Fast humanoid robot collision-free footstep planning using swept volume approximations
Nicolas Perrin, Olivier Stasse, Léo Baudouin, Florent Lamiraux, Eiichi Yoshida

To cite this version:

HAL Id: hal-00662550
https://hal.archives-ouvertes.fr/hal-00662550
Submitted on 24 Jan 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Fast humanoid robot collision-free footstep planning using swept volume approximations

Nicolas Perrin, Olivier Stasse, Léo Baudouin, Florent Lamiraux 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* + 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. But 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 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 “via point configurations” \( Q_{\text{right}} \) and \( Q_{\text{left}} \) (corresponding to balanced postures) are fixed and divide steps into two parts: an upward half-step, and a downward half-step. In [23] this restriction was 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 some experiments of real-time replanning.

Section VII contains a brief discussion on an extension of our framework to a continuous transition model, and section VIII is the conclusion.

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 times. 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:

\[ p_x = Z(x) \]  
\[ p_y = Z(y) \]

with \( Z = \frac{g}{d} \frac{d}{dt^2} \) \( t \) (3)

\((x, y)\) are the (x-axis,y-axis) coordinates of the CoM of the robot, and \( z_c \) is 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 \((ZMP)\). A classical balance criterion for biped walking is 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.

\[ \dot{p}_x(0) = \dot{p}_y(0) = \dot{p}_x(T) = \dot{p}_y(T) = 0 \]  
\[ \dot{\theta}(0) = \dot{\theta}(T) = 0 \]  
\[ SF_x(0) = SF_y(0) = SF_z(0) = SF_\theta(0) = 0 \]  
\[ SF_x(T) = SF_y(T) = SF_z(T) = SF_\theta(T) = 0 \]

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:

\[ x(0) = p_x(0) = \frac{SF_x(0)}{2} \]  
\[ y(0) = p_y(0) = \frac{SF_y(0)}{2} \]

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 swing foot orientation and the initial and final orientation of the waist are equal to the support foot orientation (at this stage the orientation of the waist changes only during downward half-steps). Besides, the line passing through the centers of the final positions of the feet is orthogonal to this orientation:

\[ x(T) = p_x(T) = 0 \]  
\[ y(T) = p_y(T) = 0 \]  
\[ \theta(0) = \theta(T) = SF_\theta(T) = 0 \]  
\[ SF_z(T) = 0 \]

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 \( \dot{x}(0), \dot{x}(T), \dot{y}(0), \text{ and } \dot{y}(T) \). 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 \( p_x(t) = \frac{SF_x(0)}{2}, \text{ and } p_y(t) = \frac{SF_y(0)}{2}, \) 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
\[ p_x(t_1) = \frac{SF_x(0)}{2}, \quad p_x(t_2) = \frac{SF_x(T)}{2}, \quad p_y(t_1) = \frac{SF_y(0)}{2}, \quad p_y(t_2) = \frac{SF_y(T)}{2}, \quad p_y(t_1) = p_y(t_2) = 0. \]
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)).

\[
\begin{align*}
V_x &= \frac{SF_x(0)}{2} - r_x(0) \\
V_y &= \frac{SF_y(0)}{2} - r_y(0) \\
W_x &= \sqrt{\frac{2}{g}} (\dot{x}(0) - r_x(0)) \\
W_y &= \sqrt{\frac{2}{g}} (\dot{y}(0) - r_y(0))
\end{align*}
\]

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 unstability (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).

Moreover, the solution obtained is supposed to be continuous 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 \( (SF_y(T)) \), 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.

\[ \text{B. Smoothing a sequence of half-steps} \]

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. Each half-step trajectory is dynamic in the sense that the inertial forces play an important role in maintaining the balance (the trajectories are not quasi-static). However at the end of each half-step a
balanced posture is reached with zero speed. This is not a satisfactory result because between each half-step the robot comes to a stop, so the walk motion is not visually smooth, and rather slow. 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. 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 \( g^1_\Delta \) and \( g^2_\Delta \) such that:

\[
g^1_\Delta(f)(t) = \begin{cases} f(t) & \text{for } t \in (0, T) \\ f(T) & \text{for } t \in (T, 2T - \Delta) \end{cases}
\]

\[
g^2_\Delta(f)(t) = \begin{cases} 0 & \text{for } t \in (T, T + \Delta) \\ f(t - T + \Delta) - f(0) & \text{for } t \in (T - \Delta, 2T - \Delta) \end{cases}
\]

