

Fast humanoid robot collision-free footstep planning using swept volume approximations

Nicolas Perrin, Olivier Stasse, Léo Baudouin, Florent Lamiroux and Eiichi Yoshida

Abstract—In this paper, we propose a novel and coherent framework for fast footstep planning for legged robots on a flat ground with 3D obstacle avoidance. We use swept volume approximations computed offline in order to considerably reduce the time spent in collision checking during the online planning phase, in which an RRT variant is used to find collision-free sequences of half-steps (produced by a specific walking pattern generator). Then, an original homotopy is used to smooth the sequences into natural motions avoiding gently the obstacles. The results are experimentally validated on the robot HRP-2.

Index Terms—footstep generation, motion planning, humanoid robots, obstacle avoidance.

I. INTRODUCTION

ARGUABLY, the one thing that most differentiate humanoid robots from their wheeled counterparts is their intrinsic ability to step over obstacles on the ground. For this reason a lot of work has been done on the problem of humanoid robot walk planning, with the aim of exploiting at best this unique capability. Since humanoid robots combine high dimensionality with underactuation, two properties that tend to drastically increase the complexity of motion planning, this problem is not easy to solve. Nevertheless, and although there is no completely satisfying solution so far, a lot of promising techniques and tools have been introduced over the past decade.

Probably the most successful approaches are based on the use of the A* algorithm with a finite transition model, i.e. a relatively small set of possible steps (see for example [23], [4], [6], [7]). For each step a corresponding configuration space trajectory is known, and it is possible to check quite quickly whether a given step will avoid the obstacles or not. Since those steps need to be connectable at will, however, it often requires the initial and final speed of the robot bodies to be zero for all the steps of the transition model. At least some parts of the gaits produced are thus static. This is for example the case in [23], and [1]. Chestnutt *et al.* avoid it in [7] by using a search space which consists in sequences of two consecutive steps. But since this search space has a higher dimensionality, in order to be expressive, transition models need to be much larger than when only isolated steps are considered. Yet, the use of the A* algorithm strongly constrains the size of the transition model. Even when the transitions are isolated

steps, the stepping capabilities are often limited because the complexity of the A* search quickly increases with the size of the transition model. Recently though, some interesting refinements have been considered in order to enhance the stepping capabilities while keeping a small transition model. In [9] for example, the steps of a set of *reference actions* (i.e. the transition model) can be slightly adjusted to avoid bad terrain locations.

In this paper, we replace the A* search by a sampling-based algorithm in order to directly deal with a large transition model, and add several other improvements to the standard $\{A^* + \text{finite transition model}\}$ approach. Here are our main contributions:

- Thanks to a walking pattern generator specifically designed, we obtain a low-dimensional search space which can be densely covered by relatively few points. With an automatically generated finite transition model of about 300 points in this search space, we are able to obtain very expressive stepping capabilities. To deal with such a large transition model, we use, instead of the classical A* search, a specific Rapidly-exploring Random Tree (RRT) algorithm.
- Each point in the transition model corresponds to a configuration space trajectory of the robot. Through extensive offline computations, for each of them we approximate the volume swept in the workspace by a part of the robot lower body (from the knees down) during the execution of the trajectory, and store it in an efficient data structure. It helps to drastically reduce the time consumed by the online planning phase when checking for collisions with the environment.
- Finally, with a simple homotopy, we quickly smooth and accelerate the trajectories obtained after the planning phase, and as a result the final motions produced are fully dynamic, a feature that often lacks with current approaches. On top of that, there is no incoherence between the planning phase and the smoothing phase, so we have the guarantee that if the planner returns a collision free solution, then the robot will execute a sequence which will also be collision free (this guarantee is up to some details –discrepancies between simulation and real world, errors of approximation, errors due to discretization, etc.–).

a) Pattern generation and smoothing homotopy:

One of the key elements of our framework is the combination between a specific walking pattern generator based on “half-steps” and a simple homotopy that can quickly smooth sequences of (half-)steps. We present both in section II (we

N. Perrin is with CNRS/LAAS, Université de Toulouse UPS, INSA, INP, ISAE 7 avenue du colonel Roche, F-31077 Toulouse, France, and CNRS-AIST Joint Robotics Laboratory, UMI3218/CRT, Tsukuba, Japan.

O. Stasse and E. Yoshida are with CNRS-AIST Joint Robotics Laboratory, UMI3218/CRT, Tsukuba, Japan.

L. Baudouin and F. Lamiroux are with CNRS/LAAS, Université de Toulouse UPS, INSA, INP, ISAE 7 avenue du colonel Roche, F-31077 Toulouse, France.

introduced them in [32]). Before the use of the homotopy, the generated sequences are called “raw”, and simply correspond to concatenations of isolated half-steps. Isolated half-steps are obtained by fixing the position of the swing foot when it is at its maximum height: this puts us in the conditions of [23] where two statically stable “via point configurations” Q_{right} and Q_{left} are fixed and divide steps into two parts: an upward half-step, and a downward half-step. In [23] this restriction was already used in order to reduce the number of trajectories to consider; we use it in order to reduce the dimensionality of the input space. The simple homotopy that we use to smooth sequences of half-steps is, to our knowledge, new in the field of humanoid robotics (but it is based on the same principle as the techniques introduced in [28] and [29]).

b) Finite transition model and swept volume approximations:

Our pattern generator benefits from an input space of dimension only 3, and therefore we can cover it with a dense grid of only relatively few points. Each point corresponds to a sequence of two half-steps. For each point of the grid we first simulate the sequence of half-steps and check that it is feasible, i.e. that it contains no self-collision and does not violate the joints limits. The points which correspond to feasible trajectories will be the elements of our transition model. In section III we explain the construction of this transition model and show how, for each of its elements, we approximate the volume swept by the robot lower body during the execution of the corresponding trajectory. By speeding up collision checks these approximations will enable us to save a considerable computation time online.

At first it might seem strange to combine precomputed swept volumes and a smoothing homotopy that modifies trajectories, but in fact in the whole process the homotopy is only applied to one *feasible* trajectory returned by the planning process during which the swept volume approximations are extensively used. When the homotopy is applied we do not use precomputed swept volumes for the collision checks.

Several efficient swept volume approximation algorithms exist, such as for example the ones introduced in [22] and in [17]. Using such advanced specific algorithms will be part of our future work, but in this paper we validate our framework with a simpler approach. Since the highest priority is the evaluation time (because approximations are used multiple times at each iteration of the RRT algorithm), we use a generic approximation algorithm which stores the results in compact tree structures that, in our case, can be used to very quickly check for collisions with obstacles of the environment. This algorithm is a slight variant of the one introduced in [31]; the variant is presented in details in [30]. The use of swept volumes is widespread in robotics, especially for path planning (see [34], [15]), but relatively absent in the field of humanoid robotics, where, for the sake of computational efficiency, simpler bounding volumes are often preferred ([39], [10]).

c) An RRT variant for footstep planning:

The last part of our framework is the planning phase. Since we have a large transition model, the traditional A* search would perform poorly. Alternatives to A* have already been

proposed. For example in [13], Harada uses a PRM (Probabilistic Roadmap Method, see [21]) approach to plan footsteps: a tree of “milestone configurations” is grown from an initial configuration to a goal configuration. At first collisions are checked only at milestone configurations, and only once a candidate path has been found is the full trajectory verified. An issue of this approach is that even though the milestones are collision-free, collisions might occur in the candidate path. Thus the process might have to be restarted several times, leading to lengthy computations.

The idea of using an RRT algorithm [26] for footstep planning was introduced in [38], where a single-node-extending and a multi-node-extending RRT methods are proposed. In section IV we follow the single-node-extending method and present a new variant of the RRT algorithm for footstep planning, where we deal separately with the sets of left and right footsteps. When a new transition (i.e. a new footstep) is considered by the RRT algorithm, we test the corresponding approximated swept volume against all the points of the objects that are close enough (we suppose that the environment is known and that obstacles are represented by point clouds: each object is contained in a bounding box, and a finite set of points is covering the object exterior surface). If one of the points lies inside the swept volume, the transition is discarded. Using point clouds for collision detection is certainly not the safest nor the most efficient approach, but we believe that it illustrates well the performance of our framework: indeed, it is important to show that we are able to rapidly plan motions even if during each iteration of the RRT algorithm the number of collision queries is high, because in real applications unknown obstacles are often acquired as untreated sets of voxels, or large triangle soups or meshes. Preliminary experiments are presented in section V, where the robot HRP-2 quickly solves complicated footstep planning problems in environments cluttered with 3D obstacles.

In section VI, we improve our implementation by using meshes to represent the swept volume approximations and the PQP algorithm [24] for collision checks. This yields a further speed-up that enables us to perform real-time replanning. Experimental results are presented in section V which precedes a brief discussion on an extension of our framework to a continuous transition model (section VII), and the conclusion (section VIII).

II. A WALKING PATTERN GENERATOR BASED ON HALF-STEPS AND A SMOOTHING HOMOTOPY

We use a classical simplified model of the robot dynamics: the Linear Inverted Pendulum Model (see [19]). In this model the mass of the robot is assumed to be concentrated in its CoM (center of mass) which is supposed to be rigidly linked to and above the robot waist at all time. Besides, the robot is supposed to have only point contacts with the walking surface. The contact points are coplanar on a horizontal plane. Thus it behaves like an inverted pendulum, and an analysis of the subsequent equations leads to a further approximation which enables the decoupling of the dynamic differential equations

for the x-axis and y-axis. They can be written as follows:

$$p_x = Z(x) \quad (1)$$

$$p_y = Z(y) \quad (2)$$

$$\text{with } Z \triangleq Id - \frac{z_c}{g} \frac{d}{dt^2} \quad (3)$$

(x, y) are the (x-axis, y-axis) coordinates of the CoM of the robot, and z_c the height of the robot center of mass which is supposed constant during the step. Let us notice that Z is a linear operator acting on functions of time. (p_x, p_y) are the (x-axis, y-axis) coordinates of the virtual Zero Moment Point (ZMP). A classical balance criterion for biped walking is that the ZMP should stay at all time inside the polygon of support (see [37]).

