Walking Patterns for Real Time Path Planning Simulation of Humanoids

Vincent Hugel[a] and Nicolas Jouandeau[b]

***Abstract*— We present here a detailed description of the walking algorithm that was designed for 3D simulation of locomotion and path planning of humanoid robots. The walking patterns described were implemented on NAO humanoid models that are used in the 3D simulation league of RoboCup to play soccer. The locomotion algorithm is based on the well known 3D-LIP model that consists of defining walking primitives of the center of mass, keeping its height constant and assuming no torque at the support foot. This paper proposes to detail how to connect the walking primitives, especially at the start of the walk. The second added value of this work resides in the rotation walking primitives that are generated differently from the linear translation walking primitives. This enables the robot to achieve fast rotation on the spot or about a center located on the longitudinal axis. The paper also addresses the issue of re-entrance, i.e. how to take into account a new walking request in real time without waiting for the end of the current walk.**

I. Introduction

RoboCup is an international project initiative that aims to foster research in humanoid and mobile robotics, but also in the area of edutainment. Real/simulated humanoid and wheeled robots participate in soccer competitions. The goal of this project is to see a team of humanoid robots play soccer efficiently against a team of humans by the year of 2050.

Simulation leagues are useful to implement new multi-agent behaviors and new locomotion algorithms. Developments in simulation leagues can be implemented on real robots and vice-versa for iterative improvements, especially between the Standard Platform League (SPL) and the Soccer Simulation League (SSL) that both use NAO robots. *rcssserver3d* is the official competition environment for the 3D Soccer Simulation League at RoboCup. It implements a soccer simulation where two teams of up to eleven humanoid robots play against each other. The official Robocup 3D simulation server is SimSpark that is a generic physical multiagent simulator system for agents in three-dimensional environments. It is based on the flexible Spark application framework. In these simulations, agents can participate in-process or out-of-process. SimSpark comes with different robot models that can play the role as agents.

Regarding locomotion first implementations were based on the parameterization of leg movements using cyclic trajectories of joint angles [1][2]. Angle trajectory parameters depend on desired longitudinal, lateral and angular velocities that can be adjusted at each walking step. Most of today algorithms use the 3D-LIP (Linear Inverted Pendulum) model [3][4]. This model was introduced by Kajita to make first Japanese humanoid robots walk efficiently [5]. Graf et al. equip their SPL team of NAO robots with locomotion skills based on the 3D-LIP model [3]. The algorithm can react in real time because it observes the center of mass (COM) and can adjust the next step to take into account a deviation of the COM from the current trajectory. Xue at al. [6] propose a variant of walking gait pattern generator by introducing a cubic polynomial of the zero moment point (ZMP) to smooth the COM trajectory. They also add a compensation to be applied to the hip roll joints that are modeled as elastic joints. The UT Austin team participates in the SPL and SSL leagues and also implements gait patterns that can be used both for the real robots and for simulation [4]. Footstep planning is at the heart of precise control. A new strategy of real time trajectory generation was proposed by Schmitz at al. that relies on offline footstep planning [7].

The walking pattern generators based on 3D-LIPM described above do not give all the details of implementation, especially regarding rotations. This paper proposes to detail how to connect the walking primitives in the 3D-LIP model at the start of the walk and how to design specific rotation primitives that are generated differently from the linear translation walking primitives. This enables the robot to achieve fast rotation on the spot or about a center located on the longitudinal axis. The paper also addresses the issue of re-entrance, i.e. how to take into account a new walking request in real time without waiting for the end of the current walk.

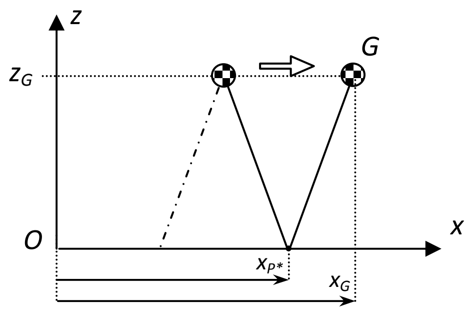

Fig. 1. 3D-LIP model. x_{P*} is the x-coordinate of real ZMP. (x_G, z_G) are the coordinates of the center of mass in the xz plane.

