x_1, y_1) = (x_A, y_A)$ and $(x_n, y_n) = (x_B, y_B)$. We consider the continuous motion $s : [0, 1] \mapsto \mathbb{R}^2$ of an open disk of diameter l_{max} defined by linear interpolation between each (x_i, y_i) and (x_{i+1}, y_{i+1}) for $i = 1, 2, \ldots, n-1$, as depicted in Figure 1 (on the left). An interesting property of this continuous disk motion is the following (it is a direct consequence of the upper bound l_{max} on the length of jumps):

Property 2.1. *For all $t \in [0, 1]$, the configuration $s(t)$ of the disk contains at least one of the flea configurations $p_1, p_2, \ldots, p_n$.*

This property suggests the definition of a new notion of collision-freeness:

Definition 2.1. *We say that a disk configuration (x, y) is collision-free if there exists at least one flea configuration (i.e. point) inside the disk which is collision-free. We call this new notion of collision-freeness the "weak collision-freeness", and say that the disk configuration is "weakly collision-free". Conversely, if all the flea configurations inside the disk are in collision (i.e. the disk does not intersect the free space), we say that the disk is in "strong collision". We say that a continuous path of disks configurations is weakly collision-free if all the configurations along that path are weakly collision-free.*

Figure 2 illustrates this definition.

We can reformulate Property 2.1:

Theorem 2.1. *If there exists a finite sequence of collision-free jumps from (x_A, y_A) to (x_B, y_B), then there also exists a weakly collision-free continuous path of the open disk of diameter l_{max} from (x_A, y_A) to (x_B, y_B).*

We prove that the converse is true:

Theorem 2.2. *If there exists a weakly collision-free continuous path of the open disk of diameter l_{max} from (x_A, y_A) to (x_B, y_B), then there exists a finite sequence of collision-free jumps from (x_A, y_A) to (x_B, y_B).*

Proof. Intuitively, the reason for this theorem to hold is that since the path is weakly collision-free, the free space intersects every configuration of the disk along that path. And because the disk is of diameter l_{max}, it follows that in order to progress along the intersection between the free space and the area swept by the disk, the flea never has to make jumps larger than l_{max}, and therefore it can go from one end of the path to the other.

Let us denote by d_2 the Euclidean distance in $\mathbb{R}^2$, and by $\mathcal{D}_{(x,y)}$ the open disk of center (x, y) and diameter l_{max}. For a point $s \in \mathcal{C}$ we denote by $d_{obs}(s) = \inf\{d_2(s, o) | o \in \mathcal{C} \setminus \mathcal{F}\}$ its distance to the obstacles. For a configuration (x, y) of the disk we define: $\delta_{obs}(\mathcal{D}_{(x,y)}) = \sup\{d_{obs}(s) | s \in \mathcal{D}_{(x,y)}\}$. A configuration (x, y) is weakly collision-free if and only if $\delta_{obs}(\mathcal{D}_{(x,y)}) > 0$. Let us consider a weakly collision-free continuous path $s : [0, 1] \mapsto \mathbb{R}^2$ from (x_A, y_A) to (x_B, y_B). We pose $d_{\inf} = \frac{1}{2} \inf\{\delta_{obs}(\mathcal{D}_{s(t)}) | t \in [0, 1]\}$. By continuity of $t \mapsto \delta_{obs}(\mathcal{D}_{s(t)})$, we have $d_{\inf} > 0$. By uniform continuity of s, there exists $0 < \epsilon < 1$ such that $\forall t \in [0, 1-\epsilon]$, $d_2(s(t), s(t+\epsilon)) < \min(d_{\inf}, l_{max})$.

Let us now consider $t \in [0, 1-\epsilon]$ and a collision-free configuration s of the flea in $\mathcal{D}_{s(t)}$. First, we know that there exists $s' \in \mathcal{D}_{s(t)}$ such that $d_{obs}(s') > d_{\inf}$. Besides, since $\mathcal{D}_{s(t)}$ is of diameter l_{max}, we have $d_2(s, s') < l_{max}$, and thus the flea can jump from s to s'. Then, since we have $d_2(s(t), s(t+\epsilon)) < \min(d_{\inf}, l_{max})$, there exists $s'' \in \mathcal{D}_{s(t+\epsilon)}$ such that $d_2(s', s'') < \min(d_{\inf}, l_{max})$. It follows that s'' is collision-free, and the flea can jump from s' to s''. So we have proved that if s is a collision-free configuration of the flea in $\mathcal{D}_{s(t)}$ with $t \in [0, 1-\epsilon]$, it is always possible to reach a collision-free configuration in $\mathcal{D}_{s(t+\epsilon)}$ with at most 2 jumps. By iteration, we deduce that a collision-free configuration $s_\alpha \in \mathcal{D}_{s(1)}$ can

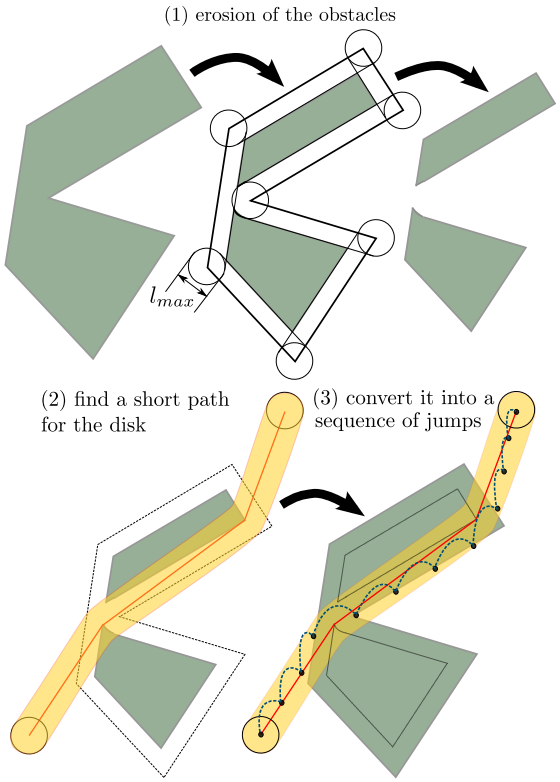

Figure 3: An efficient algorithm for flea motion planning.

be reached after a finite number of jumps. We have $d_2(s_\alpha, (x_B, y_B)) < l_{max}$, and therefore the flea can jump directly from s_α to (x_B, y_B). This concludes the proof, and an example of sequence of jumps obtained from a weakly collision-free continuous path of the disk can be seen on the right side of Figure 1. □

Together, Theorem 2.1 and Theorem 2.2 form an equivalence between the weakly collision-free paths of the disk and the collision-free sequences of jumps of the flea. It turns out that this equivalence gives an efficient algorithm to solve the flea motion planning problem. Indeed, instead of looking for a discontinuous sequence of jumps, we can first look for a continuous path of the disk, and that can be done with any conventional motion planning algorithm, provided that we implement new collision checks using Definition 2.1. Besides, in the case of flea motion planning these "weak collision" checks are very easy to implement with a morphological operation of erosion of the obstacles by an open sphere of radius $\frac{l_{max}}{2}$ (see [Serra, 1983]): the disk $\mathcal{D}_{(x,y)}$ is in strong collision with the obstacles if and only if its center (x, y) is inside the eroded obstacles. So, once the eroded obstacles are obtained, we can use any classical motion planning algorithm to find a short weakly collision-free path for the disk. To actually convert this continuous path into a finite sequence of jumps, we can apply the greedy approach already used in [Perrin et al., 2012] which consists in repeatedly trying to jump from the current disk $\mathcal{D}_{s(t)}$ to a disk $\mathcal{D}_{s(t')}$ with t' as large as possible and obtained by dichotomy. This can result in "shortcut jumps", as shown in Figure 1 (on the right). Figure 3 illustrates the whole method with a simple example where the obstacles are polygonal (in that case we can actually find the shortest path for the disk in polynomial time: see [de Berg et al., 2000], ch. 15).

Using such a transfer towards continuous motion planning has several benefits.

For the flea motion planning problems, other more direct approaches are also very efficient, and for example it would be easy to naively adapt the RRT algorithm to grow trees of discrete sequences of jumps. But for more complicated problems such as footstep planning, adapting RRT is not an easy task because it is not always clear how to extend a node *towards* a sample configuration. For this reason the current state-of-the-art method is to choose in advance a finite set of possible steps and then use A*-like algorithms to search for sequences of steps towards a goal ([Kuffner et al., 2001], [Bourgeot et al., 2002], [Chestnutt et al., 2003], [Chestnutt et al., 2005]). Although adjustments have been considered (see [Chestnutt et al., 2007]), starting by manually selecting a finite number of possible steps is not very satisfying and leads to several problems such as limited stepping capabilites. The approach we present in sections 4 and 5 is based on a similar equivalence as the one formed by theorems 2.1 and 2.2, and it enables to deal with fully continuous stepping capabilities in an efficient

and theoretically sound way.

For a hexapod, a large set of possible steps would be required in order to obtain decent stepping capabilities, and therefore methods based on A*-like algorithms are difficult to apply. This problem can be circumvented by using fixed gaits, but they are less efficient in complex environments. The method we propose in Section 6 is again based on an equivalence that transforms the problem into a fully continuous one, and it enables to very quickly find sequences of steps even in quite complex environments.