In the article [14], Harada et al. show how analytical trajectories for both the CoM and the ZMP can be derived from these equations. The ZMP trajectory is a polynomial of the time variable t , and the CoM trajectory $\begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$ has the general following form:

$$\cosh\left(\sqrt{\frac{g}{z_c}} \cdot t\right) \begin{pmatrix} V_x \\ V_y \end{pmatrix} + \sinh\left(\sqrt{\frac{g}{z_c}} \cdot t\right) \begin{pmatrix} W_x \\ W_y \end{pmatrix} + \begin{pmatrix} r_x(t) \\ r_y(t) \end{pmatrix} \quad (4)$$

where $r_x(t)$ and $r_y(t)$ are polynomials entirely determined by $p_x(t)$ and $p_y(t)$, respectively.

From this equation we see that for a given ZMP profile, there are just enough free parameters (V_x, V_y, W_x, W_y) to set the initial horizontal position and speed of the CoM:

$$\begin{pmatrix} x(0) \\ y(0) \end{pmatrix} = \begin{pmatrix} V_x + r_x(0) \\ V_y + r_y(0) \end{pmatrix} \quad (5)$$

$$\begin{pmatrix} \dot{x}(0) \\ \dot{y}(0) \end{pmatrix} = \begin{pmatrix} \sqrt{\frac{g}{z_c}} \cdot W_x + \dot{r}_x(0) \\ \sqrt{\frac{g}{z_c}} \cdot W_y + \dot{r}_y(0) \end{pmatrix} \quad (6)$$

Using these equations, next we show how to produce the C-space (configuration space) trajectory corresponding to an isolated half-step. We just need to obtain a unique C-space trajectory from a small number of half-step parameters (as we will see, in our case it will be 3 parameters). If each of the robot legs has 6 degrees of freedom or more (the redundancy can be treated using generalized inverse kinematics, see [27]), this problem can be reduced to the generation of trajectories for the waist and the feet. Besides the compulsory constant waist height, we also made a few arbitrary and convenient restrictions (which reduce the number of parameters): the pitch and roll parameters of the waist orientation will stay at zero, and similarly the swing foot will always stay parallel to the walking surface. Thus, the lower body trajectory is entirely defined by 7 functions of the time:

- the waist horizontal position: $x(t), y(t)$ (we recall that the waist and CoM are rigidly fixed)
- the waist orientation: $\theta(t)$
- the swing foot position: $SF_x(t), SF_y(t), SF_z(t)$
- the swing foot orientation $SF_\theta(t)$

A. Producing isolated half-steps

In this section we only consider upward half-steps, but the method for the generation of downward half-steps trajectories is similar.

So, let us consider an upward half-step. In order to reduce the dimensionality of the parameter space, we make several assumptions. First, we fix and denote by T the duration of any half-step. Then, we assume that the initial and final speed of the ZMP and swing foot are 0, but we *do not* assume that the CoM initial and final speed are zero.

$$\dot{p}_x(0) = \dot{p}_y(0) = \dot{p}_x(T) = \dot{p}_y(T) = 0 \quad (7)$$

$$\dot{\theta}(0) = \dot{\theta}(T) = 0 \quad (8)$$

$$S\dot{F}_x(0) = S\dot{F}_y(0) = S\dot{F}_z(0) = S\dot{F}_\theta(0) = 0 \quad (9)$$

$$S\dot{F}_x(T) = S\dot{F}_y(T) = S\dot{F}_z(T) = S\dot{F}_\theta(T) = 0 \quad (10)$$

Second, the initial vertical projection on the ground of the CoM is equal to the ZMP initial position, i.e. at the barycenter of the feet centers. Taking the center of the support foot as the origin of the Euclidean space, it gives us:

$$x(0) = p_x(0) = \frac{SF_x(0)}{2} \quad (11)$$

$$y(0) = p_y(0) = \frac{SF_y(0)}{2} \quad (12)$$

We also assume that the final horizontal position of the CoM and ZMP coincide at the center of the support foot, and that the final orientations of the swing foot and waist are equal to the orientation of the support foot. Besides, the line passing through the centers of the feet final positions is orthogonal to this final orientation:

$$x(T) = p_x(T) = 0 \quad (13)$$

$$y(T) = p_y(T) = 0 \quad (14)$$

$$\theta(T) = SF_\theta(T) = 0 \quad (15)$$

$$SF_x(T) = 0 \quad (16)$$

As a consequence of these equations, the final and initial configurations are entirely determined by 5 parameters (as shown on Fig. 1):

$$SF_x(0), SF_y(0), SF_\theta(0), SF_y(T) \text{ and } SF_z(T).$$

Besides, concerning the derivatives at the boundaries, the only free parameters are $\dot{x}(0), \dot{x}(T), \dot{y}(0)$, and $\dot{y}(T)$. This adds up to a total of 9 free parameters.

Now, we show how the ZMP trajectory is defined. An upward half-step is divided into 3 phases: during the first one, of duration t_1 , the ZMP stays at the barycenter of the feet (and the feet keep their positions as well), so we have $p_x(t) = \frac{SF_x(0)}{2}, p_y(t) = \frac{SF_y(0)}{2}$, and thus $\dot{p}_x(t) = \dot{p}_y(t) = 0$. Then there is the “shift” phase, during which the ZMP quickly shifts from its initial position to its final position, reached at time t_2 . Then, from t_2 to T , the ZMP stays at its final position, so we have $p_x(t) = p_y(t) = \dot{p}_x(t) = \dot{p}_y(t) = 0$. During the “shift” phase we set p_x and p_y as third-degree polynomials determined by the respective boundary conditions

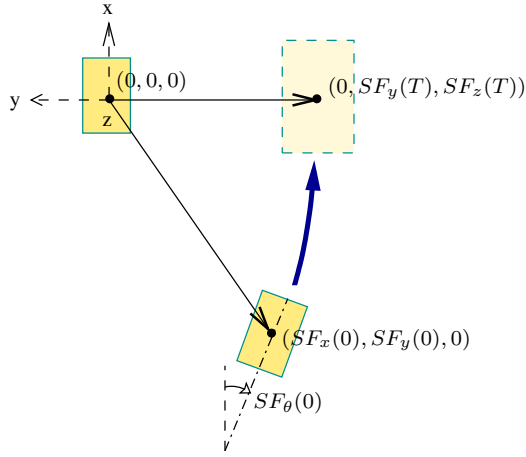


Fig. 1. Here we show an upward half-step from above. It is fully determined by the 5 parameters $SF_x(0)$, $SF_y(0)$, $SF_\theta(0)$, $SF_y(T)$ and $SF_z(T)$. A downward half-step is also fully determined by 5 parameters.

$p_x(t_1) = \frac{SF_x(0)}{2}$, $p_x(t_2) = p_x(t_1) = \dot{p}_x(t_2) = 0$, and $p_y(t_1) = \frac{SF_y(0)}{2}$, $p_y(t_2) = p_y(t_1) = \dot{p}_y(t_2) = 0$. For the downward half-step, even if the phase of double support and single support are inverted, we keep the same durations: the ZMP shift occurs between time t_1 and t_2 . In practice, we set $t_1 = T - t_2$.

Thanks to eq. (4), if we fix $SF_x(0)$, $SF_y(0)$, $\dot{x}(0)$, and $\dot{y}(0)$, we can get an analytical expression of the unique C^2 solution for $x(t)$ and $y(t)$ over $[0, T]$. The solution is unique because during the first phase, V_x , V_y , W_x and W_y are fixed by the following equations (obtained from eq. (5) and eq. (6)):

$$V_x = \frac{SF_x(0)}{2} - r_x(0) \quad (17)$$

$$V_y = \frac{SF_y(0)}{2} - r_y(0) \quad (18)$$

$$W_x = \sqrt{\frac{z_c}{g}} (\dot{x}(0) - r_x(0)) \quad (19)$$

$$W_y = \sqrt{\frac{z_c}{g}} (\dot{y}(0) - r_y(0)) \quad (20)$$

Moreover, the unique solution during the first phase leads to unique values for $x(t_1)$, $y(t_1)$, $\dot{x}(t_1)$, and $\dot{y}(t_1)$. This fixes the free parameters of the unique C^2 extension of the solution on $[t_1, t_2]$, and subsequently the free parameters of the unique C^2 extension over $[t_2, T]$. Nevertheless, those two unique C^2 solutions might violate the constraints $x(T) = 0$ and $y(T) = 0$ (eq. (13) and eq. (14)). Analyzing the impact of $\dot{x}(0)$ and $\dot{y}(0)$ in the analytical solutions, we can see that they have a monotonic influence over respectively $x(T)$ and $y(T)$, and that to one value of $x(T)$ (resp. $y(T)$) corresponds a unique value $\dot{x}(0)$ (resp. $\dot{y}(0)$). We implemented a dichotomic search for those values, and with simple methods avoided problems of numerical instability (using the fact that with only one ZMP shift and the boundary conditions $CoM(0) = ZMP(0)$ and $CoM(T) = ZMP(T)$, the solution CoM trajectories x and y are necessarily monotone).