[a]V. Hugel is with the Engineering System Lab (LISV), Université de Versailles, 10/12 av. Europe, 78140 Vélizy - France, hugel at lisv.uvsq.fr

[b] N. Jouandeau is with the Advanced Computer Science Lab (LIASD), Université Paris 8, 2 rue de la Liberté, 93526 Saint-Denis Cedex 02, France, n at ai.univ-paris8.fr

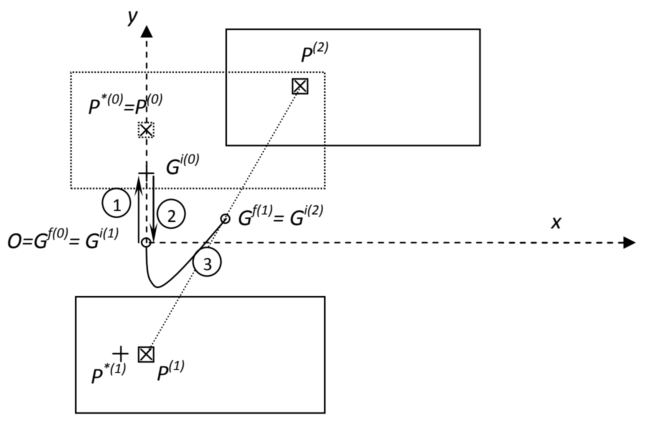

Fig. 2. First left foot step of forward-left motion. The direction of longitudinal axis x is the forward direction. Phase 1 consists of an outward sideways move of the COM to the left with zero initial and final velocities. Phase 2 is an inward sideways move to the right to reach a non zero velocity when the COM passes through the middle of the feet. This velocity is necessary to start the next move that is the first walking primitive. Phase 3 is the first walking primitive that starts the forward-left displacement to execute the first step with the left foot. The shape of phase 3 is an hyperbolic curve. In order to execute this phase the real ZMP must be behind the desired ZMP.

II. 3D-LIP MODEL

A. Presentation

When a biped robot is supporting its body on one leg its dominant dynamics can be represented by a single inverted pendulum which connects the supporting foot and the center of mass (also designated by G) of the whole robot. Figure 1 depicts such an inverted pendulum consisting of a point mass and a massless telescopic leg.

The altitude of the COM, namely z_G is kept constant, there is no torque between ground and foot, and the robot is assumed to walk on a horizontal plane. The equations that link the position of the center of mass (x_G, y_G, z_G) with the ZMP [8] of coordinates $(x_{P^*}, y_{P^*}, 0)$ are well known [5]:

$$\ddot{x}_G = \frac{g}{z_G}(x_G - x_{P^*}) \tag{1}$$

$$\ddot{y}_G = \frac{g}{z_G}(y_G - y_{P^*}) \tag{2}$$

where g is the gravity.
If we consider a constant ZMP point over each step, the analytic solution leads to a hyperbolic shape of the COM trajectory:

$$\begin{aligned} x_G(t) &= (x_G^{i(n)} - x_{P^*})\cosh(t/T_C) \\ &\quad + T_C \dot{x}_G^{i(n)} \sinh(t/T_C) + x_{P^*} \qquad (3) \\ \dot{x}_G(t) &= (x_G^{i(n)} - x_{P^*})\sinh(t/T_C)/T_C \\ &\quad + \dot{x}_G^{i(n)} \cosh(t/T_C) \qquad (4) \end{aligned}$$

where n is the step number, $x_G^{i(n)}$ and $\dot{x}_G^{i(n)}$) are respectively the COM initial x-position and the COM initial x-velocity of step n, and $T_C = \sqrt{\frac{z_G}{g}}$. The same holds for $y_G(t)$ and $\dot{y}_G(t)$.

The algorithm consists therefore of updating the COM initial position $(x_G^{i(n)}, y_G^{i(n)})$, velocity $(\dot{x}_G^{i(n)}, \dot{y}_G^{i(n)})$ and xy-coordinates of real ZMP (x_{P^*}, y_{P^*}) for each step.

The COM position and velocity at the end of step n can be calculated as:

$$\begin{bmatrix} x_G^{f(n)} \\ \dot{x}_G^{f(n)} \end{bmatrix} = \begin{bmatrix} C & T_C S \\ S/T_C & C \end{bmatrix} \begin{bmatrix} x_G^{i(n)} \\ \dot{x}_G^{i(n)} \end{bmatrix} + \begin{bmatrix} 1-C \\ -S/T_C \end{bmatrix} x_{P^*}^{(n)} \tag{5}$$

with $C = \cosh(T_{st}/T_C)$ and $S = \sinh(T_{st}/T_C)$. T_{st} is the step duration. A similar equation can be drawn to link $y_G^{f(n)}$ and $\dot{y}_G^{f(n)}$ with $y_G^{i(n)}$, $\dot{y}_G^{i(n)}$ and $y_{P^*}^{(n)}$.

The resulting trajectory is named *walking primitive*.

The parameters that can be tuned are summarized as follows:

- height of COM , z_G,
- step duration, T_{st},
- maximal step length along x-axis, ℓ_x^{max},
- maximal step length along y-axis, ℓ_y^{max},
- maximal step rotation angle, θ^{max}.