\( g^0_1(y_1) + g^0_2(y_2) \) 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 \( 0 \leq \Delta \leq T \), \( g^1_\Delta(p_{y_1}) = Z(g^1_\Delta(y_1)) \), \( g^2_\Delta(p_{y_2}) = Z(g^2_\Delta(y_2)) \). And, since \( Z \) is a linear operator:

\[
g^1_\Delta(p_{y_1}) + g^2_\Delta(p_{y_2}) = Z(g^1_\Delta(y_1) + g^2_\Delta(y_2))
\]

Operators \( g^1_\Delta \) and \( g^2_\Delta \) 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 \( \Delta \) 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 \( p_{y_2} \)) happen earlier, creating an overlap of duration \( \Delta \) between the two trajectories \( y_1 \) and \( y_2 \). Fig. 3 illustrates this effect: when we increase the value of \( \Delta \) 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, \( g^1_\Delta \) and \( g^2_\Delta \), 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 \( \Delta \) 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 \( \Delta \). Besides, if \( \Delta \) 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 \( \Delta \) becomes simply \( T \) (that is if \( t_1 = T - t_2 \), and it results in a double
support phase whose duration is $t_2 - t_1$.

For longer sequences of half-steps, we can simply repeat the procedure to smooth the whole trajectory, setting the overlaps one by one. 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 collisions), 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. We predefine a maximum overlap $\Delta_{\text{max}}$, and first, we simulate the part of the trajectory modified by the overlap $\Delta_{\text{max}}$, and check for collisions, self-collisions and joint limit violations. If none of these events occur, we set the overlap to $\Delta_{\text{max}}$. Otherwise, we use a dichotomy starting at $\Delta_{\text{max}}/2$ to quickly converge towards the largest “good” overlap value below $\Delta_{\text{max}}$. 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.

III. BUILDING THE TRANSITION MODEL AND THE SWEPt 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 $Q_{\text{left}}$ to $Q_{\text{right}}$ or $Q_{\text{right}}$ to $Q_{\text{left}}$, 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 $(Q_{\text{left}}, (x, y, \theta), Q_{\text{right}})$ or (expressed in the frame of the right foot) $(Q_{\text{right}}, (x, y, \theta), Q_{\text{left}})$. We also denote:

$$T_l = \{ (Q_{\text{left}}, (x, y, \theta), Q_{\text{right}}) \mid (x, y, \theta) \in \mathbb{R}^3 \},$$

and:

$$T_r = \{ (Q_{\text{right}}, (x, y, \theta), Q_{\text{left}}) \mid (x, y, \theta) \in \mathbb{R}^3 \}$$

We will interchangeably call the elements of $T_l$ or $T_r$ points (because of the bijection with $\mathbb{R}^3$), transitions (because the transition model will be a finite set of elements of $T_l$), sequences (each element corresponds to a downward half-step - upward half-step sequence), or trajectories. By concatenating alternatively trajectories from $T_l$ and trajectories from $T_r$, we obtain walk motions. With a symmetric robot (like HRP-2), $T_l$ and $T_r$ are symmetric in the sense that the feasibility of a sequence $(Q_{\text{left}}, (x, y, \theta), Q_{\text{right}})$ is equivalent to the feasibility of the sequence $(Q_{\text{right}}, (x, y, \theta), Q_{\text{left}})$, and that the corresponding swept volumes are symmetric. Therefore, only...
initial grid

pruned grid after feasibility tests: the transition model

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 limit 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 \(c_0, c_1, \ldots, c_q\). For a point \(p\) of the Euclidean
space, \(C(p)\) is defined as the minimum distance from \(p\) to any configuration of
the cube, minus a fixed margin \(\tau\). The margin is important to avoid errors due to
the discretization, and besides, it makes the level set \(\{p \in \mathbb{R}^3 \mid C(p) = 0\}\) 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 \(C(p) = 0\). 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]).