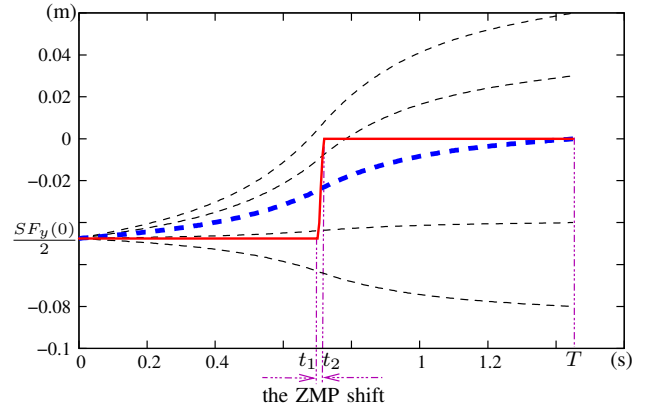


Fig. 2. We consider the upward half-step of Fig. 1, and show the corresponding ZMP trajectory along the y-axis: $p_y(t)$ (solid line). To this trajectory corresponds an infinity of C^2 solutions for $y(t)$ which all verify $y(0) = p_y(0) = \frac{SF_y(0)}{2}$, each of them being fully defined by $\dot{y}(0)$. We show several such C^2 solutions (dotted lines); the thick dotted line is the solution retained: it is the unique one verifying $y(T) = 0$.

Fig. 2 considers the half-step of Fig. 1, and it shows the trajectory of the ZMP along the y-axis as well as several C^2 solutions for $y(t)$, for different values of $\dot{y}(0)$. Only one solution is retained, the one with $y(T) = 0$. If the durations t_1 and $T - t_2$ are long enough, the values obtained for $\dot{x}(0)$, $\dot{x}(T)$, $\dot{y}(0)$ and $\dot{y}(T)$ can be neglected, and thus the CoM trajectories obtained are supposed to be C^2 continuous over $(-\infty, \infty)$. Performing tests on a real humanoid robot empirically validated this assumption: time discretization of the control law itself makes the neglected velocity unnoticeable. For the trajectories other than $x(t)$ and $y(t)$ ($\theta(t)$, $SF_x(t)$, $SF_y(t)$, $SF_z(t)$, $SF_\theta(t)$), we simply use polynomials of degree 3 that ensure C^2 smoothness and satisfying profiles, with a few specific constraints (e.g. in our implementation the swing foot always leaves the ground and lands vertically). So, we can completely define a half-step with 5 parameters (whether it is an upward half-step or a downward half-step). In our application, we decided to fix the maximum height of the swing foot ($SF_z(T)$), and the horizontal distance between the feet when the maximum height is reached (which fixes $SF_y(T)$). This puts us in the conditions of [23] where two “via point configurations” Q_{right} and Q_{left} are fixed. With these constraints only 3 parameters are needed to completely define a half-step. Once these parameters are set, we are capable of generating *unique* analytical solutions for the 7 functions of time that are required to produce the lower body trajectory in the C-space.

B. Smoothing a sequence of half-steps

Using the results of the previous section, we can generate C-space trajectories for isolated half-steps. Since they start and finish with zero speed, we can simply join them to produce sequences of half-steps. Alternating upward and downward half-steps will produce a walking motion. During each half-step, the motion is dynamically stable (i.e. not quasi-static, but the robot does not fall), however at the boundary of each half-step motion, the configuration is statically stable (i.e. it

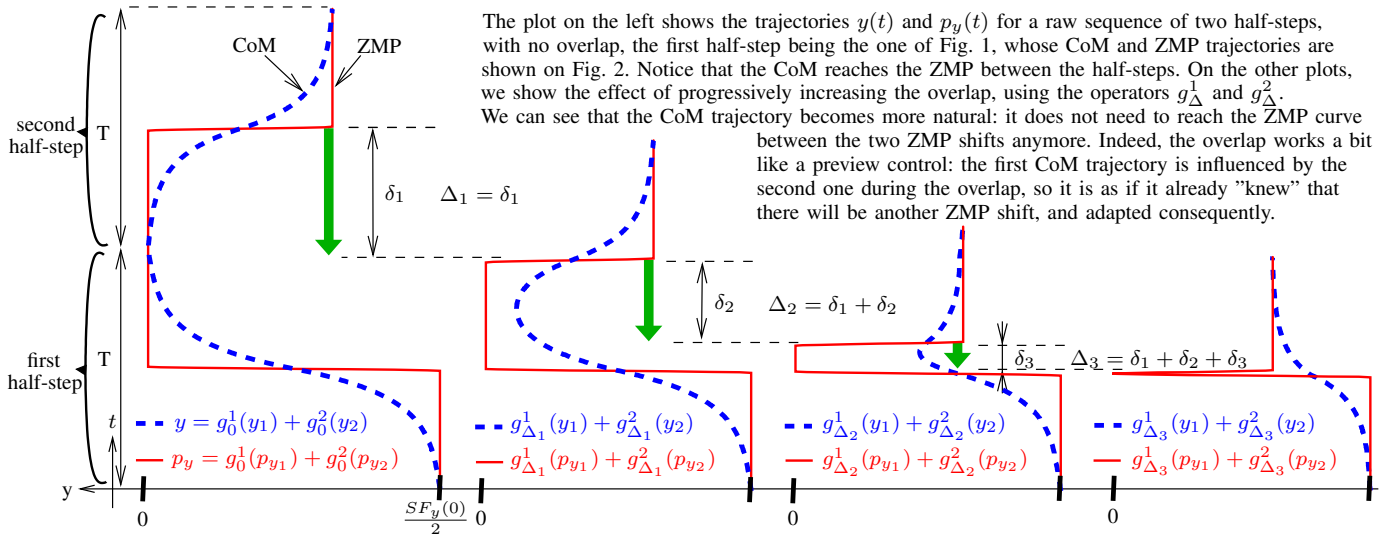


Fig. 3. Progressively increasing the overlap between two half-steps.

is a balanced posture). This is not a satisfactory result for several reasons. First, between each half-step the robot comes to a stop, so that the walk motion is not visually smooth. Recent walking pattern generators achieve much better results by using preview control (see [19]). In this section, we show how to continuously modify a sequence of half-steps using a simple homotopy, in order to make it faster and smoother along the same footstep sequence. We first show how to do so for a sequence of two half-steps, and start with the case of an upward half-step followed by a downward half-step.

1) *Upward then downward*: We consider an upward half-step followed by a downward half-step. Together the two half-steps make a classical full step: double support phase, then single support phase, and then double support phase again.

We recall that the whole C-space trajectory of the lower body during one half-step is generated by inverse geometry from 7 functions of the time. Since here we are dealing with two consecutive half-steps (with the same support foot), we have to consider 14 functions. Let us first consider for example the position of the waist (or CoM) along the y-axis, respectively for the upward half-step: $y_1(t)$, and the downward half-step: $y_2(t)$. We have $y_1(T) = y_2(0) = 0$. Let us define two operators g_{Δ}^1 and g_{Δ}^2 such that:

$$g_{\Delta}^1(f)(t) = \begin{cases} f(t) & \text{for } t \in (0, T) \\ f(T) & \text{for } t \in (T, 2T - \Delta) \end{cases} \quad (21)$$

$$g_{\Delta}^2(f)(t) = \begin{cases} 0 & \text{for } t \in (0, T - \Delta) \\ f(t - T + \Delta) - f(0) & \text{for } t \in (T - \Delta, 2T - \Delta) \end{cases} \quad (22)$$

$g_0^1(y_1) + g_0^2(y_2)$ corresponds to the simple concatenation of y_1 and y_2 without overlap. Knowing that $p_{y_1} = Z(y_1)$, $p_{y_2} = Z(y_2)$, and $y_1(T) = y_2(0) = 0$, it is quite easy to verify that for any $0 \leq \Delta \leq T$, $g_{\Delta}^1(p_{y_1}) = Z(g_{\Delta}^1(y_1))$, $g_{\Delta}^2(p_{y_2}) = Z(g_{\Delta}^2(y_2))$. And, since Z is a linear operator:

$$g_{\Delta}^1(p_{y_1}) + g_{\Delta}^2(p_{y_2}) = Z(g_{\Delta}^1(y_1) + g_{\Delta}^2(y_2)) \quad (23)$$

Operators g_{Δ}^1 and g_{Δ}^2 enable us to obtain new combined CoM and ZMP trajectories that still verify the Linear Inverted

Pendulum equations (eq. (1) and eq. (2)). Starting with $\Delta = 0$ and progressively increasing the value of Δ continuously modifies the CoM trajectory (starting from the initial trajectory $g_0^1(y_1) + g_0^2(y_2)$) to make the second ZMP shift (the one of y_2) happen earlier, creating an overlap of duration Δ between the two trajectories y_1 and y_2 . Fig. II-B illustrates this effect: when we increase the value of Δ we can see that the position of the CoM does not need to reach the center of the support foot.

We use the same operators, g_{Δ}^1 and g_{Δ}^2 , to produce an overlap between the functions of time corresponding to the waist orientation and swing foot position and orientation. Since the inverse geometry for the legs is a continuous function as long as we stay inside the joint limits, these operators used on the bodies trajectories actually implement a simple homotopy that continuously deforms the initial C-space trajectory into a smoother, more dynamic trajectory. The linearity of simplified differential equations has already been used in a similar way to produce mixtures of motions ([28] and [29]), but the purpose was to create new steps, not to smooth them nor speed them up.