III. FORWARD AND SIDEWAYS WALKING PATTERNS

A. Initiating the walk from standstill position

To initiate the walk it is necessary to execute an outward sideways move, namely phase 1, and then to execute an inward sideways move, namely phase 2 (see example of figure 2). In phase 1, initial and final velocities are zero. The first phase is useful to transfer the robot's load to one side so that $P^{*(0)}$ can serve as center of pressure for the execution of phase 2. Phase 2 is used to enable $P^{*(1)}$ to serve as next desired center of pressure for the first walking primitive, namely phase 3.

To set up the initiation of the walk, the first step consists of calculating the lateral offset of the outward sideways move in phase 1. This offset of the COM is noted $y_G^{i(0)}$.

We denote by $P^{(n)}$ the successive *desired* ZMP points under the foot. These points belong to the support foot and they are always at the same location with respect to the support foot, for example at the ankle joint center.

It must be noted that the *desired* ZMP points $P^{(n)}$ do not necessarily match the real ZMP points $P^{*(n)}$ as in the case of acceleration and deceleration. The case of acceleration occurs at the beginning of the walk (first step). The case of deceleration occurs at the end of the walk (last step). In figure 2, it can be observed that the real ZMP point is behind the desired ZMP point due to the acceleration. In the case of deceleration the real ZMP point will be ahead of the desired ZMP point. These considerations are very important since the walking algorithm must ensure that the real ZMP point remains inside the support area of the foot, otherwise the balance of the robot will be in jeopardy.

Phase 2 can be considered as a pre-step phase that obeys equation 2. The time period of this phase, denoted by $T^{(0)}$, can be adjusted. Using eq. 5, we can write:

$$y_G^{f(0)} = C_0 y_G^{i(0)} + T_C S_0 \dot{y}_G^{i(0)} + (1 - C_0) y_{P^*}^{(0)} \quad (6)$$

$$\dot{y}_G^{f(0)} = S_0 y_G^{i(0)}/T_C + C_0 \dot{y}_G^{i(0)} - S_0 y_{P^*}^{(0)}/T_C \quad (7)$$

where $C_0 = \cosh(T^{(0)}/T_C)$, $S_0 = \sinh(T^{(0)}/T_C)$.

Since $y_G^{f(0)} = 0$ and $\dot{y}_G^{i(0)} = 0$, it comes:

$$y_G^{i(0)} = (C_0 - 1)/C_0 y_P^{(0)} \quad (8)$$

$$\dot{y}_G^{f(0)} = -(Th_0/T_C) y_P^{(0)} \quad (9)$$

with $Th_0 = S_0/C_0$, and we constrain the real ZMP to be at $P^{(0)}$, i.e. $P^{*(0)} = P^{(0)}$. $P^{(0)}$ is known. $T^{(0)}$ is the unknown.

Considering phase 3 and using eq. 5, we have:

$$y_G^{f(1)} = C y_G^{i(1)} + T_C S \dot{y}_G^{i(1)} + (1 - C) y_{P^*}^{(1)} \quad (10)$$

We can take $T_{st}^{(1)} = T_{st}$ that is the cruise walk step period. To connect the first two primitives, we must have:

$$y_G^{i(1)} = y_G^{f(0)} = 0 \quad (11)$$

$$\dot{y}_G^{i(1)} = \dot{y}_G^{f(0)} \quad (12)$$

We constrain the COM final position in phase 3 to be in the middle of $[P^{(1)}P^{(2)}]$, and we also constrain $y_{P^*}^{(1)} = y_P^{(1)}$, which is known, therefore:

$$(y_P^{(1)} + y_P^{(2)})/2 = T_C S \dot{y}_G^{f(0)} + (1 - C) y_P^{(1)} \quad (13)$$

And, using eq. 9:

$$Th_0 = -\left[(y_P^{(1)} + y_P^{(2)})/2 + (C - 1) y_P^{(1)}\right]/(S y_P^{(0)}) \quad (14)$$

This equation permits to calculate the time period of the pre-step phase (phase 2):

$$T^{(0)} = T_C \tanh^{-1}(Th_0) \quad (15)$$

and to get $y_G^{i(0)}$ through eq. 8.

Phase 1 is planned as a 5th order polynomial with minimal acceleration, i.e lateral velocity increases from zero to reach a peak and then decreases to zero.

B. Cruise walking primitives

In cruise mode, that is after the first walking primitive along the y-axis, it is necessary to determine the coordinates of the real ZMP point in order to achieve the desired trajectory of the COM.

Given the initial position and velocity of the COM, the real ZMP for the next step to be achieved permits to completely define the COM trajectory thanks to eq. 3 and the similar equation for $y_G(t)$.