We have also implemented our continuous approach (with RRT used to plan continuous paths) for some random instances of 2D flea motion planning, and after comparison the results obtained indicate that a proper implementation of our approach has the potential to outperform a naive adaptation of RRT, especially in simple environments. What's more, since in our approach each edge of the tree grown by the RRT algorithm represents not one, but a whole set of potential sequences of jumps, the trees obtained are significantly smaller than trees of similar covering obtained with a naive adaptation of RRT, and this can be very useful for operations such as nearest neighbor search or tree transformations.

3 An equivalence between some discrete and continuous motion planning problems

In this section we demonstrate a general result that can be applied in particular to flea motion planning to obtain the equivalence between the discrete sequences of jumps and the continuous weakly collision-free paths of the disk. First, let us introduce some notation and definitions.

We will only consider metric configuration spaces, and denote by $dist()$ their distance functions. Let $\mathcal{C}$ be a metric configuration space. For $s \in \mathcal{C}$ and $r > 0$, we denote by $\mathring{\mathcal{S}}(s, r)$ the open sphere of center s and radius r. $\mathcal{F}$, the free space, is always assumed to be an open subset of $\mathcal{C}$. We denote by κ the Boolean-valued function of collision-freeness: $\mathcal{F} = \{s \in \mathcal{C} | \kappa(s)\}$. We denote by d_{obs} the function from $\mathcal{C}$ to $\mathbb{R}^+$ defined by $d_{obs}(s) = \inf\{dist(s, o) | o \in \mathcal{C} \setminus \mathcal{F}\}$, and for any non-empty bounded set $E \subset \mathcal{C}$, we define $\delta_{obs}(E) = \sup\{d_{obs}(s) | s \in E\}$.

We now give a definition of discrete and continuous motion planning problems. It is by no means a general definition, but we will only consider this type of problems.

Definition 3.1 (continuous motion planning problems)**.**
INPUT*: $\mathcal{C}$, κ: mapping from $\mathcal{C}$ to $\{\texttt{true}, \texttt{false}\}$, $s_i \in \mathcal{F}$ and $s_f \in \mathcal{F}$.*
OBJECTIVE*: find a continuous path $(s(t))_{t\in[0,1]}$ such that $\forall t \in [0, 1]$, $\kappa(s(t))$ (we call "valid" such a continuous path), and such that $s(0) = s_i$ and $s(1) = s_f$.*

We will also consider slight variants of these problems where the initial and final configurations are not fixed but must simply belong to some sets.

For discrete motion planning, there is an additional relation R that defines a relationship between consecutive configurations.

Definition 3.2 (discrete motion planning problems)**.**
INPUT*: $\mathcal{C}$, κ: mapping from $\mathcal{C}$ to $\{\texttt{true}, \texttt{false}\}$, R: mapping from $\mathcal{C} \times \mathcal{C}$ to $\{\texttt{true}, \texttt{false}\}$, $s_i \in \mathcal{F}$ and $s_f \in \mathcal{F}$.*
OBJECTIVE*: find a finite sequence of configurations $(s_1, s_2, \ldots, s_n)$ such that $\forall k \in \{1, \ldots, n-1\}$, $\kappa(s_k)$, $\kappa(s_{k+1})$, and $R(S_k, S_{k+1})$ (we call "valid" such a finite sequence), and such that $s_0 = s_i$ and $s_n = s_f$.*

Remark: in the rest of the paper, without ambiguity we will use the adjective "valid" to denote various types of "acceptable" configurations, paths or sequences. It will sometimes just mean "collision-free" or "weakly collision-free", and sometimes a variant of one of these notions, plus possibly additional restrictions.

The objective of the equivalence we are about to prove is to convert discrete motion planning problems into equivalent continuous motion planning problems, but not on the same configuration space, and with a different notion of collision-freeness.

The equivalence is stated below with Theorem 3.1, which has been introduced and proven

in [Perrin, 2012] with slightly different hypotheses. Another more general equivalence is studied in [Perrin, 2012], but it has less direct applications. Similar equivalence theorems between discrete and continuous motion planning problems are not common in the literature, but we can mention [Alami et al., 1994] where a reduction property shows that for some class of manipulation problems, the existence of a solution path with discrete "grasp" and "release" events is equivalent to the existence of a path where the grasp is continuously modified.

So, let us consider a discrete motion planning problem, entirely defined by the metric configuration space $\mathcal{C}$, its free space $\mathcal{F}$, the relation R and s_i and s_f.

We denote by $\mathcal{P}(\mathcal{C})$ the set of subsets of $\mathcal{C}$. We assume that there exists another metric configuration space Ω and a function $f : \Omega \to \mathcal{P}(\mathcal{C})$ such that the five following properties are verified:

1. The function $\delta_{obs} \circ f$ is continuous.
2. For all $\epsilon > 0$ there exists $\eta > 0$ such that $\forall \varphi, \varphi' \in \Omega$ verifying $dist(\varphi, \varphi') < \eta$, the set $f(\varphi) \cap f(\varphi')$ is non-empty and it intersects all the spheres $\mathring{\mathcal{S}}(s, \epsilon)$ such that $s \in f(\varphi)$.
3. $\forall \varphi \in \Omega$, $f(\varphi)$ is such that $\forall (s, s') \in f(\varphi)^2$, $R(s, s')$.
4. For any $(s, s') \in \mathcal{C}^2$ such that $R(s, s')$, there exists $\varphi \in \Omega$ such that $s \in f(\varphi)$ and $s' \in f(\varphi)$.
5. $\forall (s, s', s'') \in \mathcal{C}^3$ such that $R(s, s')$ and $R(s', s'')$, and $\forall \varphi_0 \in \Omega$ such that $s \in f(\varphi_0)$ and $s' \in f(\varphi_0)$, there exists a continuous path from φ_0 to a configuration φ_1 verifying $s'' \in f(\varphi_1)$, such that for any configuration φ along this path, we have $s' \in f(\varphi)$. This property is illustrated in Figure 4.

Example 3.1. *In the case of the flea motion planning problem of Section 2, $\Omega = \mathcal{C} = \mathbb{R}^2$, $f : s \mapsto \mathcal{D}_s$ satisfy the above properties.*

Intuitively, f is the crucial function that transfers the viewpoint from states that move discretely in a continuous space $\mathcal{C}$ to sets of potential states whose

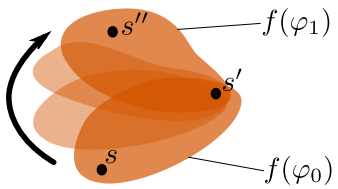

Figure 4: A continuous path from φ_0 to φ_1.

"continuous" motions in $\mathcal{P}(\mathcal{C})$ are governed by states moving continuously in Ω. Depending on the discrete problem, finding such a function f and a proper space Ω can be more or less difficult, and although some insight is given in [Perrin, 2012], it is beyond the scope of the present paper.

The aim of the five properties of f is to obtain the same kind of equivalence as the one formed by theorems 2.1 and 2.2 of Section 2. They are needed in the proofs of Theorem 3.2 and Theorem 3.3.

On Ω, we define a new notion of collision-freeness κ_Ω:

Definition 3.3 (κ_Ω)**.** *$\varphi \in \Omega$ verifies $\kappa_\Omega(\varphi)$ if and only if the intersection between $f(\varphi)$ and the free space is non-empty, i.e. $\exists s \in f(\varphi)$ such that $\kappa(s)$.*

Under the above assumptions, the following theorem holds:

Theorem 3.1. *There exists a valid finite sequence from s_i to s_f in $\mathcal{C}$ if and only if there exists a continuous path $(\chi(t))_{t\in[0,1]} \in \Omega^{[0,1]}$ such that $s_i \in f(\chi(0))$, $s_f \in f(\chi(1))$, and $\forall t \in [0, 1]$, $\kappa_\Omega(\chi(t))$.*

This theorem is an equivalence that generalizes the one formed by theorems 2.1 and 2.2. We prove the two implications of this equivalence in the next two sections (theorems 3.2 and 3.3).

3.1 From a valid discrete sequence in $\mathcal{C}$ to a valid continuous path in Ω

Theorem 3.2. *If there exists a valid sequence $(s_1 = s_i, s_2, \ldots, s_n = s_f)$, then there exists a valid continuous path $(\chi(t))_{t\in[0,1]}$ such that $s_1 \in f(\chi(0))$, $s_2 \in f(\chi(0))$, $s_{n-1} \in f(\chi(1))$ and $s_n \in f(\chi(1))$.*

Proof. We prove this implication by induction on n, the size of the valid sequence. For $n = 2$, we have $R(s_i, s_f)$, and there exists $\varphi \in \Omega$ such that $s_i \in f(\varphi)$ and $s_f \in f(\varphi)$. The stationary path such that $\forall t \in [0,1], \chi(t) = \varphi$, is valid.