In the case of an upward half-step followed by a downward half-step, increasing Δ reduces the duration of the single support phase, and therefore it increases the speed of the swing foot. To limit this effect we must bound Δ . Besides, if Δ is too large undesirable phenomena can occur, such as a negative swing foot height. To avoid these problems we set an upper bound such that the maximum overlap results in a moderately fast gait.

2) *Downward then upward*: We can apply the same technique to produce an overlap in the case of a downward half-step followed by an upward half-step. Since the last phase of the downward half-step and the first phase of the upward half-step are double support phases, the constraint on the swing foot motion disappears and the maximum bound on Δ becomes simply T (that is if $t_1 = T - t_2$, and it results in a double support phase whose duration is $t_2 - t_1$).

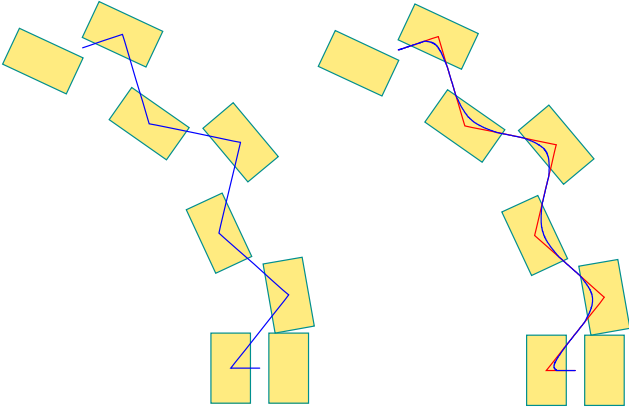


Fig. 4. We illustrate the “smoothing” of a raw sequence of half-steps. On the initial raw sequence (on the left), the support paths of the ZMP and CoM trajectories are superimposed. Then, after adjusting the overlaps, the ZMP support path stays the same but the CoM support path becomes smoother (on the right). We can smooth even more, but it reduces the duration of the single support phase that is directly linked with the swing foot speed. Therefore limitations on the swing foot speed constrain the smoothing process.

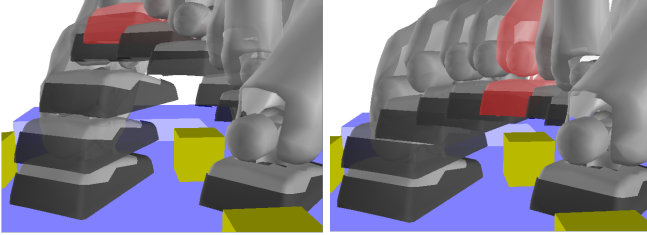


Fig. 5. On the left: a raw sequence of two half-steps avoiding a box on the ground. We can see that the swing foot reaches a high position. After smoothing (on the right), the trajectory has been modified so that the foot moves very close to the obstacle.

For longer sequences of half-steps, we can simply repeat the procedure to smooth the whole trajectory. Fig. 4 shows the results obtained with an example of raw sequence. After the smoothing, the CoM trajectory is visually smoother and besides, the new trajectory is much faster (about 3 times faster).

Changing overlaps inside a sequence of half-steps modifies the whole C-space trajectory: not only the CoM and ZMP, but also the swing foot trajectory: when the overlap is increased, the swing foot tends to move faster and closer to the ground. If one property must be preserved (for instance absence of collision), it must be checked after every modification. Since the smoothing by overlap is a continuous operator, we can use dichotomies to quickly find large acceptable values of overlaps. Let us consider an example for two consecutive half-steps. First, we try a predefined maximum overlap Δ_{max} : we simulate the part of the trajectory modified by this overlap, and check for collisions, self-collisions or joint limits violations. If these three constraints are verified, we keep this value and go to the next overlap. In the opposite case, we try again with overlap $\Delta_{max}/2$, etc. Fig. 5 shows the effect of the smoothing process on the swing foot trajectory: with the dichotomy we can quickly find a large overlap that keeps the trajectory collision-free.

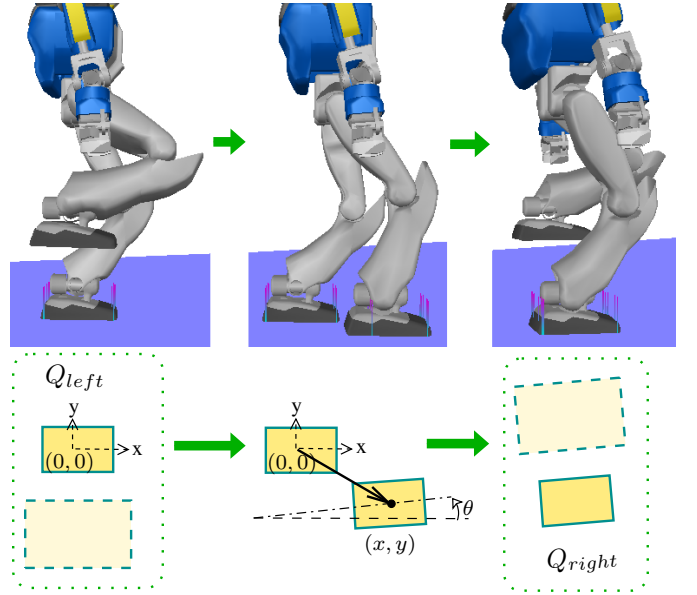


Fig. 6. Thanks to the two via point configurations Q_{left} and Q_{right} , a raw sequence of two half-steps can be entirely defined by only three parameters: x , y , and θ . Q_{left} and Q_{right} were chosen such that the swing foot is quite high. This provides good obstacle avoidance capabilities to raw sequences of steps and in the absence of obstacles, smoothing significantly reduces the height.

III. BUILDING THE TRANSITION MODEL AND THE SWEEP VOLUME APPROXIMATIONS

A. The transition model

Thanks to the walking pattern generator described in the previous section, we can produce isolated half-steps with only three parameters. If we join a downward half-step with the corresponding upward half-step, we obtain a trajectory that goes from Q_{left} to Q_{right} or Q_{right} to Q_{left} , and which is entirely defined by only three parameters, as shown on Fig. 6. We denote such a trajectory (expressed in the frame of the left foot) by $\langle Q_{left}, (x, y, \theta), Q_{right} \rangle$ or (expressed in the frame of the right foot) $\langle Q_{right}, (x, y, \theta), Q_{left} \rangle$. We also denote:

$$\mathcal{T}_l = \{ \langle Q_{left}, (x, y, \theta), Q_{right} \rangle \mid (x, y, \theta) \in \mathbb{R}^3 \},$$

and:

$$\mathcal{T}_r = \{ \langle Q_{right}, (x, y, \theta), Q_{left} \rangle \mid (x, y, \theta) \in \mathbb{R}^3 \}$$

We will interchangeably call the elements of \mathcal{T}_l or \mathcal{T}_r points (because of the bijection with \mathbb{R}^3), transitions (because the transition model will be a finite set of elements of \mathcal{T}_l), sequences (each element corresponds to a *downward half-step - upward half-step* sequence), or trajectories. By concatenating alternatively trajectories from \mathcal{T}_l and trajectories from \mathcal{T}_r , we obtain walk motions. With a symmetric robot (like HRP-2), \mathcal{T}_l and \mathcal{T}_r are symmetric in the sense that the feasibility of a sequence $\langle Q_{left}, (x, y, \theta), Q_{right} \rangle$ is equivalent to the feasibility of the sequence $\langle Q_{right}, (x, -y, -\theta), Q_{left} \rangle$, and that the corresponding swept volumes are symmetric. Therefore, only one transition model was built, on the space \mathcal{T}_l , but it can be used by symmetry on \mathcal{T}_r . To build it, as explained on Fig. 7, we first covered a reasonably large domain of \mathcal{T}_l with regularly

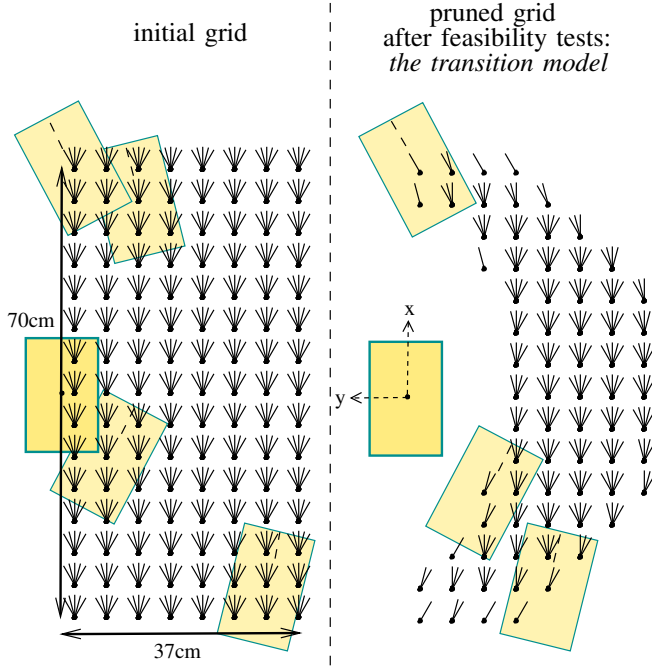
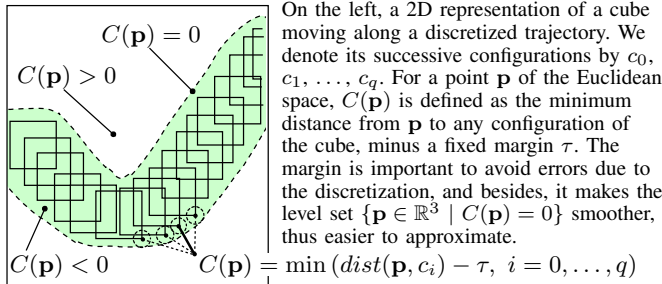


Fig. 7. An initial grid of 600 points covers the input space. To each of the 120 values of (x, y) correspond 5 possible orientations. All the corresponding trajectories (generated by the walking pattern generator presented in section II) are sequences of two half-steps. We test each of them, checking for self-collisions and joint limits violations, and remove all the unfeasible ones. The 276 remaining points form the transition model \mathcal{M}_l used for planning.