The real ZMP x-coordinate is calculated by minimizing the quantity Q_x below [5], given the desired final position of the COM $(x_G^d, \dot{x}_G^d)$ (y-coordinate is managed in a similar way):

$$Q_x = \alpha(x_G^d - x_G^{f(n)})^2 + \beta(\dot{x}_G^d - \dot{x}_G^{f(n)})^2 \quad (16)$$

where α and β are positive weights. The minimizing of Q_x through $\partial Q_x/\partial x_{P^*} = 0$ gives:

$$x_{P^*} = f_1(x_G^d - C x_G^{i(n)} - T_C S \dot{x}_G^{i(n)}) + f_2(\dot{x}_G^d - \frac{S}{T_C} x_G^{i(n)} - C \dot{x}_G^{i(n)}) \quad (17)$$

with $f_1 = -\frac{\alpha(C-1)}{D}$, $f_1 = -\frac{\beta S}{T_C D}$, and $D = \alpha(C-1)^2 + \beta(S/T_C)^2$.

In our application we use the model of the NAO humanoid [9] (height about $0.57[m]$, weight around $4.5[kg]$, and 22 degrees of freedom). Parameters α and β were respectively fixed to 80 and 0.8. These values were determined experimentally to minimize the differences between $P^{(n)}$ and $P^{*(n)}$.

Final desired position of the COM (x_G^d, y_G^d) is calculated from the previous desired position of the ZMP using the current values of the longitudinal and lateral steps. The longitudinal step length ℓ_x and lateral step length ℓ_y are calculated as follows:

$$\ell_x = d_{fwd}/nb\_steps \quad (18)$$

$$\ell_y = 2\, d_{sdw}/nb\_steps \quad (19)$$

where d_{fwd} and d_{sdw} are the forward distance and the sideways distance to travel respectively. nb_steps is the number of steps. A step is defined by the number of times legs are lift off. There is no double support phase. Each time a leg touches ground the other leg takes off. To walk sideways the robot needs to make two steps to advance length ℓ_y: the first foot moves ℓ_y, and the second foot moves also ℓ_y to be parallel with the first foot, but the COM only travels ℓ_y. Hence the factor 2 in the above formula for ℓ_y. The number of steps is calculated as follows:

```
1: procedure $calculate\_nb\_steps(d_{fwd}, d_{sdw}, \ell_x^{max}, \ell_y^{max})$
2:     $\epsilon \leftarrow 0.001$
3:     $nb\_steps_x \leftarrow \lfloor |d_{fwd}|/\ell_x^{max} \rfloor$
4:     $remaining\_d_{fwd} \leftarrow |d_{fwd}| - nb\_steps_x \ell_x^{max}$
5:     if $remaining\_d_{fwd} > \epsilon$ then
6:         $nb\_steps_x \leftarrow nb\_steps_x + 1$
7:     end if
8:     if $|d_{sdw}| > \epsilon$ then
9:         if $nb\_steps_x mod\ 2 = 1$ then
10:            $nb\_steps_x \leftarrow nb\_steps_x + 1$
11:        end if
12:    end if
13:    $nb\_steps_y \leftarrow \lfloor |d_{sdw}|/\ell_y^{max} \rfloor$
14:    $remaining\_d_{sdw} \leftarrow |d_{sdw}| - nb\_steps_y \ell_y^{max}$
15:    $nb\_steps_y \leftarrow 2 nb\_steps_y$
16:    if $remaining\_d_{sdw} > \epsilon$ then
17:        $nb\_steps_y \leftarrow nb\_steps_y + 2$
18:    end if
19:    if $nb\_steps_x > nb\_steps_y$ then
20:        $nb\_steps \leftarrow nb\_steps_x$
21:    else
22:        $nb\_steps \leftarrow nb\_steps_y$
23:    end if
```

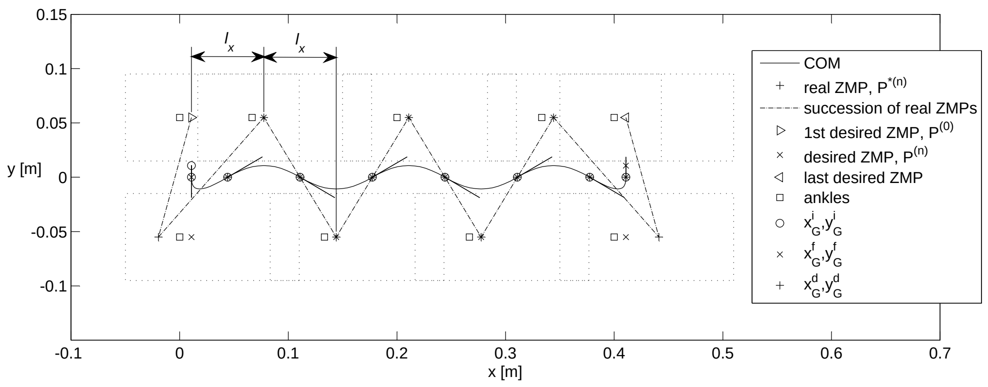

Fig. 3. Forward walk of $0.4[m]$ from left to right. ℓ_x is the step length. Footprints are in dotted lines. The COM trajectory is depicted. Successive real ZMPs are joined for better understanding.

```
24:        return $nb\_steps$
25:    end procedure
```

The instruction in line 9 is necessary to have even number of steps for sideways moves.

