A Biped Walking Pattern Generator based on "Half-Steps" for Dimensionality Reduction
Nicolas Perrin, Olivier Stasse, Florent Lamiraux, Eiichi Yoshida

To cite this version:
Nicolas Perrin, Olivier Stasse, Florent Lamiraux, Eiichi Yoshida. A Biped Walking Pattern Generator based on "Half-Steps" for Dimensionality Reduction. IEEE International Conference on Robotics and Automation (ICRA 2011), May 2011, Shanghai, China. hal-00593659

HAL Id: hal-00593659
https://hal.archives-ouvertes.fr/hal-00593659
Submitted on 16 May 2011

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.
A Biped Walking Pattern Generator based on “Half-Steps” for Dimensionality Reduction

Nicolas Perrin* and Olivier Stasse** and Florent Lamiraux*** and Eiichi Yoshida**

Abstract— We present a new biped walking pattern generator based on “half-steps”. Its key features are a) a 3-dimensional parametrization of the input space, and b) a simple homotopy that efficiently smooths the gait trajectory corresponding to a fixed sequence of steps. We show how these features can be ideally combined in the framework of sampling-based gait planning. We apply our approach to the robot HRP-2 and are able to quickly produce smooth and dynamically stable trajectories that are solutions to a difficult problem of gait planning.

I. INTRODUCTION, RELATED WORK AND OVERVIEW

A. Introduction and related work

In this article, we consider the following framework for fast gait planning for humanoid robots:

1) Use a low dimensional space of steps to plan a first “raw” feasible gait sequence towards a goal location, using a sampling-based planning algorithm.

2) Smooth the gait obtained, and play the result on the robot.

Sampling-based planning algorithms have been extensively used in the context of gait planning for humanoid robots (see for example [12], [2]). One of the problems often encountered is the gap between the trajectories used for the search of footsteps, and the trajectories used for the final whole-body motion generation. For instance, the sequence of footsteps planned might lead to self-collisions when used as an input for the whole-body motion generation. In this article, we seek coherence at this level and introduce a new walking pattern generator aimed at being used in both steps of the framework. The fact that it is based on half-steps (we define what is a half-step in section II-A) gives it a low-dimensional input space: compared to pattern generators based on full steps, the dimensionality is divided by 2. Our approach shares several similarities with the article [12], where Kuffner et al. present an algorithm for planning safe navigation strategies for biped robots moving in obstacle-cluttered environments. In [12], in order to reduce the number of transition trajectories between two consecutive foothstep placements, the authors introduce two intermediate postures $Q_{right}$ and $Q_{left}$ that serve as via points for all footstep transitions. $Q_{right}$ and $Q_{left}$ correspond to default postures in which the robot is balanced entirely on either the right or left foot respectively, with the other foot suspended high above the walking surface. We use the same intermediate postures as extremities for our half-steps. The main difference between our approach and the one of [12] is that at the planning phase, instead of looking for statically stable motions like in [12], we directly use the low-dimensional space of dynamic motions (the half-steps) offered by our walking pattern generator. Thanks to its low dimensionality, we can replace tests on this space by approximation functions that have been learned offline (we use the same approximation algorithm as in [14], but avoid the problems of dimensionality that it encountered). Those approximations can be evaluated in a few microseconds which makes them suitable for sampling-based planning techniques. Besides, since the domains of feasibility approximated are continuous, we have more flexibility than the approaches based on the use of the A* search algorithm with a finite set of actions (see [3], [2]), and even more than the approaches where footsteps of a finite action set can be locally adjusted ([5]). The gaits we obtain after the planning phase are sequences of half-steps that all start and finish with zero speed. Even if the half-steps are dynamic, they are still statically stable at their extremities, and as a result the gaits obtained contain strong speed variations (they frequently reach zero speed) and unnecessary sway motions of the CoM. For this reason even if those raw sequences are better than statically stable motions, they are still very poor compared to trajectories generated by state-of-the-art walking pattern generators. We show that we can cope with this issue by using a very simple homotopy which continuously deforms a raw sequence into a smoother and more dynamic sequence where the zero speed configurations have totally disappeared.

The following overview states the different components of our contribution.