On the left, a 2D representation of a cube moving along a discretized trajectory. We denote its successive configurations by c_0, c_1, \dots, c_q . For a point \mathbf{p} of the Euclidean space, $C(\mathbf{p})$ is defined as the minimum distance from \mathbf{p} to any configuration of the cube, minus a fixed margin τ . The margin is important to avoid errors due to the discretization, and besides, it makes the level set $\{\mathbf{p} \in \mathbb{R}^3 \mid C(\mathbf{p}) = 0\}$ smoother, thus easier to approximate.

The 2D plot on the bottom-left shows how the approximation algorithm recursively divides the Euclidean space into small boxes in order to adaptively approximate the surface $C(\mathbf{p}) = 0$. The approximated surface defines an approximation of the volume swept by the cube. A view of this swept volume approximation is displayed on the 3D plot on the bottom right.

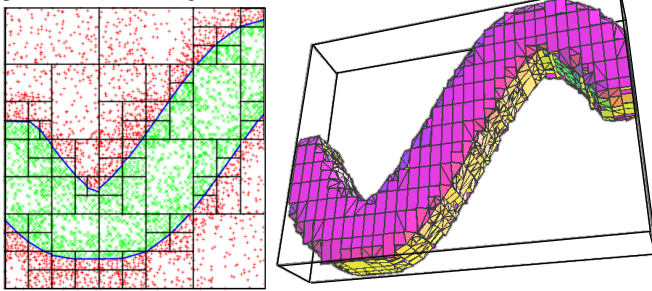


Fig. 8. An example of swept volume approximation. The data structure obtained is a bit similar to an adaptively sampled distance field (see [11]).

spaced points. Considering the robot (HRP-2) dimensions and joint limits, this domain was defined as the following box:

$$\mathcal{B}_l = \{\langle Q_{left}, (x, y, \theta), Q_{right} \rangle \mid x \in [-0.35m, +0.35m], \\ y \in [-0.37m, -0.02m], \theta \in [-30^\circ, +30^\circ]\}$$

We covered the box \mathcal{B}_l with 600 points (15 possible values for x , 8 possible values for y , 5 possible values for θ), and for each point, using discretized trajectories –one for each body of the robot legs–, we verified the feasibility of the corresponding *downward half-step - upward half-step* sequence. If any self-collision (which were checked using the algorithm introduced in [3]) or joint limit violation occurred, the point was discarded.

The 276 remaining points all correspond to feasible sequences, and they form the transition model.

We denote by $\mathcal{M}_l \subset \mathcal{B}_l$ this finite transition model, $\mathcal{M}_r \subset \mathcal{B}_r$ is defined by symmetry. We denote by $\mathcal{S}(\mathcal{M}_l, \mathcal{M}_r)$ the set of finite *feasible* sequences (s_1, s_2, \dots, s_n) alternating left foot and right foot support.

B. The swept volume approximations

For each of the 276 points of the transition model, we build an approximation of the volume swept by the lower part of the robot (from the knees down) during the corresponding *downward half-step - upward half-step* sequence. The algorithm used is the one described in [30]: given a transition $z \in \mathcal{M}_l$ it learns through adaptive sampling the sign of the mapping $C_z(\mathbf{p})$ which returns the distance (minus a fixed margin -1cm in our case–) between a point \mathbf{p} of the Euclidean space and the finite set of polyhedra consisting of all the configurations of the robot legs bodies along their discretized trajectories during the sequence corresponding to z . Fig. 8 illustrates an example of this process. The important property of the approximation algorithm used is that it stores the result in a tree structure which can be evaluated extremely quickly. The computation time saved is considerable: with the approximation, checking whether a point is outside or inside one of the swept volumes we consider is done in $4\mu\text{s}$. This is about 2,000 times faster than with the normal evaluation of $C_z(\mathbf{p})$.

For a transition $z = \langle Q_{left}, (x, y, \theta), Q_{right} \rangle \in \mathcal{M}_l$, we denote by $V_z(\mathbf{p})$ the corresponding swept volume approximation ($V_z(\mathbf{p}) > 0$ if and only if \mathbf{p} is outside the approximated swept volume). If $z' = \langle Q_{left}, (x, -y, -\theta), Q_{right} \rangle \in \mathcal{M}_r$, we can easily obtain the approximation $V_{z'}$ by applying a symmetry to V_z ; thus only 276 swept volume approximations are needed. With an Intel(R) Xeon(R) 2.00Ghz CPU, it took a bit less than 48 hours to generate them all, but we believe that by using state-of-the art swept volume approximation algorithms (and maybe only afterwards apply our algorithm to obtain reapproximations that can be evaluated very fast), we should be able to significantly reduce this offline computation time.

Fig. 9 shows 5 of the 276 swept volume approximations.

IV. FOOTSTEP PLANNING WITH A VARIANT OF RRT

In this section, we present a simple adaptation of the RRT algorithm for footstep planning, quite similar to the one introduced in [38].

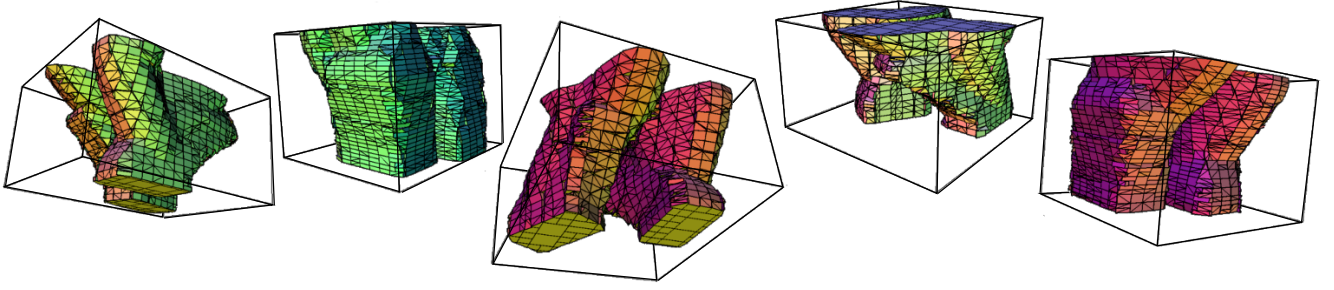


Fig. 9. 3D representations of 5 swept volumes approximations

Let us first define the search space. Since in our formalism we connect single support phases, the search space is $S = \{(q, x, y, \theta) \mid q \in \{Q_{left}, Q_{right}\}, (x, y, \theta) \in \mathbb{R}^3\}$, where q is the support foot, (x, y) the position of the support foot (relatively to a fixed reference), and θ its orientation (relatively to a fixed reference). The transition model being an alternation between \mathcal{M}_l and \mathcal{M}_r , we can apply transitions to states of the search space using the operator δ :

$$\begin{aligned} \delta((q, x, y, \theta), (q', x', y', \theta'), \bar{q}) \\ = (\bar{q}, x' \cos(\theta) - y' \sin(\theta), x' \sin(\theta) + y' \cos(\theta), \theta + \theta'), \end{aligned}$$

where $\overline{Q_{left}} = Q_{right}$ and $\overline{Q_{right}} = Q_{left}$. In practice, we will use only a compact subset of the search space, depending on the environment \mathcal{E} . We denote it by $S_{|\mathcal{E}}$. For example, if the robot stays in a $5m \times 5m$ room, we naturally use these dimensions to define $S_{|\mathcal{E}}$ and bound x and y . Considering the classical RRT algorithm (see [26]), the only operation that cannot be straightforwardly adapted to the context of footstep planning is the extension towards random samples (to find the nearest neighbor we use the Euclidean metric, ignoring the orientations). Let $(q, x, y, \theta) \in S$ be a random sample of the search space, and (q', x', y', θ') the nearest state in the search tree. In [38], two options are considered: either add to the tree all the successors of (q', x', y', θ') , or just one random successor. Due to the size of our transition model, we chose to follow the latter strategy. Fig. 10 shows one issue of this approach: in some cases, it is difficult to extend the search tree towards a given region. To cope with this problem, many options are possible. We simply chose to alternatively look for nearest states with left support foot and nearest states with right support foot. It leads to our RRT variant presented in Algorithm 1 (we stop the while loop when a path to the goal region has been found, or when a *sufficiently short* path has been found). We based our implementation on a fast and modular open-source code by Karaman and Frazzoli which uses kd-trees for fast nearest neighbor queries (this code implements RRT and RRT*, the algorithm introduced in [20]).

Further analyses and improvements of the variants of RRT for footstep planning can probably help to obtain faster results, but are out of the scope of this paper.

V. PRELIMINARY EXPERIMENTS

The framework presented in this paper was experimentally tested on the robot HRP-2.