Let us now assume that the result is true for any sequence of size $n \geq 2$, and consider a valid sequence of size $n+1$: $(s_1 = s_i, s_2, \ldots, s_{n+1} = s_f)$. Let $(\chi(t))_{t\in[0,1]} \in \Omega^{[0,1]}$ be a valid path such that $s_1 \in f(\chi(0))$, $s_2 \in f(\chi(0))$, $s_{n-1} \in f(\chi(1))$ and $s_n \in f(\chi(1))$. We have $R(s_{n-1}, s_n)$ and $R(s_n, s_{n+1})$, so there exists a continuous path $(\vartheta(t))_{t\in[0,1]} \in \Omega^{[0,1]}$ from $\chi(1)$ to a configuration $\varphi_f \in \Omega$ verifying $s_{n+1} \in f(\varphi_f)$, such that for any configuration ϑ along this path, we have $s_n \in f(\vartheta)$, and thus $\kappa_\Omega(\vartheta)$. Appending this path to the path $(\chi(t))_{t\in[0,1]}$ gives us a valid continuous path, and this conludes the proof of Theorem 3.2. □

3.2 From a valid continuous path in Ω to a valid discrete sequence in $\mathcal{C}$

Theorem 3.3. *If there exists a valid continuous path $(\chi(t))_{t\in[0,1]} \in \Omega^{[0,1]}$ with $s_i \in \chi(0)$ and $s_f \in \chi(1)$, then there exists a valid sequence $(s_1 = s_i, s_2, \ldots, s_n = s_f)$.*

Proof. The second property of f implies that for any $\epsilon > 0$, there exists $N \in \mathbb{N}^*$ such that the sequence $(\chi(0/N), \chi(1/N), \ldots, \chi(N/N))$ verifies the following property: $\forall k \in \{0, \ldots, N-1\}$, for any configuration $s \in f(\chi(k/N))$, the sphere $\mathring{\mathcal{S}}(s, \epsilon)$ intersects the non-empty set $f(\chi(k/N)) \cap f(\chi((k+1)/N))$. Besides, since $\mathcal{F}$ is an open set, we have: $\forall t \in [0,1]$, $\delta_{obs}(f(\chi(t))) > 0$. And $t \mapsto \delta_{obs}(f(\chi(t)))$ is continuous, so there exists $d_{\text{inf}} > 0$ such that $\forall t \in [0,1]$, $\delta_{obs}(f(\chi(t))) > d_{\text{inf}}$. It follows that for ϵ small enough, for all $t \in [0,1]$ there exists $s_t \in f(\chi(t))$ collision-free and such that the sphere $\mathring{\mathcal{S}}(s_t, \epsilon)$ is entirely inside $\mathcal{F}$. We thus deduce that all the sets $f(\chi(k/N)) \cap f(\chi((k+1)/N))$ have a non-empty intersection with $\mathcal{F}$. Using the property that two elements s, s' of the same set $f(\chi(t))$ are always such that $R(s, s')$, we can construct a valid sequence $(s_1, s_2, \ldots, s_{N+2})$ such that $s_1 = s_i$, $s_{N+2} = s_f$, and $\forall k \in \{2, \ldots, N+1\}$, $s_k \in f(\chi((k-2)/N)) \cap f(\chi((k-1)/N)$. Such a construction is illustrated in Figure 5, and it concludes the proof of Theorem 3.3, and Theorem 3.1. □

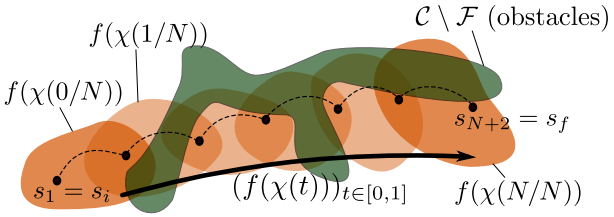

Figure 5: From a valid continuous path $(\chi(t))_{t\in[0,1]} \in \Omega^{[0,1]}$ to a valid finite sequence of configurations in $\mathcal{C}$.

In the two next sections, we present applications of Theorem 3.1 to more practical problems.

4 Application to vision-based reactive footstep planning

In this section we apply Theorem 3.1 to the problem of footstep planning for humanoid robots. In previous work ([Perrin et al., 2011], [Perrin et al., 2012]), we used a similar approach, but without establishing a link with a more general theorem such as Theorem 3.1, and with several significant differences. In particular, the approach presented in this section is designed to efficiently use input heightmaps acquired by stereo vision, while in [Perrin et al., 2011] and [Perrin et al., 2012] 3D models of the obstacles were assumed to be known in advance.

In sections 4.1 to 4.4, we show how to apply Theorem 3.1 so as to obtain an efficient algorithm for planning sequences of steps on heightmaps. Then, in section 5, we describe our decentralized architecture for real experiments of reactive footstep replanning with the DLR-Biped robot. Obstacle avoidance is safe if every obstacle is included in the heightmap, but not all environments can be represented by heightmaps, and in previous work, we addressed this issue by using a hybrid bounding box ([Perrin et al., 2012]) to perform full-body obstacle avoidance in complex environments.

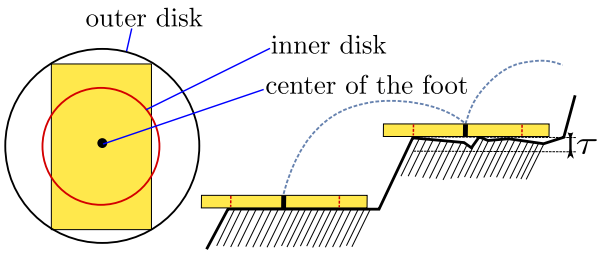

Figure 6: On the left the outer and inner disks are displayed, the latter one being a bit greater than the width of the foot. As shown on the right, the region between the inner and outer circles does not necessarily have to be in contact with the ground for a location to be safe.

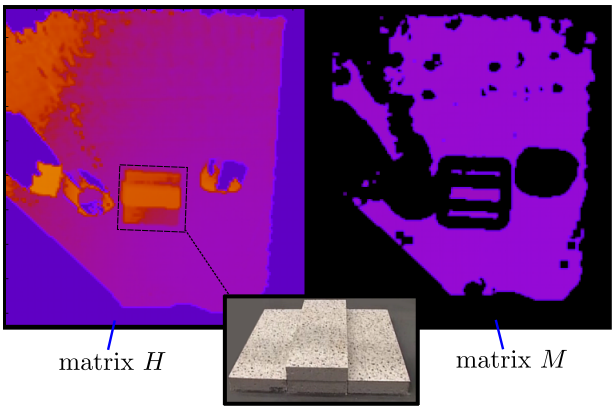

Figure 7: On the left is displayed the heightmap of an environment containing some small stairs. The points in the purple zone in the map on the right are the safe foot locations in this environment.

4.1 Construction of the map of safe foot locations

Let us assume that a discretized heightmap of the environment is known, that is to say a matrix $H = (h_{i,j})_{0 \leq i \leq k_1, 0 \leq j \leq k_2}$, with $h_{i,j} = h(x_i, y_j)$ (we use a fixed resolution along the x and y axes). Our first objective is to build a map of the locations where the feet of the robot can safely land. To keep a low dimensionality and simplify the application of Theorem 3.1, we assume that the robot has circular feet. Since the real robot (the DLR-Biped) has in fact rectangular feet, we consider two disks, the inner and outer disks, as shown in Figure 6. For every location (x_a, y_b) in the map, we first find the maximum height h_m among all the values of $h(x_i, y_j)$ such that (x_i, y_j) is inside the outer disk centered at (x_a, y_b). Then, we only consider the locations $(x_{i'}, y_{j'})$ inside the inner disk centered at (x_a, y_b), and we verify that all the heights $h(x_{i'}, y_{j'})$ are above $h_m - \tau$, where τ is a very small threshold determined experimentally. If this property is verified, we consider the location (x_a, y_b) as safe, and associate it with a height close to h_m which depends on the values of H inside the inner disk. This verification ensures us that most of the part of the foot inside the inner disk is in contact with an almost flat and horizontal region of the ground. Once this verification has been made for all the locations in H, we obtain a new matrix M that stores all the safe landing locations together with the height at which the foot should land. Figure 7 illustrates the construction of the matrix M for an example heightmap built from a stereo image pair.

4.2 Application of Theorem 3.1

In this section we explain how we apply Theorem 3.1 to footstep planning. For now, we do not take into account the height of the footsteps, and use the configuration space $\mathcal{C} = (SE(2))^2$ (one element of $SE(2)$ for each foot, specifying the position of its center and its orientation). We use $\Omega = SE(2)$, and denote elements of $SE(2)$ by triples (x, y, θ). We also define the "difference" between two angles θ and θ' as the value $dif_a(\theta, \theta') = \theta' - \theta + 2\pi n$ with $n \in \mathbb{Z}$ minimizing $|\theta' - \theta + 2\pi n|$. On $\mathcal{C}$, we use a distance of the form:

$$\begin{aligned} dist(((x_l, y_l, \theta_l), (x_r, y_r, \theta_r)), ((x'_l, y'_l, \theta'_l), (x'_r, y'_r, \theta'_r))) \\ = A.\, d_2((x_l, y_l), (x'_l, y'_l)) + B.\, |dif_a(\theta_l, \theta'_l)| \\ + A.\, d_2((x_r, y_r), (x_r, y'_r)) + B.\, |dif_a(\theta_r, \theta'_r)|, \end{aligned}$$

with $A > 0$ and $B > 0$.

f is a function that maps elements of Ω to sets of configurations in $\mathcal{C}$. The sets $f(x, y, \theta)$ are similar to the object Φ defined in [Perrin et al., 2011].