The COM final desired velocities are calculated using step lengths, namely ℓ_x and ℓ_y, and step time, namely T_{st}.

C. Connection between walking primitives

To enable connection between walking primitives the COM final position in the current primitive becomes the initial position of the COM in the next primitive. This ensures continuity of position and velocity of the COM. When the robot has to halt, the end of the walk is managed in a similar way, that is a next-to-last walking primitive for the COM to reach the final x-position and a last primitive along the y-axis to set the velocity to zero. Then a sideways move is needed to bring the COM to the middle of the feet. This motion is done by interpolation with zero initial and final velocities.

Figure 3 shows the successive steps executed during a forward walk of $0.4[m]$. It also depicts the COM trajectory and the successive desired and real ZMPs.

IV. TURN-IN-PLACE WALK

Rotation about a point situated on the longitudinal axis was managed differently. Real ZMP points are calculated given additional constraints of symmetry of the motion. The two additional constraints are the following:

1) The velocity of the COM at the beginning of the walking primitive n must be parallel with the segment joining desired ZMP points $P^{(n-1)}$ and $P^{(n)}$ where $P^{(n)}$ is the current ZMP point. So the velocity at the end of the walking primitive n must be parallel with the segment joining desired ZMP points $P^{(n)}$ and $P^{(n+1)}$.
2) The final position of the COM must be on the bisector of segment $[P^{(n)}P^{(n+1)}]$ passing through the center of rotation C.

Figure 4 shows the COM trajectory for an anticlockwise turn about C of $40[deg]$.

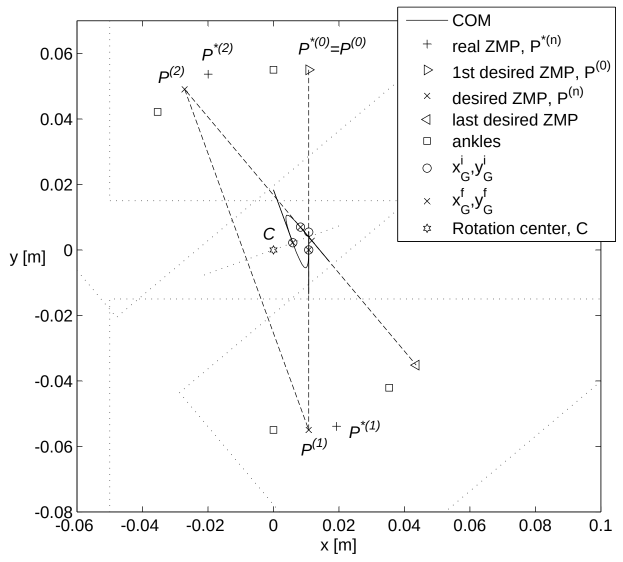

Fig. 4. Anticlockwise turn-in-place of $40[deg]$ in one step. Rotation takes place about C. Feet are initially parallel with the longitudinal axis. Desired ZMPs are joined to show that the COM follows the same successive directions.

A. Initiating the rotation

Like in the translation mode described previously the motion starts with an outward sideways move – phase 1 –, followed by an inward sideways move – phase 2 –. Phase 3 is the first walking primitive that contributes to the rotation of the COM. The problem here consists of calculating the coordinates of real ZMP $P^{*(1)}$ for the first walking primitive, taking into account both constraints enumerated above.

The first constraint can be expressed as:

$$\left(\begin{bmatrix} \dot{x}_G^{f(1)} \\ \dot{y}_G^{f(1)} \\ 0 \end{bmatrix} \times \begin{bmatrix} x_{P21} \\ y_{P21} \\ 0 \end{bmatrix} \right) . \boldsymbol{z} = 0$$

$$\Leftrightarrow \dot{x}_G^{f(1)} y_{P21} - \dot{y}_G^{f(1)} x_{P21} = 0 \tag{20}$$

where $x_{P21} = x_P^{(2)} - x_P^{(1)}$ and $y_{P21} = y_P^{(2)} - y_P^{(1)}$.