B. Overview

The overall algorithm that we use for fast gait planning is described by the following steps:

1) Offline approximation using the new pattern generator proposed in this paper in order to learn domains of feasible half-steps. Contribution 1: thanks to the low dimensionality of the space of half-steps, accurate approximations can be obtained in a reasonable time (1 hour).

2) Using the RRT* algorithm recently introduced in [11], we quickly obtain a feasible (no self-collision, 2D obstacle avoidance) raw sequence of half-steps (it took about 14 seconds for the generation of a sequence of
28 half-steps). The RRT* algorithm works by growing a random tree in the configuration space, and the functions approximated help us to validate extremely quickly the edges (i.e. the half-steps) between two configurations. Contribution 2: the low dimensionality and the coherence of our approach enabled us to approximate the correct feasibility tests without any additional restriction, and thus we obtain results that are more sound than in [14], and more expressive than if the domain of feasibility was defined by an expert user, as in [4] for example.

3) Contribution 3: once a raw sequence is obtained, a simple homotopy is used to smooth it into a fluid gait. It is still compulsory to check through simulation that the smoothed sequences stay feasible, but since we use a dichotomy to set the parameters that govern our homotopy, the number of simulations needed always stays reasonable (the smoothing of 28 half-steps is done in 12 seconds).

In section II, we present the principles of our walking pattern generator based on half-steps, and introduce the operators that we use to perform the homotopy on raw sequences of half-steps. In section III, we briefly show how we practically approximated continuous domains of feasible half-steps (feasible for the robot HRP-2). Finally, in section IV, we show how we applied the recent algorithm RRT* [11] which, thanks to our approximation functions, could consider approximately approximated continuous domains of feasible half-steps trajectories for the robot HRP-2. The RRT* algorithm works by growing quickly the edges (i.e. the half-steps) between two convex hulls. Thanks to our approximation functions, could consider extremely approximate the correct feasibility tests without any additional restriction, and thus we obtain results that are more sound than in [14], and more expressive than if the domain of feasibility was defined by an expert user, as in [4] for example.

II. A WALKING PATTERN GENERATOR BASED ON HALF-STEMS

We use a classical simplified model of the robot dynamics: the Linear Inverted Pendulum Mode (see [8]). In this model the mass of the robot is assumed to be concentrated in its CoM 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. 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:

\[ p_x = x - \frac{z_c}{g} \dot{x} \]
\[ p_y = y - \frac{z_c}{g} \dot{y} \]

where \((x, y)\) 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. \((p_x, p_y)\) are the (x-axis,y-axis) coordinates of the virtual Zero Moment Point (ZMP), which is a very important point in humanoid robotics: a classical stability criterion for biped walking is that the ZMP should stay at all time inside the polygon of support (defined as the convex hull of the set of points of contact of the robot in contact with the walking surface; see [16]).

In the article [7], 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 \(\left[ x(t), y(t) \right] \) has the general following form:

\[
cosh(\sqrt{\frac{g}{z_c}} \cdot t) \left( V_x V_y \right) + \sinh(\sqrt{\frac{g}{z_c}} \cdot t) \left( W_x W_y \right) + \left( r_x(t), r_y(t) \right) \]

where \(r_x(t)\) and \(r_y(t)\) are polynomials entirely determined by \(p_x(t)\) and \(p_y(t)\) (which are also polynomials).

Using these equations, in the next section we show how to produce the lower body C-space (configuration space) trajectory corresponding to an isolated half-step. Thanks to a few assumptions on the inverse geometry of the legs, this problem can be reduced to the production of trajectories for the waist and the feet. With a few additional assumptions we can show that this C-space trajectory of the lower body is, for any half-step, entirely defined by the 7 following functions of the time:

- the CoM horizontal position: \(x(t)\), \(y(t)\) (equal to the waist horizontal position)
- the waist horizontal orientation (the yaw): \(\theta(t)\)
- the swing foot position: \(SF_x(t)\), \(SF_y(t)\), \(SF_z(t)\)
- the swing foot horizontal orientation \(SF_\theta(t)\)

A. Producing isolated half-steps

There are two types of half-steps: upward and downward. Any full step can be divided into two parts: the first one is the upward half-step where the swing foot ends up at its highest position, and the second is the downward half-step where the swing foot starts at its highest position to finish on the ground, reaching the next footprint. In this section we only consider upward half-steps, but the method for the generation of downward half-steps trajectories is similar.