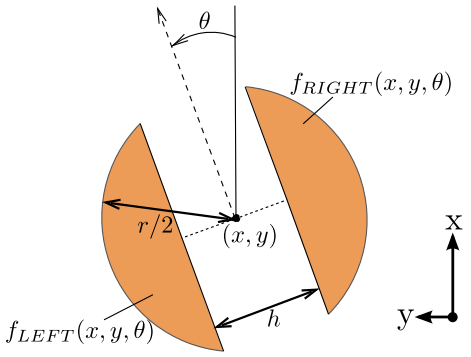

Figure 8: $f_{LEFT}(x,y,\theta)$ and $f_{RIGHT}(x,y,\theta)$.

They are based on two subsets of $\mathbb{R}^2$, $f_{LEFT}(x,y,\theta)$ and $f_{RIGHT}(x,y,\theta)$, which depend on two parameters, r and h. $f_{LEFT}(x,y,\theta)$ and $f_{RIGHT}(x,y,\theta)$ are symmetric portions of a disk, and are defined as shown in Figure 8. We also define $\widehat{f_{LEFT}}(x,y,\theta)$ and $\widehat{f_{RIGHT}}(x,y,\theta)$, subsets of $SE(2)$ that depend on two parameters $\alpha_{in} > 0$ and $\alpha_{out} > 0$, as follows:

$$\widehat{f_{LEFT}}(x,y,\theta) = \{(x,y,\beta) | (x,y) \in f_{LEFT}(x,y,\theta) \wedge -\frac{\alpha_{in}}{2} \leq dif_a(\theta,\beta) \leq \frac{\alpha_{out}}{2}\}$$

$$\widehat{f_{RIGHT}}(x,y,\theta) = \{(x,y,\beta) | (x,y) \in f_{RIGHT}(x,y,\theta) \wedge -\frac{\alpha_{out}}{2} \leq dif_a(\theta,\beta) \leq \frac{\alpha_{in}}{2}\}$$

Finally, we have:

$$f(x,y,\theta) = \widehat{f_{LEFT}}(x,y,\theta) \times \widehat{f_{RIGHT}}(x,y,\theta) \quad (1)$$

It can be shown that for any open free space $\mathcal{F} \subset \mathcal{C}$, f verifies the two first assumptions required for Theorem 3.1 to apply. Indeed, the function f basically only rotates and translates the fixed set $\widehat{f_{LEFT}}(0,0,0) \times \widehat{f_{RIGHT}}(0,0,0)$; hence, the sets $f(x,y,\theta)$ are modified in a continuous manner, and consequently $\delta_{obs} \circ f$ is continuous. Besides, the simple geometry of the open sets $f(x,y,\theta)$ explains why for s and s' very close in $SE(2)$, $f(s) \cup f(s')$ and $f(s) \cap f(s')$ are very close in the sense of the Hausdorff distance, which implies the second assumption required by Theorem 3.1.

In the problem of footstep planning, the relation R should define the stepping capabilities of the robot, i.e. $R(s,s')$ should be verified if and only if we can go from the configuration s to the configuration s' in one step. Yet, it is more convenient to deal with sequences of two steps (because both footprints can be modified), so we will use a relation R such that $R(s,s')$ is verified if and only if it is possible to go from s to s' with two steps (we fix rules so that these two steps are unambiguously defined by s and s'). The relation R will of course not capture the real stepping capabilities of the robot, but it must be a conservative approximation. This means that the steps allowed by R should always be feasible on the robot, and R should not be too restrictive so as not to under-exploit the robot capabilities.

For $s, s' \in \mathcal{C}$, we actually define $R(s,s')$ as follows:

$$R(s,s') \equiv \exists \varphi \in SE(2),\ s \in f(\varphi) \wedge s' \in f(\varphi) \quad (2)$$

We do not prove it here, but this definition implies the validity of the fourth and fifth assumptions required by Theorem 3.1 (for the fourth assumption, it is a direct consequence of the definition of R; for the fifth assumption, it is a consequence of some geometric properties of f).

So, in order to approximate the stepping capabilites of the robot in a conservative way, we can tune the parameters r, h, α_{in} and α_{out}. The configurations that can be used by the robot with this definition of R are described in Figure 9 (with the left foot used as a reference). Attention must be paid to the fact that the restrictions imposed by R are not only on the configurations between steps: they are also on the *transitions* between configurations (see [Perrin et al., 2011] for a visual description of these restrictions). On Figure 9, we can see that the maximum distance between the centers of the feet is r, while h limits the closeness of the feet. The orientation of the right foot relatively to the left one is comprised between $-\alpha_{out}$ and α_{in}.

To adjust the values of r, h, α_{in} and α_{out}, we use an empirical approach. The first constraint is that these values should lead to steps without self-collisions. Due to the particular structure of the legs of the DLR-Biped, a self-collision can easily occur

when its feet are turned inward, even just slightly: see Figure 10. For this reason, we use asymmetric constraints on the foot orientations, and set α_{in} to 1° and α_{out} to 15°.

Furthermore, we sample many configurations of the robot with both feet on the ground (and at the same height), obtaining the results shown in Figure 11. The left foot being used as a reference, each point represents a position of the center of the right foot, and for each location, we test several orientations ranging from $-\alpha_{out}$ to α_{in}. The configuration of the robot legs is set by inverse kinematics. In Figure 11, the points inside the region defined by the thick line segments are the ones for which there exists at least one orientation so that self-collisions almost occur, i.e. some body of the left leg is at most at distance 2cm from a body of the right leg. The other points are "safe" configuration, and we must choose h so that only safe configurations can occur. That's why we set $h = 20$cm. Besides, the steps allowed by R should be executed without failure by the control algorithm. This limits the maximum length of steps, hence it limits r. With the values of α_{in}, α_{out} and h already set, we test various gaits and various step lengths to tune r. The value chosen according to these tests is $r = 32$cm. Using such parameters leads to the validity of the third assumption required by Theorem 3.1: for any $(x, y, \theta) \in \Omega = SE(2)$, and any $s, s' \in f(x, y, \theta)$, it is actually possible to go from s to s' with two steps, without falling and without self-collisions.

Hence, all the assumptions of Theorem 3.1 are verified, and we can apply it. This means that in order to look for sequences of steps, we can reason in the smooth configuration space $\Omega = SE(2)$. The new notion of collision on this space is κ_Ω, verified for $(x, y, \theta) \in \Omega$ if and only if there exists a configuration $s \in \mathcal{C}$ inside $f(x, y, \theta)$ such that the centers of both left and right foot are at safe locations (which can be checked with the matrix M). Thanks to Theorem 3.1, we know that any valid continuous path in Ω can be converted into a finite sequence of valid steps. Figure 12 illustrates an example of such conversion, from a valid continuous path in $SE(2)$ to a finite sequence of valid steps (for the sake of clarity, on Figure 12 we use and display inner disks smaller

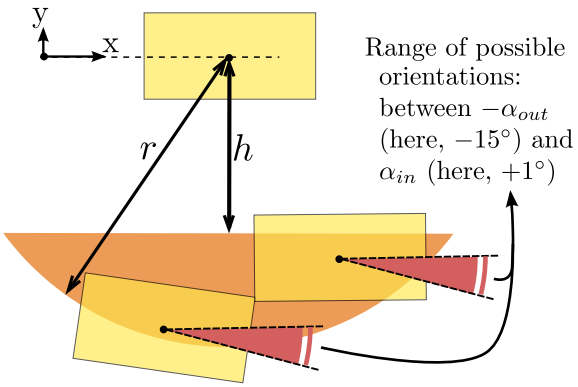

Figure 9: The configurations allowed by the definition of R according to equation (2).

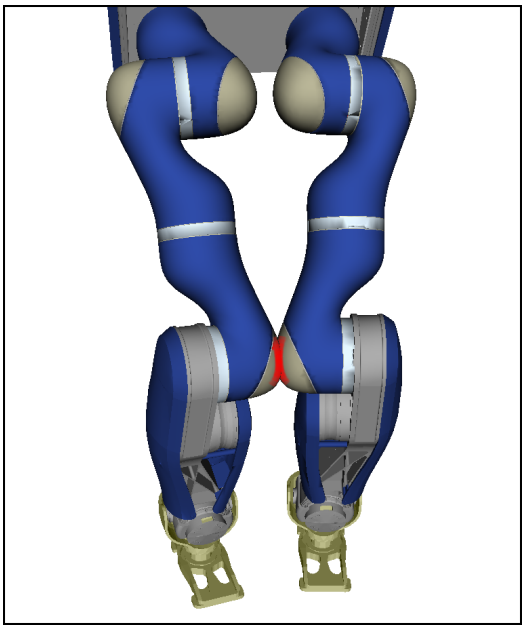

Figure 10: Due to the structure of the DLR-Biped legs, self-collisions are likely to occur when its feet are turned inward.