The second constraint can be expressed as:

$$\begin{bmatrix} x_G^{f(1)} - x_C \\ y_G^{f(1)} - y_C \end{bmatrix} \cdot \begin{bmatrix} x_{P21} \\ y_{P21} \end{bmatrix} = 0$$

$$\Leftrightarrow (x_G^{f(1)} - x_C)x_{P21} + (y_G^{f(1)} - y_C)y_{P21} = 0 \tag{21}$$

Replacing $\dot{x}_G^f$ and $\dot{y}_G^f$ using eq. 5 in eqs 20 and 21 yields:

$$\begin{cases} a\ x_P^{(1)} + b\ y_P^{(1)} + d\ \dot{y}_G^{i(1)} + d = 0 \\ e\ x_P^{(1)} + f\ y_P^{(1)} + g\ \dot{y}_G^{i(1)} + h = 0 \end{cases} \tag{22}$$

which can be written as:

$$\begin{bmatrix} a & b \\ e & f \end{bmatrix} \begin{bmatrix} x_{P*}^{(1)} \\ y_{P*}^{(1)} \end{bmatrix} = - \begin{bmatrix} c \\ g \end{bmatrix} \dot{y}_G^{i(1)} - \begin{bmatrix} d \\ h \end{bmatrix} \tag{23}$$

with $a = y_{P21}$, $b = x_{P21}S/T_C$, $c = -x_{P21}C$, $d = y_{P21}Sx_G^{i(0)}/T_C$, $e = (1-C)x_{P21}$, $f = (1-C)y_{P21}$, $g = T_C S y_{p12}$, and $h = Cx_G^{i(0)}x_{P21} - x_C x_{P21} - y_C y_{P21}$, knowing that $x_G^{i(1)} = x_G^{i(0)}$, $\dot{x}_G^{i(1)} = 0$, and $y_G^{i(1)} = 0$.

The solution for the equation system 23 is:

$$\begin{cases} x_{P*}^{(1)} = u\ \dot{y}_G^{i(1)} + v \\ y_{P*}^{(1)} = r\ \dot{y}_G^{i(1)} + s \end{cases} \tag{24}$$

with $u = (-c\ f + b\ g)/\Delta$, $v = (-d\ f + b\ h)/\Delta$, $r = (-a/g + e\ c)/\Delta$, $s = (-a\ h + d\ e)/\Delta$, and $\Delta = a\ f - b\ e = -S(1-C)(P^{(1)}P^{(2)})^2/T_C$.

Here $\dot{y}_G^{i(1)}$ can be chosen so that $P^{*(1)}$ is closest to $P^{(1)}$. This comes to minimize the quantity:

$$J = (x_{P*}^{(1)} - x_P^{(1)})^2 + (y_{P*}^{(1)} - y_P^{(1)})^2 \tag{25}$$

Solving for $\partial J/\partial \dot{y}_G^{i(1)} = 0$ gives:

$$\dot{y}_G^{i(1)} = -\frac{u(v - x_P^{(1)}) + r(s - y_P^{(1)})}{u^2 + r^2} \tag{26}$$

Coordinates of $P^{*(1)}$ can then be calculated thanks to eq. 24.

B. *Cruise mode*

$P^{*(2)}$ is calculated as the symmetric point of $P^{*(1)}$ with respect to the bisector of segment joining $P^{(1)}$ and $P^{(2)}$. $P^{*(3)}$ is calculated as the rotated point of $P^{*(2)}$ after rotation over a step.

Therefore the successive real ZMP can be calculated as follows:

- $P^{*(2k)}$ is the symmetric point of $P^{*(2k-1)}$, for $k \in \{1 \ldots n\}$, with respect to bisector $[P^{(2k)}P^{(2k-1)}]$.
- $P^{*(2k+1)}$ is the rotated point of $P^{*(2k)}$, for $k \in \{1 \ldots n\}$, after step rotation.

Figure 5 shows the successive steps with desired and real ZMPs for an anticlockwise rotation of $90[deg]$.

V. Reentrance

A walking request in linear translation needs desired displacements in longitudinal and lateral directions, and desired step time as inputs. This enables left/right forward/backward moves. A walking request in rotation needs desired angular displacement, the coordinate of the rotation center on the longitudinal axis, and desired step time as inputs. This enables clockwise and counterclockwise rotations about points located on the longitudinal axis.

Depending on the nature of the current walk – translation or rotation –, two kinds of reentrance are considered. The first kind occurs when the new walk request is of the same nature, i.e. current translation to another translation or rotation to another rotation on the spot. The second kind of request occurs when the new walk request is of different nature than the current walk. This appears with a change from a translation to a rotation and vice-versa.