Now, 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 don’t assume that the CoM initial and final speed are zero. Furthermore, the initial vertical projection on the ground of the CoM is equal to the ZMP initial position, i.e. the barycenter of the feet centers. Taking the center of the support foot as the origin of the Euclidean space, it gives
us: \( x(0) = p_x(0) = \frac{SF_y(0)}{2} \), and \( y(0) = p_y(0) = \frac{SF_y(0)}{2} \).

We also assume that the final horizontal positions of the CoM and ZMP coincide at the center of the support foot \((x(T) = p_x(T) = y(T) = p_y(T) = 0)\), and that the final orientations of the swing foot and waist are equal to the orientation of the support foot \((\theta(T) = SF_y(T) = 0)\).

Besides, the line passing through the centers of the feet final positions is orthogonal to the support foot orientation: \( SF_x(T) = 0 \).

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

\[
SF_x(0), SF_y(0), SF_y(0), SF_y(T) \text{ and } SF_x(T).
\]

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 \( p_x(t) = \frac{SF_y(t)}{2}, p_y(t) = \frac{SF_x(t)}{2}, \) and \( \dot{p}_x(t) = \dot{p}_y(t) = 0 \). 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 \( p_x(t_1) = \frac{SF_y(t_1)}{2}, p_x(t_2) = p_y(t_1) = \dot{p}_x(t_1) = \dot{p}_y(t_1) = 0, \) and \( p_y(t_1) = \frac{SF_x(t_1)}{2}, p_y(t_2) = p_x(t_2) = \dot{p}_y(t_2) = \dot{p}_x(t_2) = 0 \). 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 \( t_1 = T - t_2 \).

With the ZMP profile set, if we fix \( SF_x(0), SF_y(0), \dot{x}(0), \) and \( \dot{y}(0) \), we have a unique \( C^2 \) solution for \( x(t) \) and \( y(t) \) over \([0, T]\). The boundary conditions fix the parameters \( V_x, V_y, W_x, W_y \) (eq. (4) and eq. (5)), and eq. (3) gives us the analytical expression of the solution. Yet, the solution might violate the constraints \( x(T) = 0 \) and \( y(T) = 0 \).

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 (we used the fact that in our conditions, the functions \( x \) and \( y \) are necessarily monotone).

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 practical values obtained for \( \dot{x}(0) \) and \( \dot{y}(0) \) can be neglected, and thus the CoM trajectories obtained are supposed to be \( C^2 \) continuous over \((-\infty, \infty)\). When we tested our approach on the robot HRP-2, which executes trajectories with an additional closed-loop control system aiming at preserving the balance (see [9]), the very small discontinuities of the derivatives were cancelled by this controller. Actually, they might even be completely erased by the time discretization since they were not noticeable in the torque profiles.

For the trajectories other than \( x(T) \) and \( y(T) \) (\( \theta(t), SF_x(t), SF_y(t), SF_x(t), SF_y(t) \)), we simply use third-order polynomials that ensure \( C^2 \) continuity and satisfying profiles, with a few specific constraints (e.g. in our implementation the swing foot always leave and reach the ground 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 \( SF_x(T) \), 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 [12] 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 the 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 (not statically stable), but at the extremities of each half-step, the configuration is statically stable. This is not a satisfying result because the corresponding walking is really unsteady, unnatural, and contains unnecessary sway motions. In this section, we show how we can start from a simple concatenation of half-steps (that is to say an awkward walk), and then continuously modify it towards a much smoother and quicker sequence that will realize the same steps. We first show how to do so for a sequence of two half-steps, and then continue 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 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:

\[
g_1^{\Delta}(f)(t) = \begin{cases} f(t) & \text{for } t \in [0, T] \\ f(T) & \text{for } t \in [T, 2T - \Delta] \end{cases}
\]

\[
g_2^{\Delta}(f)(t) = \begin{cases} f(t - \Delta) - f(0) & \text{for } t \in [T - \Delta, 2T - \Delta] \\ f(t) & \text{for } t \in [0, T - \Delta] \end{cases}
\]