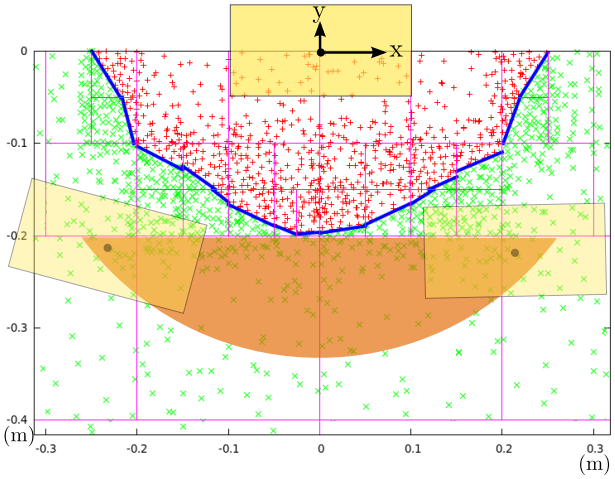

Figure 11: This shows how the values h and r (see Figure 9) have been computed for the DLR Biped: the left foot is kept at the zero position and orientation, and each sample point corresponds to a position of the center of the right foot. For each of these sample points, stances obtained with different orientations of the right foot ranging from $-15°$ to $1°$ have been tested. The points inside the region defined by the thick line segments are the ones for which some orientation of the right foot can bring the robot very close to self-collisions. h must be chosen so as to exclude this region, while r can be set as large as the walking control algorithm allows it.

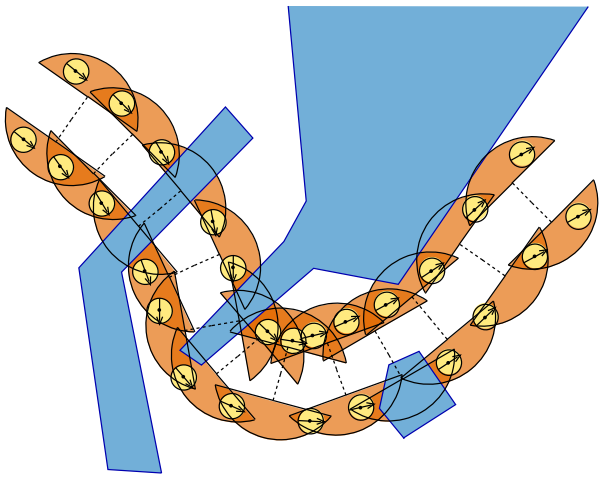

Figure 12: Conversion: from a valid continuous path in $SE(2)$ to a valid finite sequence of steps.

than the ones corresponding to the dimensions of the DLR-Biped).

There is however a problem remaining: so far we haven't introduced any verification that would prevent the robot from walking on stairs with large step height. Thus, we add a new test in κ_Ω: for a configuration $(x, y, \theta) \in \Omega$ to be valid (collision-free), there must not only be a valid configuration $s \in f(x, y, \theta)$, there must also be a valid configuration in $f(x, y, \theta)$ such that the minimum height z of both feet locations (these heights are stored in the matrix M) verifies the following property: no point inside the disk of center (x, y) and radius $r + \rho$ (where ρ is a small positive constant) is above $z+m_{STAIR}$, where $m_{STAIR} > 0$ is an empirically defined constant that depends on the robot capabilities and roughly denotes the maximum height of steps on which the robot can walk. This constraint also prevents the robot from walking close to high obstacles while still allowing it to step over small obstacles (of height less than m_{STAIR}).

4.3 Construction of the matrix M_{κ_Ω}

The tests κ_Ω are more complex than usual collision checks. In this section we describe the construction of a matrix M_{κ_Ω} that stores the results of the tests for every location of the discretized heightmap.

Let us consider a location (x_a, y_a) in the discretized map. For the tests κ_Ω, the orientation matters, so the matrix M_{κ_Ω} will be 3-dimensional. We use a $1°$ precision, and thus to every location (x_a, y_b) correspond 360 tests whose results must be stored in M_{κ_Ω}: $(x_a, y_b, 0°), (x_a, y_b, 1°), \ldots, (x_a, y_b, 359°)$. For every location (x_a, y_b) in the discretized map, we perform a computation to find all the orientations β such that $\kappa_\Omega(x_a, y_b, \beta)$ is verified. First of all, we compute the maximum height z_{max} inside the disk of center (x, y) and radius $r + \rho$ (see previous section). We only consider safe locations that are above height

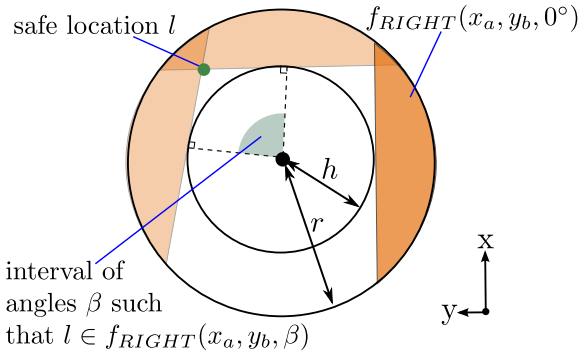

Figure 13: Finding orientations β such that $f_{RIGHT}(x_a, y_b, \beta)$ contains a given safe location.

$z_{max} - m_{STAIR}$. As shown on Figure 13, for every such safe location l we can easily find the interval of angles β such that l is inside $f_{RIGHT}(x_a, y_b, \beta)$ (resp. $f_{LEFT}(x_a, y_b, \beta)$). By scanning the locations inside the disk of center (x, y) and radius r (and outside the disk of center (x, y) and radius h) in a smart order (and without considering the locations at a distance from (x_a, y_a) less than h), we can manage to compute the union of these intervals in an efficient way. Afterwards, it is easy to fill the 360 values in M_{κ_Ω} associated with the location (x_a, y_a). After repeating this process for all the locations in H, we obtain the full matrix M_{κ_Ω}, which can be used to compute the tests κ_Ω in constant time.

4.4 Footstep planning simulation

Algorithm 1 sums up the 4 steps of the main loop of our footstep planning algorithm. After the discretized heightmap of the environment has been received, we first construct the matrix M, and then the matrix M_{κ_Ω}.

The fact that these matrices are constructed at every iteration of the loop might seem unefficient, but it has actually several advantages. The main one is that it is very well suited for parallel computations (but we do *not* use parallel computations in the implementation described in the present paper). Thanks to the matrix M_{κ_Ω}, the complex collision checks κ_Ω

Algorithm 1

1: **while** (true) **do**
2: Receive the updated heightmap matrix H.
3: Construct the matrix M that stores safe foot locations.
4: Construct the matrix M_{κ_Ω}.
5: If needed, perform motion planning in $SE(2)$, using M_{κ_Ω} for collision checks. Then, use a greedy approach (similar to the one of the algorithm for flea motion planning: see Section 2) to convert the continuous path into a finite sequence of steps.
6: **end while**

can be evaluated extremely quickly, and thus, the path planning itself is very fast (it is done in $SE(2)$, a space with only 3 dimensions). As a result, the most time-consuming part of the computation is the construction of the two matrices M and M_{κ_Ω}. Unlike classical motion planning algorithms which are difficult to parallelize (see for example [Pan et al., 2010]), it is straight forward to make the construction of M and M_{κ_Ω} parallel. Besides, the time required for the construction of M and M_{κ_Ω} does not vary a lot, while the randomness in most motion planning algorithms makes their duration hard to predict. It follows that our approach is well suited for systems with hard real-time constraints.

Figure 14 shows a simulation of footstep planning. We use a kinematic model of the DLR-Biped, but there is no physics simulation. The robot starts in an initial configuration and must find a sequence of steps toward a goal. The heightmap used is the same as the one displayed in Figure 7. We resend it at every iteration of the loop of Algorithm 1, and the matrices M and M_{κ_Ω} are rebuilt from scratch every time. The heightmap is a 200 by 200 matrix, which means that for M_{κ_Ω}, $200 \times 200 \times 360 = 14,400,000$ values have to be computed and filled at each iteration. For the motion planning, we use the RRT-Connect algorithm [Kuffner and Lavalle, 2000] from the OMPL library [OMPL, 2010] to find paths in $SE(2)$, and then use a greedy approach (as explained in Section 2) to convert them into finite sequences of steps. Thanks to the matrix M_{κ_Ω} (collision checks in con-