In case of reentrance of linear translation, the new request is taken into account at the next step, i.e at the next walking primitive. The final position and velocity of the COM in the current walking primitive serve as initial inputs to the new walking primitive. The new ZMP is calculated taking into account the desired position and velocity of the COM at the end of the new walking primitive. These desired values are calculated thanks to the new desired step length and the new step time derived from the walk request arguments. The distance that remains to be traveled is adjusted according to the distance covered between the time of the request and the time at the end of the current walking primitive. The step length is calculated according to the distance to be traveled, taking into account a maximal step length. When it comes to change direction, from pure forward to pure backward and conversely, the leg that is standing back, respectively standing forth, at the end of the current primitive must execute a step on the spot so that the other leg can execute a first step in the opposite direction. In case of pure left to right motions and conversely, one step on the spot is only needed when the leg opposite to the new direction of motion has to move first at the end of the current walking primitive.

Regarding reentrance of pure rotation, the new walking primitive associated with the new request of rotation is only triggered when the feet are parallel again, e.g. when the right leg – resp. left leg – turns left – resp. right – in the air, this is to avoid to halt rotation motion with no parallel legs. Motion halt with parallel legs enables to start new patterns either rotation or translation from the beginning. One consequence of this choice is that the new rotation pattern may be delayed by the additional time of one walking primitive necessary to place the follower leg parallel to the leader leg. Equation 24 is used to calculate the new real ZMP points that are useful to define the COM trajectory of the new walking primitive. To change direction it is necessary for the follower leg relative to the new rotation direction to make a step on the spot to enable the leader leg to start to move in the opposite direction.

The connection between a translation and a rotation mo-

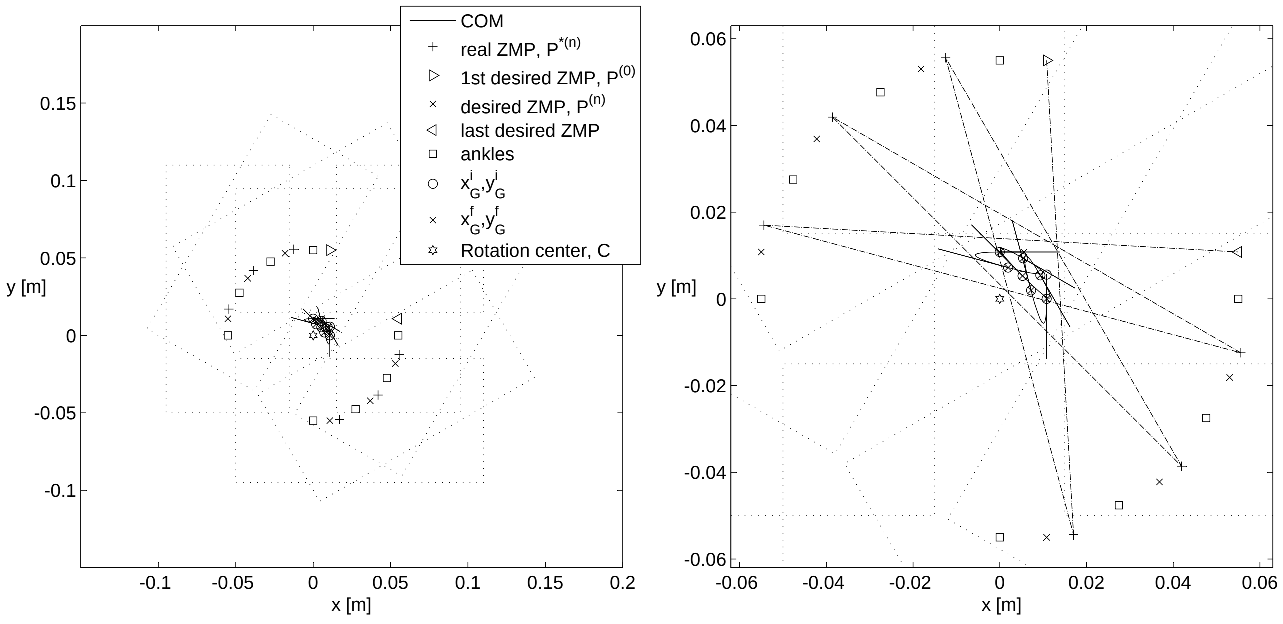

Fig. 5. Anticlockwise turn-in-place of $90[deg]$ in two steps. Feet are initially parallel with the longitudinal axis. The right-hand side view is a zoom of the left-hand side view. On the right-hand side successive real ZMPs are joined.

tion is made easy insofar as the new walking pattern is triggered when feet become parallel. If the current walk is a translation motion it is stopped to have feet parallel before starting the rotation, and conversely. Halting the motion consists of executing a specific terminal walking primitive that is different from the cruise walking primitive, then to execute an outward sideways move followed by an inward sideways move to bring the COM between the feet to reach the standing position.

VI. Inverse kinematics

The origin of the coordinate reference frame where COM and ZMP points are calculated is the point on the ground between both ankle centers at the beginning of the walking pattern.