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

\[ \mathcal{B}_l = \{ (Q_{lft}, (x, y, \theta), Q_{right}) \mid x \in [-0.35m, +0.35m], \]
\[ y \in [-0.37m, -0.02m], \theta \in [-30°, +30°] \} \]

We covered the box \(\mathcal{B}_l\) with 600 points (15 possible values
for \(x\), 8 possible values for \(y\), 5 possible values for \(\theta\)),
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 se-
tances, 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 \(S(\mathcal{M}_l, \mathcal{M}_r)\) the set
of finite feasible sequences \(s_1, s_2, \ldots, s_n\) 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 down-
ward 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
\(C_z(p)\) which returns the distance (minus a fixed margin –1cm
in our case–) between a point \(p\) 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\(\mu\)s. This is about 2,000 times faster
than with the normal evaluation of \(C_z(p)\).

For a transition \(z = (Q_{lft}, (x, y, \theta), Q_{right}) \in \mathcal{M}_l\), we
denote by \(V_z(p)\) the corresponding swept volume approximation
\(V_z(p) > 0\) if and only if \(p\) is outside the approximated swept
volume. If \(z' = (Q_{lft}, (x, -y, -\theta), Q_{right}) \in \mathcal{M}_r\), 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 VARIATION 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].

Let us first define the search space. Since in our formalism we connect single support phases, the search space is

\[ S = \{(q, x, y, \theta) | \langle q, (x', y', \theta') \rangle = (\overline{Q}_{left}, \overline{Q}_{right}) \}, (x, y, \theta) \in \mathbb{R}^3 \}, \]

where \( q \) is the support foot, \((x, y)\) the position of the support foot (relatively to a fixed reference), and \( \theta \) 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 \( \delta \):

\[
\delta (\langle q, x, y, \theta, \rangle; (q, (x', y', \theta')) = (\overline{Q}_{left}, \overline{Q}_{right}) = (Q_{left}, Q_{right}) \}
\]

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 \( (q', x', y', \theta') \) the nearest state in the search tree. In [38], two options are considered: either add to the tree all the successors of \( (q', x', y', \theta') \), 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.

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 \( (s_1, s_2, \ldots, s_n) \in S(\mathcal{M}_l, \mathcal{M}_r) \) which reaches the goal. Then, we use the homotopy of section II-B to smooth the sequence \( (s_1, s_2, \ldots, s_n) \), so that to obtain the final fast
Algorithm 1 RRT variant for footstep planning

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

and dynamic trajectory that will be performed by the robot.

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.

<table>
<thead>
<tr>
<th>Setup 1: the goal is to touch the circled zone with one foot.</th>
<th>RRT variant: (average on 10 attempts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>time to reach a solution</td>
<td>number of steps of the solution</td>
</tr>
<tr>
<td>8.60s</td>
<td>21.1 steps</td>
</tr>
</tbody>
</table>

A* search:

<table>
<thead>
<tr>
<th>Setup 2: the goal is to touch the circled zone with one foot.</th>
<th>RRT variant: (average on 10 attempts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAIL</td>
<td>29.8s</td>
</tr>
</tbody>
</table>

A* search:

<table>
<thead>
<tr>
<th>Setup 1: the goal is to touch the circled zone with one foot.</th>
<th>RRT variant: (average on 10 attempts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>time to reach a solution</td>
<td>number of steps of the solution</td>
</tr>
<tr>
<td>8.60s</td>
<td>21.1 steps</td>
</tr>
</tbody>
</table>

A* search:

<table>
<thead>
<tr>
<th>Setup 2: the goal is to touch the circled zone with one foot.</th>
<th>RRT variant: (average on 10 attempts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>time to reach a solution</td>
<td>number of steps of the solution</td>
</tr>
<tr>
<td>29.8s</td>
<td>24.3 steps</td>
</tr>
</tbody>
</table>

A* search:

<table>
<thead>
<tr>
<th>Setup 1: the goal is to touch the circled zone with one foot.</th>
<th>RRT variant: (average on 10 attempts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAIL</td>
<td>29.8s</td>
</tr>
</tbody>
</table>

A* search:

<table>
<thead>
<tr>
<th>Setup 2: the goal is to touch the circled zone with one foot.</th>
<th>RRT variant: (average on 10 attempts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAIL</td>
<td>29.8s</td>
</tr>
</tbody>
</table>

A* search:

<table>
<thead>
<tr>
<th>Setup 1:</th>
<th>RRT variant: (average on 10 attempts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>time to reach a solution</td>
<td>number of steps of the solution</td>
</tr>
<tr>
<td>8.60s</td>
<td>21.1 steps</td>
</tr>
</tbody>
</table>

A* search:

<table>
<thead>
<tr>
<th>Setup 2:</th>
<th>RRT variant: (average on 10 attempts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>time to reach a solution</td>
<td>number of steps of the solution</td>
</tr>
<tr>
<td>29.8s</td>
<td>24.3 steps</td>
</tr>
</tbody>
</table>

A* search:

<table>
<thead>
<tr>
<th>Setup 1:</th>
<th>RRT variant: (average on 10 attempts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAIL</td>
<td>29.8s</td>
</tr>
</tbody>
</table>

A* search:

<table>
<thead>
<tr>
<th>Setup 2:</th>
<th>RRT variant: (average on 10 attempts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAIL</td>
<td>29.8s</td>
</tr>
</tbody>
</table>

A* search:

<table>
<thead>
<tr>
<th>Setup 1:</th>
<th>RRT variant: (average on 10 attempts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAIL</td>
<td>29.8s</td>
</tr>
</tbody>
</table>

A* search:

<table>
<thead>
<tr>
<th>Setup 2:</th>
<th>RRT variant: (average on 10 attempts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAIL</td>
<td>29.8s</td>
</tr>
</tbody>
</table>
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 $\binom{|M|}{h} - 1$ transitions ($|M|$ being the size of the transition model) before overcoming the local minimum. This can be done if both $|M|$ and $h$ are relatively small, but since in our case $|M| = 276$, 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 independent, 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 41.6s. Fig. 13 illustrate the effect of the smoothing on the foot trajectories.

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® Xeon® 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
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 ([19], [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 $M_l$ and $M_r$ (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 $z \in M_l$; instead of approximating (the sign of) $C_z(p)$ for a finite set of values of $z$, we would need to approximate $C(z, p)$ 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 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 the 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


Nicolas Perrin graduated from the Ecole Normale Supérieure de Lyon in 2009. He received the M.Sc. degree in Mathematical Logic and Foundations of Computer Science from the University of Paris VII in 2008. He is currently Ph.D. student at LAAS-CNRS, Toulouse, France. During his Ph.D. studies he worked from 2008 to 2011 at CNRS-AIST JRL (Joint Robotics Laboratory), Tsukuba, Japan. His current research is in humanoid robots and motion planning.

Olivier Stasse is a Senior Researcher (CR-1) at LAAS-CNRS, Toulouse. He has been assistant professor in Computer Science at University of Paris 13. He received a Ph.D. in Intelligent Systems (2000) from University of Paris 6. His research interests include humanoids robots, and more specifically motion generation motivated by vision. From 2003 to 2011, he was with the Joint French-Japanese Robotics Laboratory (JRL) in Tsukuba, Japan. He has been Finalist for the Best Paper Award at ICAR 2007, finalist for the Best Video Award at ICRA 2007 and received the Best Paper Award at ICMA 2006.

Léo Baudouin graduated in Master Image and Vision from Blaise Pascal University of Clermont-Ferrand in 2011. He will graduate from Clermont-Ferrand IFMA engineering school of mechatronics in 2012. In 2011 he worked at CNRS-AIST JRL (Joint Robotics Laboratory) in Japan and at LAAS-CNRS in France on humanoid robot real-time re-planning.

Florent Lamiraux graduated from the Ecole Polytechnique Paris in 1993. He received the Ph.D. degree in Computer Science from the Institut National Polytechnique de Toulouse in 1997 for his research on Mobile Robots. He worked two years at Rice University as a Research Associate. He is currently Directeur de Recherche at LAAS-CNRS, working in humanoid robots.

Eiichi Yoshida received M.E. and Ph.D. degrees on Precision Machinery Engineering from Graduate School of Engineering, the University of Tokyo in 1993 and 1996 respectively. In 1996 he joined former Mechanical Engineering Laboratory, Tsukuba, Japan. He is currently senior research scientist, in Intelligent Systems Research Institute, National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan. From 1990 to 1991, he was visiting research associate at Swiss Federal Institute of Technology at Lausanne (EPFL). He served as Co-Director of AIST/JS-CNRS/ST2I Joint French-Japanese Robotics Laboratory (JRL) at LAAS-CNRS, Toulouse, France, from 2004 to 2008. He is currently Co-Director of CNRS-AIST JRL (Joint Robotics Laboratory), UMI3218/CRT, AIST, Japan, since 2009. His research interests include robot task and motion planning, modular robotic systems, and humanoid robots.