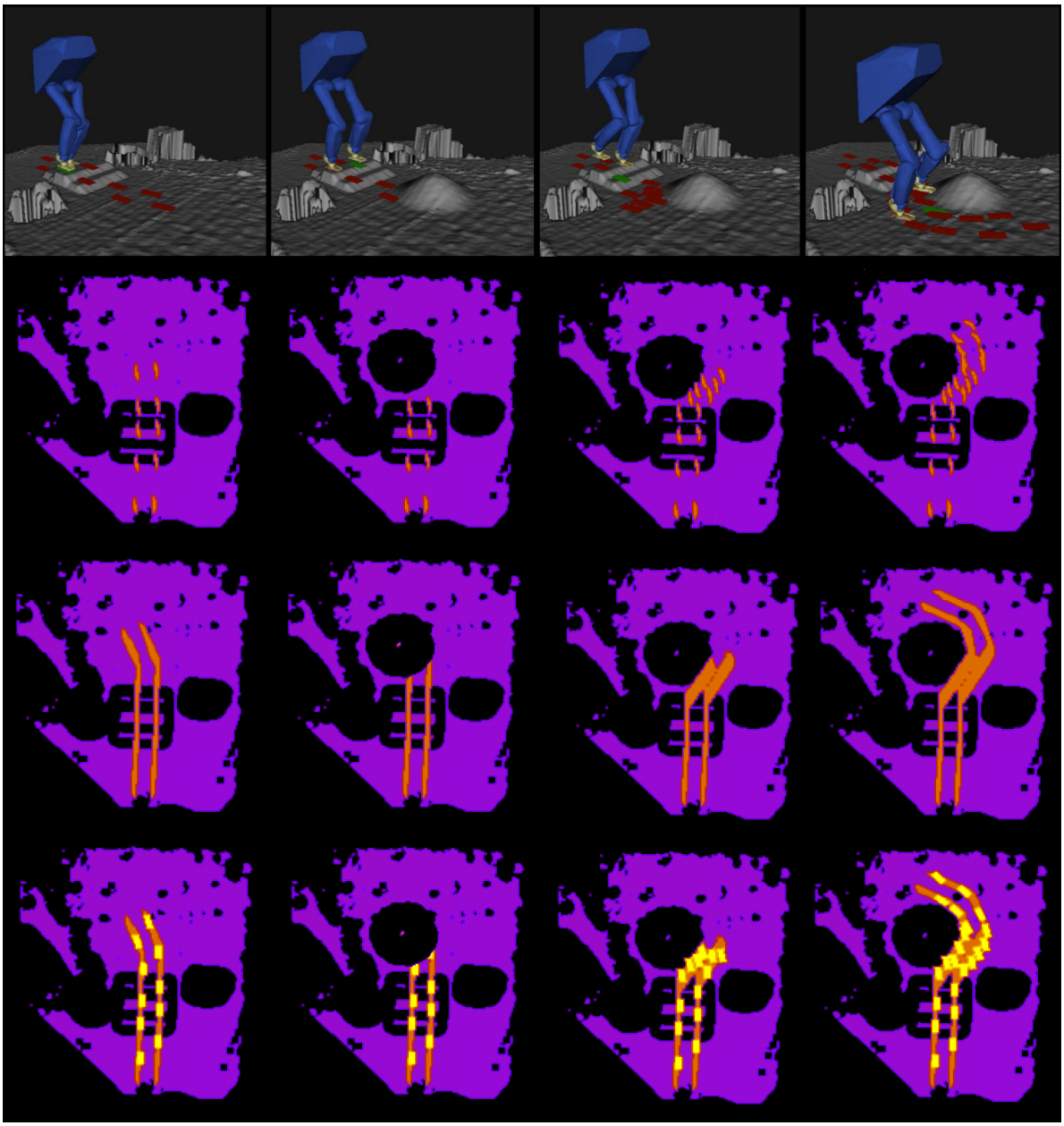

Figure 14: A simulation of footstep planning. Below the snapshots are displayed maps of safe locations (the matrix M) with additional data. In the first row below the snapshots are shown the sets $f_{LEFT}(\varphi) \cup f_{RIGHT}(\varphi)$ for milestones $\varphi \in SE(2)$ chosen by the RRT-Connect algorithm, for a part of the last path found. In the second row are shown "traces" of these sets along the paths. In the row at the bottom, we also display the footprints resulting from the greedy conversion to finite sequences of steps.

stant time), the motion planning is very fast, so we actually repeat this process (RRT-Connect + greedy conversion) 10 times, and keep the shortest sequence of steps. To obtain short sequences of steps, some more efficient strategies are made possible by our novel continuous approach, such as trajectory optimization on the path found in $SE(2)$ (see for example [Park et al., 2012]), or kinodynamic planning (see for example [Sucan and Kavraki, 2008]). The use of such strategies seems promising but is out of the scope of this paper.

Simultaneously to motion planning, on the same computer, a walking pattern generator is simulated on another thread. It takes as input the sequence of future steps, which is updated at each iteration. The communication between the planning and control threads simulates the communication protocol used for the experiments on the real DLR-Biped, explained in the next section.

In the simulation shown on Figure 14, the robot first has to find its way through small stairs, and after a few seconds, a bump is artificially introduced in the heightmap, forcing the robot to quickly find another path toward the goal. For 10 trials of this simulation on an Intel(R) Core(TM) i7 1.60GHz CPU, we measure at each iteration (of the loop of Algorithm 1) the time required for the construction of the matrices M and M_{κ_Ω}. Figure 15 displays the mean time together with the standard deviation for those 10 trials, and only for the first 11 iterations of the loop (after that, the robot had reached or almost reached the goal in every trial). The first iteration takes a bit longer (2.4s in average) because of the initial memory allocation. The variations of the computation times are mostly due to the thread that does the walking pattern generation and the 3D display. For the same 10 trials, we also measure the time required at every iteration for motion planning, i.e. the 10 executions of the RRT-Connect algorithm, each followed by a greedy conversion of the path found in $SE(2)$ into a finite sequence of steps. Path planning is performed at every iteration, even if the environment has not changed since the previous iteration. Since the robot is getting gradually closer to the goal, motion planning becomes easier, which explains the decreasing tendency of the computation times that can be seen in Figure 16. At

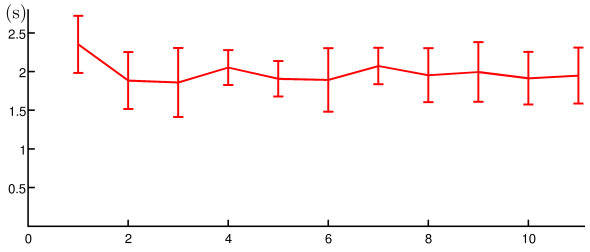

Figure 15: Average time and standard deviation for the construction of the matrices M and M_{κ_Ω}, at each iteration of the loop of Algorithm 1 (10 trials).

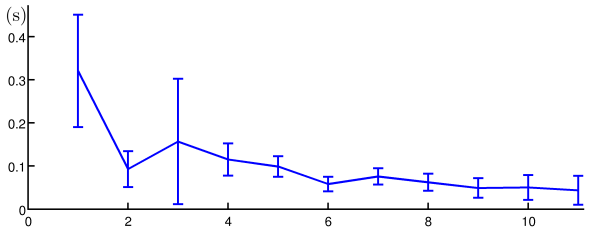

Figure 16: Average time and standard deviation for motion planning in $SE(2)$ and the conversion to finite sequences of steps, at each iteration of the loop of Algorithm 1 (10 trials).

the first iteration, the robot has to find a sequence of steps that goes across the small stairs, which explains why motion planning takes substantially more time. A second peak can be observed at the third iteration: it is usually the iteration before which the bump appears in the heightmap. The average time for motion planning is always less than 0.33s. This means that one execution of RRT-Connect followed by the greedy conversion takes in average no more than 33ms.

5 A decentralized software architecture for reactive footstep planning

In this section we describe the decentralized software architecture that we used for real experiments on the DLR-Biped.

Unlike control or vision, motion planning is a task that can be done at a relatively low frequency, and it does not need to interact quickly with the robot sensors. For this reason, we perform motion planning on a computer which is not embedded in the robot. The advantage of this approach is that it reduces the onboard computational load and energy consumption. The drawback is that it would not be suitable if the robot were to explore an unknown environment with significant communication delay (or no communication at all).

We use a total of 3 computers: 2 on the robot, and one remote computer for the footstep planning. On the robot, there is one computer for the control, and one for the vision.

The role of the vision computer is to build the heightmap matrix H. In the simulation, the size of this matrix was 200 by 200, but in the real experiment we use a 100 by 100 matrix in order to make the vision algorithms faster. The resolution is 4cm, which means that the size of the area covered by the heightmap is 4m by 4m. Stereo image pairs are frequently obtained by the robot cameras and used to locally update the values of the heightmap matrix. In the areas where no data has been acquired yet, we assume a flat floor. The heightmap is updated about every second, but this update does not necessarily include data from a stereo image pair obtained one second before: the delay can vary roughly between 1 and 3 seconds. Whenever the heightmap file is being modified, we use a mutex to block any attempt at reading the file. Details on the stereo vision algorithm can be found in [A.Stelzer et al., 2012].