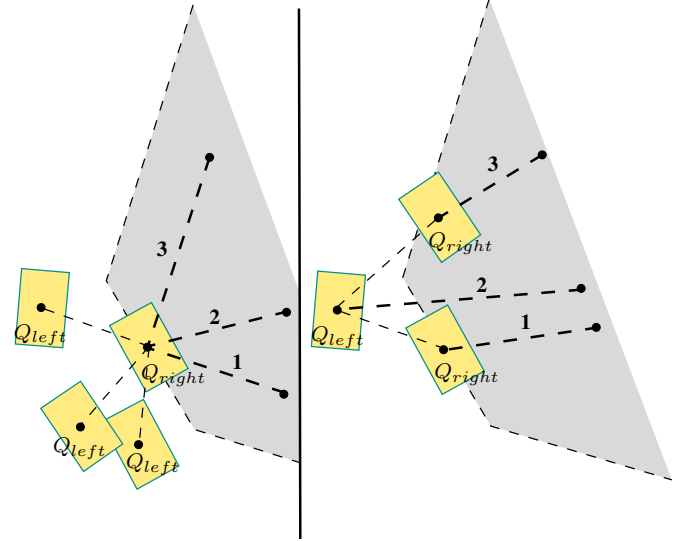


Fig. 10. The advantage of separating left and right support feet during nearest neighbor queries.

- *On the left* (global nearest neighbor): all the points in the gray region have the same nearest neighbor $(Q_{right}, x, y, \theta)$, but no successor of $(Q_{right}, x, y, \theta)$ is inside the gray region. Therefore numerous samples are required before extending the search tree towards the gray region.
- *On the right* (alternate nearest neighbors): when only states with left support foot are considered, the nearest neighbor will not be $(Q_{right}, x, y, \theta)$, but maybe one of its successors. With the alternation strategy, the search tree is more likely to quickly grow inside the gray region.

We studied the two Experimental Setups described on Fig. 11, where 2D obstacles (holes in the ground) are combined with 3D obstacles. The 3D obstacles shown on Fig. 11 have the same size as the ones in the real environment (see Fig. 12), but are smaller than the ones used for the collision checks (a margin is needed because of the robot drift during the real-world experiments).

The construction of the solution trajectory is divided into two parts: first, during the planning phase, just as explained in the previous section, we use a specific variant of RRT to find a sequence $(s_1, s_2, \dots, s_n) \in \mathcal{S}(\mathcal{M}_l, \mathcal{M}_r)$ which reaches the goal. Then, we use the homotopy of section II-B to smooth the sequence (s_1, s_2, \dots, s_n) , so that to obtain the final fast and dynamic trajectory that will be performed by the robot.

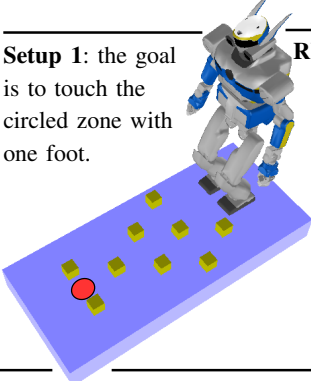
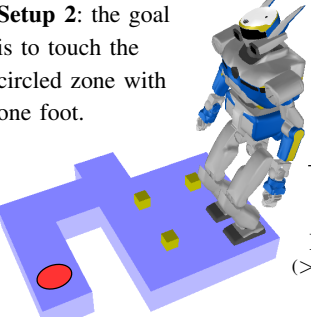
	time to reach a solution	number of steps of the solution	number of calls to the swept volumes approximations
Setup 1: the goal is to touch the circled zone with one foot.	RRT variant: (average on 10 attempts)		
	8.60s	21.1 steps	1,700,000
	A* search:		
	7.14s	6 steps	1,560,000
Setup 2: the goal is to touch the circled zone with one foot.	RRT variant: (average on 10 attempts)		
	29.8s	24.3 steps	1,920,000
	A* search:		
	FAIL (>10min)	–	–

Fig. 11. Experimental setups and results of the planning. The computations are made with an Intel(R) Xeon(R) 2.00Ghz CPU. Remark: in Setup 1, the exterior surface of the 3D obstacles (the boxes on the ground) is covered by 250 points. In Setup 2, the exterior surface of the 3D obstacles is covered by 75 points.

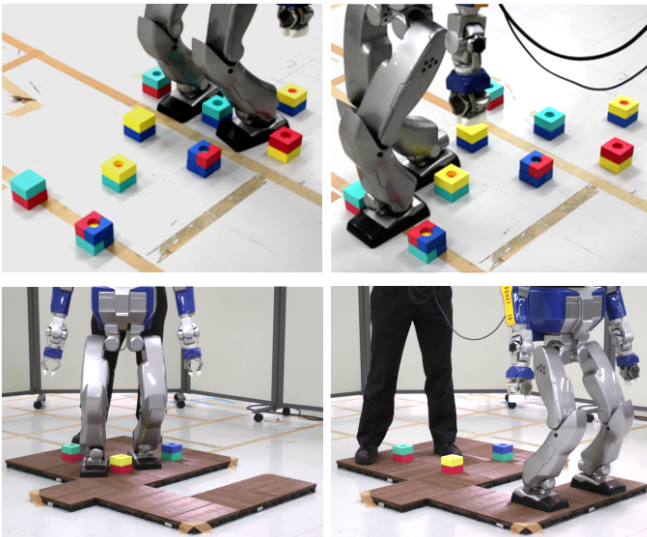


Fig. 12. Experimental results: the robot HRP-2 executing planned trajectories. Above: the so-called toy problem of walking in a child's bedroom avoiding toys on the ground.

Algorithm 1 RRT variant for footstep planning

```

1:  $\mathbb{T}.\text{init}(x_{\text{init}} \in S_{|\mathcal{E}})$ 
2:  $i \leftarrow 0$ 
3:  $\text{stop\_condition} \leftarrow \text{false}$ 
4: while  $\neg \text{stop\_condition}$  do
5:   Pick a random state  $x_{\text{rand}} \in S_{|\mathcal{E}}$ 
6:    $i++$ 
7:   if  $i == 0 \bmod 2$  then
8:      $x_{\text{near}} \leftarrow \{$  among states with left support foot,
9:       nearest neighbor of  $x_{\text{rand}}$  in the tree  $\mathbb{T}$ 
10:   else
11:      $x_{\text{near}} \leftarrow \{$  among states with right support foot,
12:       nearest neighbor of  $x_{\text{rand}}$  in the tree  $\mathbb{T}$ 
13:   end if
14:   Pick a random transition  $s_{\text{rand}} \in \mathcal{M}_l$ .
15:   Using the approximated swept volumes, verify that
16:   starting from state  $x_{\text{near}}$ , the transition  $s_{\text{rand}}$  does not
17:   collide with any point of the obstacle point clouds.
18:   if NO COLLISION then
19:      $\mathbb{T}.\text{add\_node}(\delta(x_{\text{near}}, s_{\text{rand}}))$ 
20:      $\mathbb{T}.\text{add\_edge}(x_{\text{near}}, s_{\text{rand}}, \delta(x_{\text{near}}, s_{\text{rand}}))$ 
21:     if  $\delta(x_{\text{near}}, s_{\text{rand}})$  is close enough to the goal and
22:     the path to  $\delta(x_{\text{near}}, s_{\text{rand}})$  is short enough then
23:        $\text{stop\_condition} \leftarrow \text{true}$ 
24:     end if
25:   end if
26: end while

```

A. The planning phase: RRT vs. A*

We implemented a classical A* search algorithm and compared it with the RRT variant introduced in the previous section. For the costs required by A* we used a simple heuristic where the estimated remaining cost is derived from the Euclidean distance, and the cost of a path is the sum of each (fixed) transition cost.

Better heuristics can often be obtained, such as for example heuristics derived from a mobile robot planner that looks for continuous paths between the initial position and the goal, but because they do not take stepping over capabilities into account, such heuristics tend to severely misjudge costs in very constrained environments like the ones we consider here (for a review on the association { A* + heuristic } see [5], chapter 8). Finding a robust heuristic that would perform well in challenging environments is as hard as solving the problem without using A*: that is why we tried to directly apply RRT. Other approaches of interest include planning algorithms based on inflated heuristics (see [12]): they usually find solutions faster than a classical A* search, but they are not as efficient as RRT to avoid local minima. Their main advantage over RRT is that they provide suboptimality bounds; however, due to the particularity of the problem of footstep planning, it is not clear whether such bounds can still be obtained in our context. Finally it might be interesting to try to adapt control-based strategies such as [35], but the adaptation would be far from straightforward.

In Setup 1 and 2 we fixed an upper bound, and stopped the

execution of RRT or A* as soon as a path of cost smaller than this upper bound was found.

As shown by the results on Setup 1, without strong local minima, the time needed by RRT and A* to find a solution is approximately the same, but A* finds a better trajectory cost (it finds solutions with fewer steps).

On the other hand we can see with the results on Setup 2 that when the transition model is large, A* seems much more sensitive to local minima than RRT: indeed A* fails to find a solution on Setup 2, whereas the RRT method consistently finds solutions in less than 40 seconds (and 29.8 seconds in average).

This is easily explainable because A* usually has to explore a subtree of fixed height h (which depends on the heuristic costs used) before being able to avoid a local minimum. Therefore it will try about $(|\mathcal{M}|^h - 1) \left(\frac{|\mathcal{M}|}{|\mathcal{M}| - 1} \right)$ transitions ($|\mathcal{M}|$ being the size of the transition model) before overcoming the local minimum. This can be done if both $|\mathcal{M}|$ and h are relatively small, but since in our case $|\mathcal{M}| = 276$, the complexity can quickly become insurmountable.

As a randomized approach, RRT does not have this caveat, and that is why we think it is more suitable than A* when the transition model is large.

A remark on the time saved thanks to the swept volume approximations: on the Setup 1 whose environment contain a lot of points (250), we can see that during their execution both the RRT variant and A* make about 200,000 calls to a swept volume approximation every second. Without the approximations, these 200,000 calls would be replaced by more than 26 minutes spent in collision checking.