The two functions \( g_1^{\Delta}(y_1) \) and \( g_2^{\Delta}(y_2) \) are \( C^2 \) continuous for any \( 0 \leq \Delta \leq T \), and \( g_1^{\Delta}(y_1) + g_2^{\Delta}(y_2) \) corresponds to the simple concatenation of \( y_1 \) and \( y_2 \) without overlap. If the ZMP profiles corresponding to the CoM trajectories \( y_1 \) and \( y_2 \) are respectively \( p_{y_1} \) and \( p_{y_2} \), then starting from the equation (1) it is easy to verify that for any \( 0 \leq \Delta \leq T \),
\[
y = g_1^{\Delta}(y_1) + g_2^{\Delta}(y_2)
\]

is a solution of the differential equation:

\[
g_1^{\Delta}(p_{y_1}) + g_2^{\Delta}(p_{y_2}) = y - \frac{1}{g} \ddot{y}
\]

Therefore the 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 \( g_1^0(y_1) + g_2^0(y_2) \)) to make the second ZMP shift (the one of \( y_2 \)) 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 easily see that the position of the CoM does not need to reach the center of the support foot.

We use the same operators, \( g_1^{\Delta}_A \) and \( g_2^{\Delta}_A \), to produce an overlap between the functions of the time corresponding to the waist orientation and swing foot position and orientation.

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 \( \Delta \). Since we also use the operators \( g_1^{\Delta}_A \) and \( g_2^{\Delta}_A \) for the swing foot trajectory, a natural upper bound appears:

\[
\Delta < \min(T - t_2, t_1)
\]

(with \( \Delta > \min(T - t_2, t_1) \) the swing foot height would sometimes be negative).

After adjusting \( \Delta \) we obtain a more natural step, where the waist does not have an exaggerated sway motion. On top of that, even if we did fix the configuration of the lower body when the swing foot is at its maximum height, after the “smoothing” this special configuration will not be reached anymore (it is replaced by a flexible mixture between two configurations), so even if only 3 parameters were used to define the half-steps, combining them with overlap considerably widens the range of possible steps.

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 \( T \).

3) For longer sequences of half-steps: If we consider a sequence of three half-steps, for example upward, downward, upward, we can first apply the operators \( g_1^{\Delta}_A \) and \( g_2^{\Delta}_A \) in order to obtain an overlap between the two first half-steps. That will give us a sequence ending as a downward half-step. Thus after resetting the origin accordingly to the last half-step and adjusting the acceptable range for \( \Delta \), we can apply again the operators to produce an overlap with the third half-step. For more half-steps, we can simply repeat the procedure to smooth the whole sequence.

In the next section, we briefly present how we built approximation functions that enable us to quickly decide whether a given half-step is feasible or not. In section IV where we plan raw sequences of half-steps, we use this quick decision procedure on every half-step considered, and therefore we will only deal with raw sequences of feasible half-steps (our definition of feasible is presented in the next section).
Fig. 4. We illustrate the “smoothing” of a raw sequence of half-steps. On the initial raw sequence (on the left) we can see that the time-free ZMP (for example by generating self-collisions). Our procedure resembles the ZMP support path stays the same but the CoM support path becomes smoother (on the right, ZMP in red and CoM in blue). The distances to these two constraints are checked after 3 times faster) and more natural, because the unnecessary sway motions are reduced and the repeated speed changes does not involve self-collisions and does not violate any joint limit. The distances to these two constraints are checked after.

Thus, after adjusting the overlaps, the ZMP support path stays the same but the CoM support path becomes smoother (on the right, ZMP in red and CoM in blue). (for example by generating self-collisions). Our procedure relates to quickly find an acceptable value as large as possible.

Fig. 4 shows the results obtained with an example of a raw sequence. After the smoothing, the CoM trajectory is more fluid. The new gait obtained is much faster (about 3 times faster) and more natural, because the unnecessary sway motions are reduced and the repeated speed changes are avoided.