The control algorithm is based on capture point dynamics [Englsberger et al., 2011], and its purpose is to execute a list of steps received from the planner. Every step has an identifier. The control also needs to send some information to the planner: it

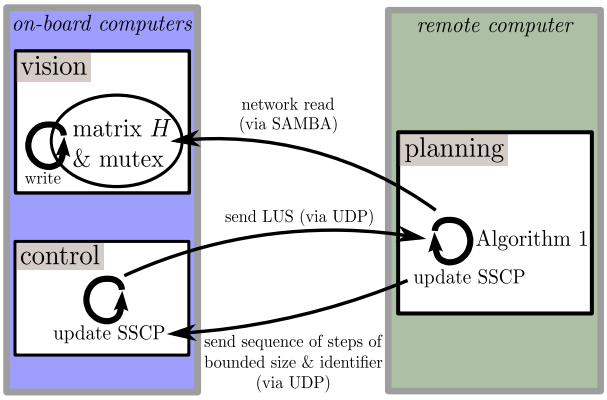

Figure 17: Communications between the 3 computers for vision, control and planning.

frequently sends the identifier of what we call the "last unmodifiable step" (LUS). This step depends on many parameters of the control algorithm, for example the walking speed. If the robot walks slowly, even the current step might be modifiable. But if the robot walks fast, it might have to execute at least the next two steps as they are currently planned. Since the LUS is difficult to predict precisely, we use a simple definition verified empirically. Knowing the LUS, the planner can decide from where (in the sequence of steps currently planned) to replan when necessary.

When a sequence of steps is found by the planner, only a constant number of steps is sent to the control. For example, only the first 7 steps of the sequence are sent (unless the sequence contains less than 7 steps). Every time a new list of steps is sent, both the planner and the controller update the "sequence of steps currently planned" (SSCP). During an iteration of the loop of Algorithm 1, replanning is done if the SSCP does not reach the current goal, or if it is not valid anymore in the current environment described by the matrix H. Since only a few steps are sent at each iteration, the SSCP grows progressively toward the goal. The advantage of this is that as the initial state for the replanning gets closer to the goal, motion planning becomes easier and better solutions can be expected in the same amount of time.

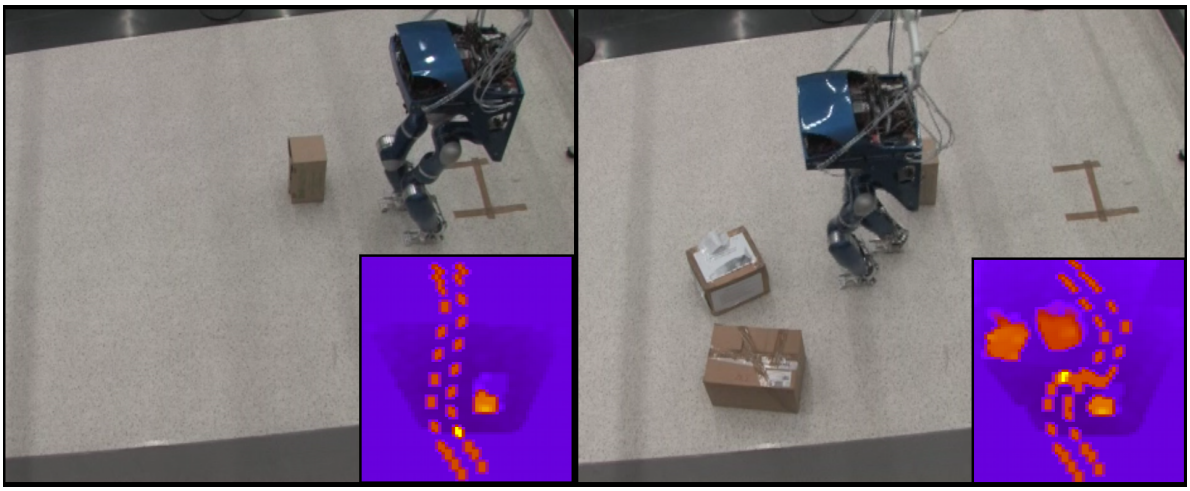

Figure 18: An experiment of reactive footstep planning with the DLR-Biped

Before starting the replanning, the planner decides which step of the SSCP will define the initial state. The sequence of steps found must be concatenated after this step in order to update the SSCP. Therefore, whenever a new list of steps is sent to the controller, the planner also indicates the identifier of the step after which the list must be concatenated. Of course, this identifier must never be before the LUS. As we have seen in the previous section, the computation time of our footstep planning algorithm does not vary a lot, and therefore, knowing approximately the duration of one step, we can use an appropriate margin to avoid the case where the SSCP would have to be modified before the LUS. For example, if the computation time for the planning is always much shorter than the duration of one step, it is safe to always replan only after the step following the LUS.

Figure 17 summarizes the communications between the 3 computers.

Figure 18 shows the result of an experiment on the DLR-Biped. On a flat surface, the robot first plans a sequence of steps toward a goal location. On its way to the goal, some obstacles are suddenly put in front of the robot which then updates the heightmap with the stereo vision algorithm and quickly replans a new sequence of steps.

6 Application to legged locomotion planning for a hexapod robot

In this section, we show that we can also apply Theorem 3.1 in order to use a continuous approach for planning the walking motion of a hexapod robot (cf. Figure 19). Our objective is to make the hexapod walk on uneven terrain with non-gaited locomotion planning (which is typically computationally costly). Again, the uneven terrain is described by a heightmap matrix which is used to set the height of the contact positions. We ignore the contact orientations, so we use $(\mathbb{R}^2)^6$ as the configuration space (it is easy to define a heuristic that sets a unique whole-body configuration from the 6 contact positions; in particular, we require the robot main body to remain horizontal). Here is how we define the valid states: from the heightmap we infer what locations are allowed for individual contacts similarly as in the previous section (except that we only consider one disk per leg, not an interior and an exterior disk), and for a configuration in $(\mathbb{R}^2)^6$, we require our heuristic to lead to a valid whole-body configuration that does not collide with the heightmap.

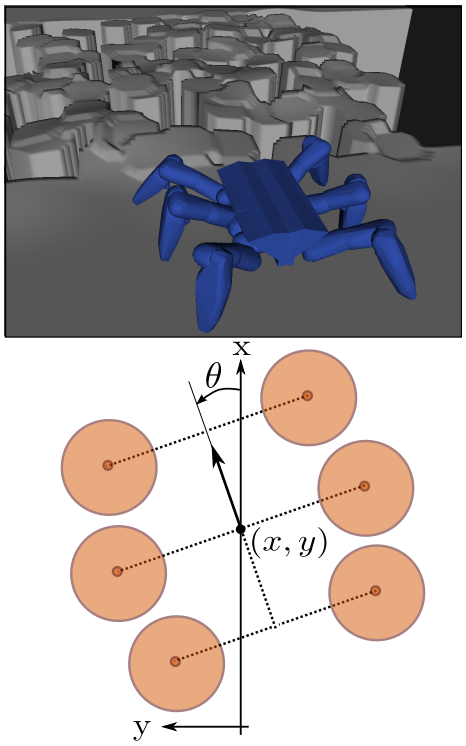

Figure 19: *At the top*: the hexapod in front of an uneven terrain it just crossed (cf. Figure 20). The real version of this hexapod does not exist, but its legs have the same structure as the legs of the DLR-Biped. *At the bottom*: the six open disks that restrict the configurations and steps of the hexapod (each leg must have its contact with the ground within its assigned disk).

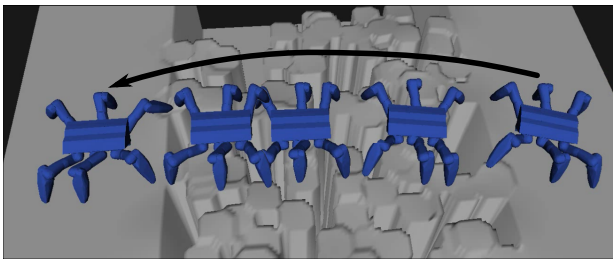

Figure 20: The motion accross this challenging terrain was planned in 57ms.

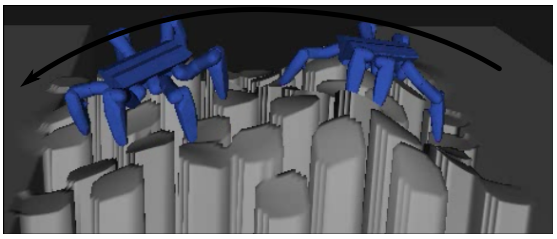

Figure 21: The algorithm presented in this section can be slightly improved so as to allow locomotion on more complex terrains. The motion accross this terrain was planned in 0.33s.

We try to apply Theorem 3.1 with the configuration space $\Omega = SE(2)$. For a configuration (x, y, θ), we define $f(x, y, \theta)$ as the set of configurations in $(\mathbb{R}^2)^6$ such that each contact belongs to an open disk, as shown on Figure 19 (it is important that the disks are disjoint). With this assumption, we first replace κ by heuristic checks: for a configuration $s \in f(x, y, \theta)$, $\kappa(s)$ is verified if the contacts are safe (i.e. the heightmap is almost flat around their locations), if the maximum height difference between two contacts with the ground is less than some threshold, and if the maximum height of the heightmap in the "robot zone" (i.e. the convex hull of the contacts) is not much higher than the height of the contacts. We consider transitions where all the 6 legs are moved at the same time, and we say that a transition from $s \in (\mathbb{R}^2)^6$ to $s' \in (\mathbb{R}^2)^6$ is allowed if and only if there exists $(x, y, \theta) \in SE(2)$ such that $s \in f(x, y, \theta)$ and $s' \in f(x, y, \theta)$. With this restriction, we can verify that the conditions of Theorem 3.1 apply (only the fifth property is difficult to verify), and thus we can use the equivalence to convert our problem of locomotion planning into a continuous motion planning in $SE(2)$ (we use again the library OMPL and the algorithm RRT-Connect to perform motion planning). Once the conversion of a continuous path is done, we obtain a finite sequence of transitions for which the 6 foot locations are changed at each transition. It is not difficult to convert it into a sequence of feasible transitions where at most 3 feet are moved at the same time (but sometimes 2, or just one).