B. The smoothing phase

Once a trajectory avoiding the obstacles has been found by the planner, since it consists in a concatenation of isolated half-steps, we can use the homotopy described in section II-B to smooth it. One overlap parameter has to be set for each pair of consecutive half-steps, and since the overlaps are independant, they can be set sequentially. This means that we can start to execute the trajectory on the robot even if only a few initial overlaps have been set, the next overlaps being computed during the execution of the trajectory. Let us notice that the dichotomy search for the best overlap time is an “any-time process” that can be interrupted if computation time is too long, the current result being anyway not worse than the initial raw motion. Another important remark: since we cannot know in advance the swept volumes for the trajectories involved in the smoothing processes, we have to use classical collision checks. We measured the overlaps computation time for 10 raw sequences of half-steps obtained in Setup 1, and 10 raw sequences obtained in Setup 2. In all cases, the duration of the smoothing was less than the final trajectory execution time. For the solutions in Setup 1, the average time needed for the smoothing was 14.4s, and the average execution time of the final trajectory was 31.1s. For the solutions in Setup 2, the average duration of the smoothing was 13.4, and the average execution time of the final trajectory was 41.6s. Fig. 13 illustrate the effect of the smoothing on the foot trajectories.

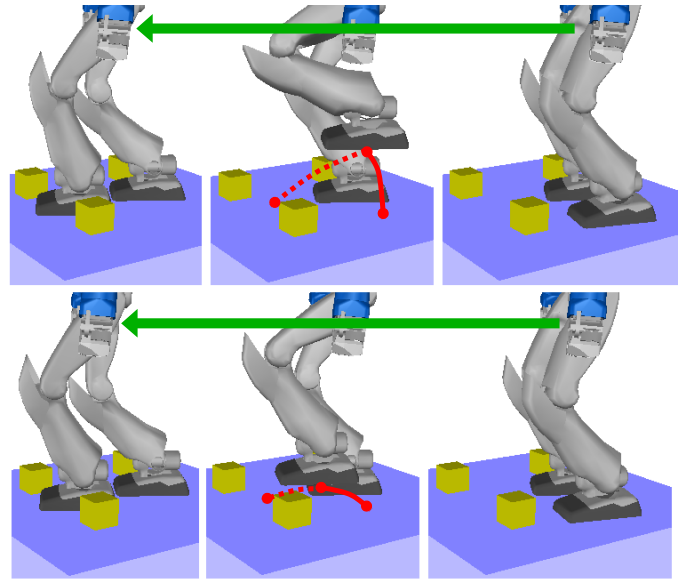


Fig. 13. Above: a raw sequence of two half-steps. Below: the smoothed sequence. When there are no obstacles, the swing foot trajectory of the smoothed sequence depends on the minimum time between two ZMP shifts, which is fixed in advance in order to bound the speed of the feet.

VI. A MORE ADVANCED IMPLEMENTATION

The way we deal with collisions in the preliminary experiments is clearly not optimal: we represent obstacles by covering them with points on their exterior surface, and all the points are always taken into account. The results showed that the swept volume approximations can be called a great number of times in a short period, proving that significant speed-up can be obtained compared to frequent collision checks along *a priori* unknown trajectories. What is more, in some case, point clouds are a very natural input, and it would be interesting to see if we can organize them in a good structure so that to use our approximation functions in an efficient way. This is beyond the scope of this paper, but we can already obtain better results by using state-of-the-art collision detection algorithms. First, we can notice that our swept volume approximations are defined by intersections of small boxes with planes. Thus, it is easy to construct meshes that describe the swept volume approximations (we actually use simplified meshes, i.e. they have a slightly simpler geometry than the initially precomputed approximations). With these 276 meshes, we will use the PQP algorithm [24] for collision checks. The main advantage we obtain by doing so is that when the obstacles are represented by classical meshes as well, PQP stores them in bounding volume hierarchies that reduce the complexity of collision checks.

With this method a significant speed-up is reached: with the Setup 2 of Fig. 11, we performed 1000 trials with a slightly faster CPU (Intel(R) Xeon(R) 2.40GHz) but overall in similar conditions. A solution was always found, and the average time required was only 1.60 seconds, which is almost 20 times faster than the preliminary results. The average number of steps of the solution was 28.5 steps, and in average 18,000 collision checks were needed before finding a solution.

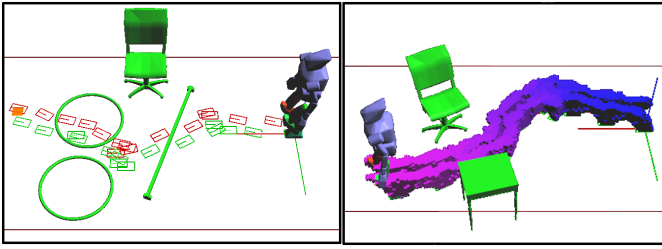


Fig. 14. On the left: a sequence of steps found in a complex environment. On the right, we show for one sequence of steps the concatenation of the swept volumes which are simplified meshes obtained from the original swept volume approximations. For the upper body simpler bounding boxes are used for the collision checks.

With this new implementation we tested our algorithm in more complex environments in simulation and also used it to perform real-time replanning in experiments where the position of the robot and obstacles is acquired by motion capture. The details of the framework used for these experiments are described in [2]. Fig. 14 shows two simulations, and Fig. 15 shows an experiment during which a bar placed 5cm above the ground is moved while the robot is executing its initial plan. The robot is then able to quickly find a new plan and successfully steps over the bar in its new configuration before reaching the goal.

VII. DISCUSSION ON AN EXTENSION TO CONTINUOUS TRANSITION MODELS

Even if the expressiveness of a continuous transition model can be approached by the one of a large finite transition model, a continuous transition model would still be preferable.

Several useful techniques would be easier to apply with a continuous transition model: local footstep modifications ([9], [8]), extraction of convex regions in the transition model in order to use optimization techniques to determine foot placements ([16]), path deformation ([18]), etc.

RRT and other sampling-based algorithms (e.g. PRM, see [21]) would be easier to adapt with a continuous transition model, so it would cause no problem at the planning phase. Besides, it would not be difficult to approximate the feasibility regions so as to obtain continuous transition models \mathcal{M}_l and \mathcal{M}_r (although it might be hard to obtain the guarantee that all transitions are indeed feasible). But then, the main issue would be the need to approximate swept volumes which depend on a continuous parameter $z \in \mathcal{M}_l$: instead of approximating (the sign of) $C_z(\mathbf{p})$ for a finite set of values of z , we would need to approximate $C(z, \mathbf{p})$ which depends on 6 parameters. It does not correspond anymore to the approximation of a single swept volume, so the state-of-the-art algorithms for swept volume approximation cannot be directly used, and we would probably need to keep a generic approximation algorithm, like the one used in this paper. Since it took already almost 48 hours to approximate the swept volumes of the finite transition model, for a continuous transition model an accurate approximation would probably be excessively time consuming. In that case it is likely that instead of trying to compute the swept volumes more efficiently, other collision detection routines should be

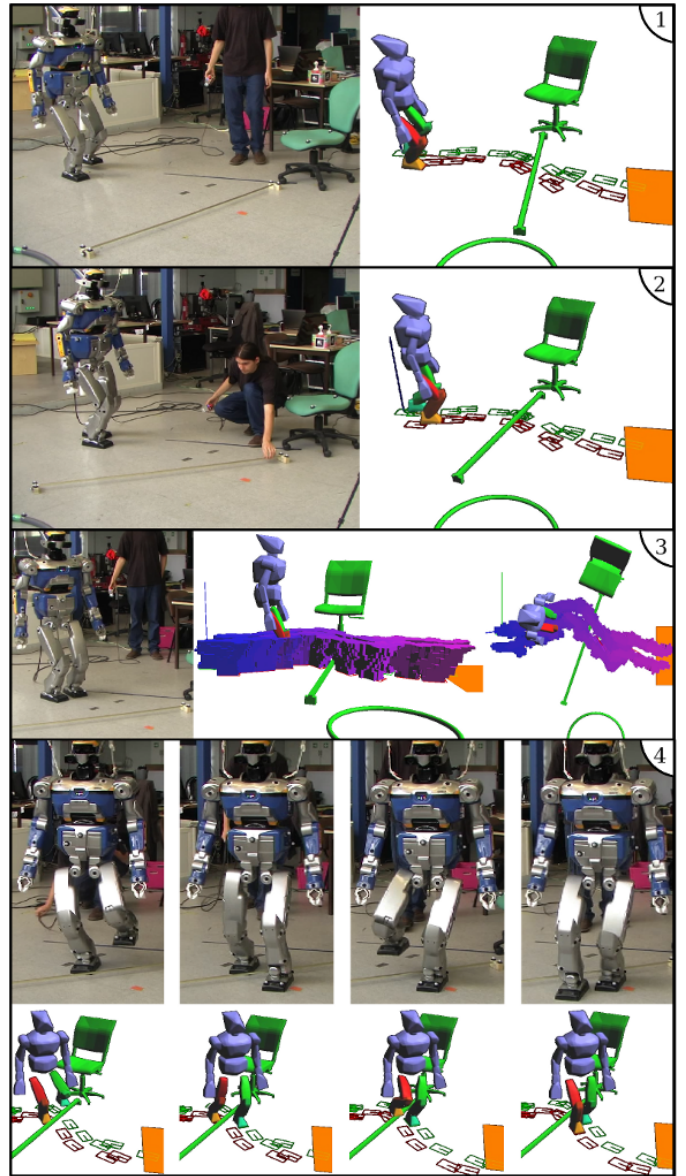


Fig. 15. (1): HRP-2 starts to execute the sequence initially found. (2): the bar is suddenly moved, and the current sequence of step would lead to collisions. (3): while walking, HRP-2 is able to compute a new sequence of steps towards the goal (we show the concatenation of the swept volumes which indeed avoid the bar). (4): the robot finally steps over the bar while at the same time it tries to optimize the rest of the path towards the goal. Remark: due to uncertainty on positions, we use a model of bar that is thicker than the actual bar.

taken into account, such as continuous collision detection [40], GPU-based approaches [25] or other variants (e.g. [36], [33], ...).