III. APPROXIMATING FEASIBILITY REGIONS

Since any raw half-step is, as we have seen in section II, completely defined by 3 parameters, we can simulate it offline to predict its feasibility. The feasibility of a half-step can depend on many factors; here, we simply assume that a half-step produced by our pattern generator is feasible if it

\[
g \quad \text{(s)}
\]

\[
p_t = \left( p_1 \right) + \left( p_2 \right)
\]

\[
g_1(p_t) + g_2(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[
g_3(p_t)
\]

\[

IV. USING THE RRT* ALGORITHM FOR FAST FOOTSTEP PLANNING

The low dimensionality of our walking pattern generator based on half-steps enables us to accurately learn feasibility regions. Since the approximation functions created can be evaluated extremely quickly, we can use them to consider a large number of raw sequences of only feasible half-steps. For example, let us consider a known environment with flat ground and only 2D obstacles (holes in the walking surface). When the robot receives a goal location, it first looks for an acceptable raw sequence of half-steps. Then, once a feasible raw sequence leading to the goal location while avoiding obstacles has been found, we use the technique presented in section II to transform it into a smoother gait sequence, and execute the result on the robot.

During the planning phase, we use the recent RRT* algorithm (see [11]) to grow a tree of random half-steps exploring the environment and looking for a path leading to the goal location. In our case each node in the tree corresponds to a
Fig. 5. The goal here was to escape from the small "island" in the bottom-right and join the border. The footprints of the solution found are in yellow (light yellow for the left foot, dark yellow for the right foot), with dashed arrows indicating the orientation of the foot. On an Intel Xeon 2GHz CPU, the raw sequence of 28 half-steps was found with the RRT* algorithm in about 1s (during which 356944 calls were made to our approximation functions), and then the computation time needed for the sequence smoothing was about 12s. We successfully performed the escape sequence with the robot HRP-2. This experiment is shown on the video that comes with this paper.

In this article we presented a new walking pattern generator based on half-steps. Since 3 parameters entirely define a half-step, we could easily learn offline (through simulation) the domains of feasibility of the half-steps, and then introduce the result in an implementation of RRT* which enables us to very quickly obtain a feasible concatenation of half-steps leading to the goal location. We also introduced two concatenation operators which continuously produce an overlap between consecutive half-steps. Starting from a concatenation of half-steps with zero speed at both extremities, by using these operators and checking (through simulation) that no self-collision appears, we obtain a much smoother and more satisfying trajectory. The continuity of the process brings coherence to our approach: there is no gap between the trajectories used during planning and the fully dynamic ones that are finally played on the robot. This coherence and the domains of feasibility learned offline enabled us to quickly find solutions to the problem of planning dynamic gait sequences in an environment where the robot feet can only move around in a very narrow space (Fig. 5), a problem which is difficult to solve with classical methods.

B. Future Works

Several directions can be considered for future work. First of all, we would like to use our approach in an experiment with online reactive footstep planning. To achieve this, the main problem is probably the accurate localization of the robot. We also need to speed up a bit our algorithms. We could use heuristics to speed up the RRT* search. For example, we could try to find a way to calibrate some of the parameters used by the RRT* algorithm (see [6]), or control better the sampling domain (see [17]). The smoothing could also be done progressively, while the sequence is being executed instead of beforehand.

Aside from computation time considerations, our main goal is to be able to deal with non-negative obstacles.
possible approach is to use the two parameters that we freeze for the generation of half-steps, in order to obtain flexible stepping-over motions. Another interesting approach would be to approximate the volumes swept by the half-steps, and then use them to test extremely quickly whether a given sequence will collide with the environment or not. We could also try to use, instead of fixed trajectories, motion primitives for the generation of half-steps (see for example [13] for an application of motion primitives to leg motions). Since each half-step is a relatively short motion, we might be able to learn and store very precise properties concerning those motion primitives (the low dimensionality could help to do so in reasonable time). Then, we could again consider at first sequences with no time overlap, and, once a good candidate is chosen, continuously try to increase the overlap between the motion primitives. This might enable us to go further in our quest for planning and replanning in real-time smooth gaits so that they robustly and reactively avoid obstacles, and adapt to perturbations.

VI. ACKNOWLEDGMENTS

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

REFERENCES