This original technique for legged locomotion planning is convenient and fast: in the example described in Figure 20 where the hexapod must go across an uneven and challenging terrain, the whole planning (continuous planning *and* two-stage conversion into a discrete sequence of steps) was done in 57ms on an Intel(R) Core(TM) i7 1.60GHz CPU. The algorithm can even be improved so as to change the hexapod's orientation according to the current height of its legs, which enables it to cross even more challenging terrains. The motion accross the terrain shown on Figure 21 was planned in 0.33s. However, we cannot directly use the same method to solve planning problems as complex as the ones considered in [Hauser et al., 2006], but it is an interesting compromise between gaited methods and more complex approaches such as [Hauser et al., 2006].

7 Conclusion

The main contribution of this paper is to define a set of conditions that make discrete motion planning for legged robot equivalent to continuous motion planning in a dedicated configuration space using the notion of weak-collision freeness. This equivalence provides a new way to reason about planning discrete actions in continuous spaces.

We have successfully applied the technique to various systems and it has revealed rather efficient in terms of computation. Furthermore, we believe that it can have applications beyond footstep or multi-contact planning, such as for example in dextrous manipulation or hybrid systems.

References

[Alami et al., 1994] Alami, R., Laumond, J.-P., and Siméon, T. (1994). Two manipulation planning algorithms. *1st Workshop on the Algorithmic Foundations of Robotics (WAFR'94)*.

[A.Stelzer et al., 2012] A.Stelzer, Hirschmüller, H., and Görner, M. (2012). Stereo-vision-based navigation of a six-legged walking robot in unknown rough terrain. *Int. Journal of Robotics Research*, 31(4):381–402.

[Boissonnat et al., 2000] Boissonnat, J.-D., Devillers, O., and Lazard, S. (2000). Motion planning of legged robots. *SIAM Journal on Computing*, 30(1):218–246.

[Bourgeot et al., 2002] Bourgeot, J.-M., Cislo, N., and Espiau, B. (2002). Path-planning and tracking in a 3d complex environment for an anthropomorphic biped robot. In *IEEE Int. Conf. on Intelligent Robots and Systems (IROS'02)*, volume 3, pages 2509–2514.

[Bouyarmane and Kheddar, 2011] Bouyarmane, K. and Kheddar, A. (2011). Multi-contact stances planning for multiple agents. In *IEEE Int. Conf. on Robotics and Automation (ICRA'11)*, pages 5246 – 5253.

[Bullo and Lewis, 2004] Bullo, F. and Lewis, A. D. (2004). *Geometric Control of Mechanical Systems*, volume 49 of *Texts in Applied Mathematics*. Springer Verlag.

[Chestnutt et al., 2003] Chestnutt, J., Kuffner, J. J., Nishiwaki, K., and Kagami, S. (2003). Planning biped navigation strategies in complex environments. In *IEEE/RAS Int. Conf. on Humanoid Robots (Humanoids'03)*.

[Chestnutt et al., 2005] Chestnutt, J., Lau, M., Cheung, G., Kuffner, J. J., Hodgins, J., and Kanade, T. (2005). Footstep planning for the honda asimo humanoid. In *IEEE Int. Conf. on Robotics and Automation (ICRA'05)*, pages 631–636.

[Chestnutt et al., 2007] Chestnutt, J., Nishiwaki, K., Kuffner, J. J., and Kagami, S. (2007). An adaptive action model for legged navigation planning. In *IEEE/RAS Int. Conf. on Humanoid Robots (Humanoids'07)*, pages 196–202.

[Dalibard et al., 2011] Dalibard, S., El Khoury, A., Lamiraux, F., Taix, M., and Laumond, J.-P. (2011). Small-space controllability of a walking humanoid robot. In *IEEE/RAS Int. Conf. on Humanoid Robots (Humanoids'11)*.

[de Berg et al., 2000] de Berg, M., van Kreveld, M., Overmars, M. H., and Schwarzkopf, O. (2000). *Computational Geometry: Algorithms and Applications*. Springer-Verlag.

[Eldershaw and Yim, 2001] Eldershaw, C. and Yim, M. (2001). Motion planning of legged vehicles in an unstructured environment. In *IEEE Int. Conf. on Robotics and Automation (ICRA'01)*, pages 3383–3389.

[Englsberger et al., 2011] Englsberger, J., Ott, C., Roa, M. A., Albu-Schäffer, A., and Hirzinger, G. (2011). Bipedal walking control based on capture point dynamics. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'11)*, pages 4420–4427.

[Goodwine, 1997] Goodwine, B. (1997). *Control of Stratified Systems with Robotic Applications*. PhD thesis, California Institute of Technology.

[Gutmann et al., 2005] Gutmann, J.-S., Fukuchi, M., and Fujita, M. (2005). Real-time path planning for humanoid robot navigation. In *Int. Joint Conf. on Artificial Intelligence (IJCAI05)*, pages 1232–1237.

[Harmati and Kiss, 2001] Harmati, I. and Kiss, B. (2001). Motion planning algorithms for stratified kinematic systems with application to hexapod robot. *Journal of Acta Cybernetica*, 15(2):225–240.

[Hauser et al., 2006] Hauser, K., Bretl, T., Latombe, J.-C., and Wilcox, B. (2006). Motion planning for a six-legged lunar robot. In *7th Workshop on the Algorithmic Foundations of Robotics (WAFR'06)*, pages 16–18.

[Kanoun et al., 2011] Kanoun, O., Laumond, J.-P., and Yoshida, E. (2011). Planning foot placements for a humanoid robot: A problem of inverse kinematics. *Int. Journal of Robotics Research*, 30(4):476–485.

[Kavraki et al., 1996] Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. on Robotics and Automation*, 12:566–580.

[Kuffner and Lavalle, 2000] Kuffner, J. J. and Lavalle, S. (2000). RRT-Connect: An efficient approach to single-query path planning. In *IEEE Int. Conf. on Robotics and Automation (ICRA'00)*, pages 995–1001.

[Kuffner et al., 2001] Kuffner, J. J., Nishiwaki, K., Kagami, S., Inaba, M., and Inoue, H. (2001). Footstep planning among obstacles for biped robots. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'01)*, pages 500–505.

[Lafferriere and Sussmann, 1991] Lafferriere, G. and Sussmann, H. J. (1991). Motion planning for controllable systems without drift. In *IEEE Int. Conf. on Robotics and Automation (ICRA'91)*, pages 1148–1153.

[LaValle and Kuffner, 2000] LaValle, S. M. and Kuffner, J. J. (2000). Rapidly-exploring random trees: Progress and prospects. In *4th Workshop on the Algorithmic Foundations of Robotics (WAFR'00)*, pages 293–308.

[OMPL, 2010] OMPL (2010). The Open Motion Planning Library. `http://ompl.kavrakilab.org`.

[Pan et al., 2010] Pan, J., Lauterbach, C., and Manocha, D. (2010). g-Planner: Real-time motion planning and global navigation using GPUs. In *24th AAAI Conf. on Artificial Intelligence*.

[Park et al., 2012] Park, C., Pan, J., and Manocha, D. (2012). ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments. In *Int. Conf. on Automated Planning and Scheduling*.

[Perrin, 2012] Perrin, N. (2012). From discrete to continuous motion planning. *10th Workshop on the Algorithmic Foundations of Robotics (WAFR'12)*.

[Perrin et al., 2012] Perrin, N., Stasse, O., Lamiraux, F., Kim, Y. J., and Manocha, D. (2012). Real-time footstep planning for humanoid robots

among 3d obstacles using a hybrid bounding box. In *IEEE Int. Conf. on Robotics and Automation (ICRA'12)*.

[Perrin et al., 2011] Perrin, N., Stasse, O., Lamiraux, F., and Yoshida, E. (2011). Weakly collision-free paths for continuous humanoid footstep planning. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'11)*, pages 4408–4413.

[Sastry, 1999] Sastry, S. (1999). *Nonlinear Systems: Analysis, Stability, and Control*, volume 10 of *Interdisciplinary Applied Mathematics*. Springer.

[Serra, 1983] Serra, J. (1983). *Image Analysis and Mathematical Morphology*. Academic Press, Inc.

[Sucan and Kavraki, 2008] Sucan, I. A. and Kavraki, L. E. (2008). Kinodynamic motion planning by interior-exterior cell exploration. In *8th Workshop on the Algorithmic Foundations of Robotics (WAFR'08)*.

[Yoshida et al., 2008] Yoshida, E., Esteves, C., Belousov, I., Laumond, J.-P., Sakaguchi, T., and Yokoi, K. (2008). Planning 3D collision-free dynamic robotic motion through iterative reshaping. *IEEE Trans. on Robotics*, 24(5):1186–1198.