VIII. CONCLUSION

In this paper, we have described a novel and coherent framework for footstep planning, which includes a walking pattern generator based on half-steps, a simple homotopy for trajectory smoothing, swept volume approximations for fast collision checking, and an RRT variant for footstep planning. We used this framework on robot HRP-2 to quickly plan dynamic sequences of walk in environments cluttered with 3D and 2D obstacles. Although computed in a few seconds

and with the theoretical guarantee that they actually avoid the obstacles, the executed trajectories seem very natural: no pauses, no exaggerated motions to avoid small obstacles, and a large diversity of foot placements.

ACKNOWLEDGMENT

This work was supported by a grant from the RBLINK Project, Contract ANR-08-JCJC-0075-01.

REFERENCES

- [1] Y. Ayaz, K. Munawar, M. Bilal Malik, A. Konno, and M. Uchiyama. Human-like approach to footstep planning among obstacles for humanoid robots. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'06)*, 2006.
- [2] L. Baudouin, N. Perrin, T. Moulard, O. Stasse, and E. Yoshida. Real-time replanning using 3D environment for humanoid robot. Submitted to the 11th IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids'11). Available at <http://homepages.laas.fr/nperrin/submitted/humanoids11-lbaudouin.pdf>, 2011.
- [3] M. Benallegue, A. Escande, S. Miossec, and A. Kheddar. Fast c^1 proximity queries using support mapping of sphere-torus-patches bounding volumes. In *IEEE Int. Conf. on Robotics and Automation (ICRA'09)*, pages 483–488, 2009.
- [4] J.-M. Bourgeot, N. Ciso, and B. Espiau. Path-planning and tracking in a 3D complex environment for an anthropomorphic biped robot. In *IEEE Intl. Conf. on Intelligent Robots and Systems (IROS'02)*, pages 2509–2514, 2002.
- [5] J. Chestnutt. *Navigation Planning for Legged Robots*. PhD thesis, Carnegie Mellon University, 2007.
- [6] J. Chestnutt, J. Kuffner, K. Nishiwaki, and S. Kagami. Planning biped navigation strategies in complex environments. In *IEEE Int. Conf. on Humanoid Robots (Humanoids'03)*, 2003.
- [7] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade. Footstep planning for the honda asimo humanoid. In *IEEE Int. Conf. on Robotics and Automation (ICRA'05)*, pages 631–636.
- [8] J. Chestnutt, P. Michel, K. Nishiwaki, J. Kuffner, and S. Kagami. An intelligent joystick for biped control. In *IEEE Int. Conf. on Robotics and Automation (ICRA'06)*, pages 860–865, 2006.
- [9] J. Chestnutt, K. Nishiwaki, J.J. Kuffner, and S. Kagami. An adaptive action model for legged navigation planning. In *IEEE/RAS Int. Conf. on Humanoid Robots (Humanoids'07)*, pages 196–202, 2007.
- [10] M. Elmoggy, C. Habel, and J. Zhang. Online motion planning for hoap-2 humanoid robot navigation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'09)*, 2009.
- [11] S.F. Frisken, R.N. Perry, A.P. Rockwood, and T.R. Jones. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *27th annual conference on Computer graphics and interactive techniques (SIGGRAPH'00)*, pages 249–254, 2000.
- [12] J. P. Gonzalez and M. Likhachev. Search-based planning with provable suboptimality bounds for continuous state spaces. In *4th Annual Symposium on Combinatorial Search (SOCS'11)*, 2011.
- [13] K. Harada. Motion planning for a humanoid robot based on a biped walking pattern generator. In *Motion Planning for Humanoid Robots*, pages 192–197. Springer, 2010.
- [14] K. Harada, S. Kajita, K. Kaneko, and H. Hirukawa. An analytical method for real-time gait planning for humanoid robots. *I. J. Humanoid Robotics*, 3(1):1–19, 2006.
- [15] T. Hasegawa, K. Nakagawa, and K. Murakami. Collision-free path planning of a telerobotic manipulator based on swept volume of teleoperated manipulator. In *5th IEEE Int. Symp. on Assembly and Task Planning*, 2003.
- [16] A. Herdt, N. Perrin, and P.-B. Wieber. Walking without thinking about it. In *IEEE Int. Conf. on Intelligent Robots and Systems (IROS'10)*, 2010.
- [17] J.C. Himmelstein, E. Ferre, and J.-P. Laumond. Swept volume approximation of polygon soups. *IEEE Transactions on Automation Science and Engineering*, 7(1):177–183, 2009.
- [18] L. Jaillet and T. Simon. Path deformation roadmaps. In *7th Workshop on the Algorithmic Foundations of Robotics (WAFR'06)*, 2006.
- [19] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, and K. Yokoi. Biped walking pattern generation by using preview control of zero-moment point. In *IEEE Int. Conf. on Robotics and Automation (ICRA'03)*, pages 1620–1626, 2003.
- [20] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. In *Robotics Science and Systems VI*, 2010.
- [21] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12:566–580, 1996.
- [22] Y.J. Kim, G. Varadhan, M.C. Lin, and D. Manocha. Fast swept volume approximation of complex polyhedral models. In *8th ACM symposium on Solid modeling and applications*, pages 11–22, 2003.
- [23] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Footstep planning among obstacles for biped robots. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'01)*, pages 500–505, 2001.
- [24] E. Larsen, S. Gottschalk, M.C. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. In *IEEE Int. Conf. on Robotics and Automation (ICRA'00)*, pages 3719–3726, 2000.
- [25] C. Lauterbach, Q. Mo, and D. Manocha. gproximity: Hierarchical gpu-based operations for collision and distance queries. *Comput. Graph. Forum*, pages 419–428, 2010.
- [26] S.M. LaValle and J.J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *4th Workshop on the Algorithmic Foundations of Robotics (WAFR'00)*, pages 293–308, 2000.
- [27] Y. Nakamura and H. Hanafusa. Optimal redundancy control of robot manipulators. *Int. Journal of Robotics Research*, 6:32–42, 1987.
- [28] K. Nishiwaki, K. Nagasaka, M. Inaba, and H. Inoue. Generation of reactive stepping motion for a humanoid by dynamically stable mixture of pre-designed motions. In *IEEE Int. Conf. on Systems, Man, and Cybernetics*, pages 902 – 907, 1999.
- [29] K. Nishiwaki, T. Sugihara, S. Kagami, M. Inaba, and H. Inoue. On-line mixture and connection of basic motions for humanoid walking control by footprint specification. In *IEEE Int. Conf. on Robotics and Automation (ICRA'01)*, pages 4110–4115, 2001.
- [30] N. Perrin, O. Stasse, F. Lamiroux, and E. Yoshida. Adaptive sampling-based approximation of the sign of multivariate real-valued functions. Technical report, 2010. Available at <http://hal.archives-ouvertes.fr/docs/00/54/48/91/PDF/approx.pdf>.
- [31] N. Perrin, O. Stasse, F. Lamiroux, and E. Yoshida. Approximation of feasibility tests for reactive walk on hrp-2. In *IEEE Int. Conf. on Robotics and Automation (ICRA'10)*, pages 4243–4248, 2010.
- [32] N. Perrin, O. Stasse, F. Lamiroux, and E. Yoshida. A biped walking pattern generator based on “half-steps” for dimensionality reduction. In *IEEE Int. Conf. on Robotics and Automation (ICRA'11)*, pages 1270–1275, 2011.
- [33] H. Schmidl, N. Walker, and M. C. Lin. Cab: Fast update of obb trees for collision detection between articulated bodies. *Journal of Graphics Tools*, 9:1–9, 2004.
- [34] F. Schwarzler, M. Saha, and J.-C. Latombe. Exact collision checking of robot paths. In *5th Workshop on the Algorithmic Foundations of Robotics (WAFR'02)*, 2002.
- [35] I.A. Sucan and L.E. Kavraki. Kinodynamic motion planning by interior-exterior cell exploration. In *8th Workshop on the Algorithmic Foundations of Robotics (WAFR'08)*, 2008.
- [36] M. Tang, Y. J. Kim, and D. Manocha. CCQ: Efficient local planning using connection collision query. In *9th Workshop on the Algorithmic Foundations of Robotics (WAFR'10)*, pages 229–247, 2010.
- [37] M. Vukobratovic and B. Borovac. Zero-moment point – thirty five years of its life. *Int. Journal of Humanoid Robotics*, 1(1):157–173, 2004.
- [38] Z. Xia, G. Chen, J. Xiong, Q. Zhao, and K. Chen. A random sampling-based approach to goal-directed footstep planning for humanoid robots. In *IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics (AIM'09)*, pages 168–173, 2009.
- [39] E. Yoshida, I. Belousov, C. Esteves, and J.-P. Laumond. Humanoid motion planning for dynamic tasks. In *IEEE/RAS Int. Conf. on Humanoid Robots (Humanoids'05)*, pages 1–6, 2005.
- [40] X. Zhang, S. Redon, M. Lee, and Y. J. Kim. Continuous collision detection for articulated models using taylor models and temporal culling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, 26(3):15, 2007.