The COM is considered to be fixed with respect to the torso. This is not actually true but this approximation is enough for our application. The longitudinal offset between COM and torso point is set to $0.11[m]$, this is to have more margin for the real ZMP point between the ankle and the heel of the foot.

Inverse kinematics to get joint angles from COM trajectory is implemented thanks to the inverse geometric model (IGM). The convention used is the Khalil and Kleinfinger convention [10] that is useful to get successive transformation matrices from one joint coordinate frame to the other [11] given a specific set of parameters. The coordinates of the knee center are calculated first to get knee angle, then ankle joint angles and finally hip joint angles. Knee angle is obtained by calculating the square distance of the segment joining the hip center to the ankle center. Ankle joint angles are obtained by solving for the 3d transformation matrix at the ankle. Hip joint angles are obtained by solving for the 3d transformation matrix at the hip.

VII. Simulations

The robot used for simulations is the NAO robot from Aldebaran-Robotics [9]. The parameters of the NAO robot are [12]:

- HipOffsetY $= 0.05[m]$
- HipOffsetZ $= 0.085[m]$
- FootHeight $= 0.045[m]$
- length of femur $Lf = 0.1[m]$
- length of tibia $Lt = 0.103[m]$

The walking parameters are:

- $\ell_x^{max} = 0.07[m]$
- $\ell_y^{max} = 0.07[m]$
- $\theta^{max} = 40[deg]$
- $T_s = 0.02[sec]$
- $T_{st} = 0.24[sec]$ for translation, and $T_{st} = 0.18[sec]$ for rotation
- $z_G = 0.3[m]$

These patterns were used with success in the 3D simulation league of the RoboCup Dutch Open that took place in Eindhoven, Netherlands at the end of April 2012 (Fig. 6). The patterns were used to move the robots on the field to the ball and to push it to the opponent goal. The walk was fairly stable but motion speed was lower than that of the best three teams. The maximal average forward velocity was $0.22[m.s^{-1}]$. The maximal average angular velocity was $1.0[rad.s^{-1}]$. Velocities can be increased be decreasing the time step and setting sampling time for commands update to $10[ms]$. Additional velocity gain can be obtained by the use of learning procedures, e.g. genetic algorithms, to tune walking pattern parameters through the minimizing of a cost function.

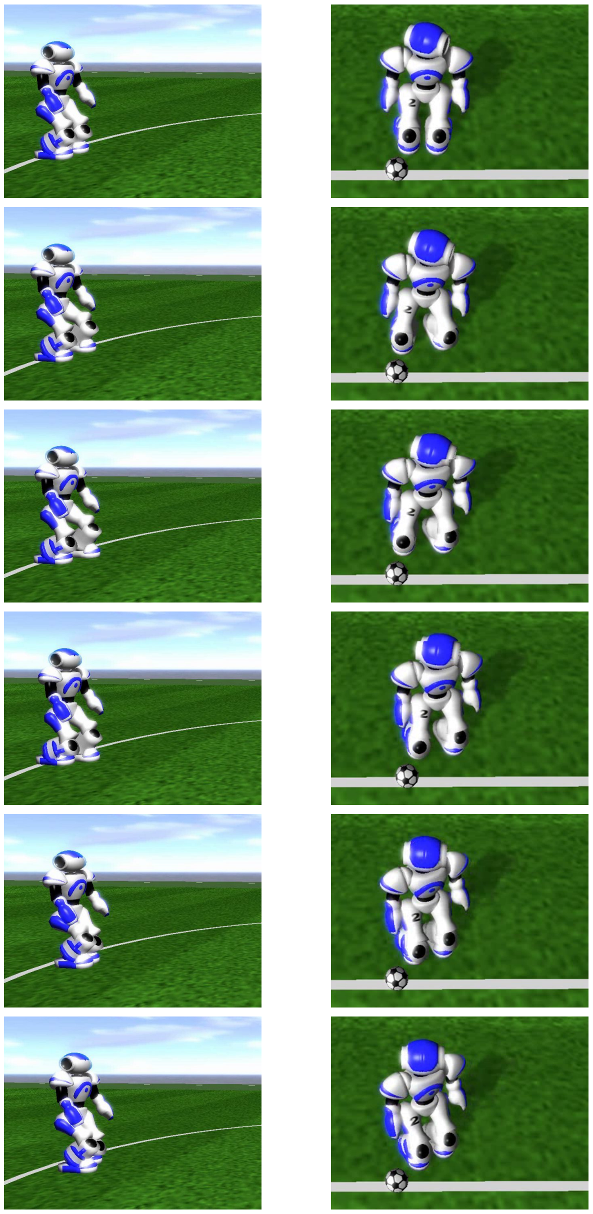
Fig. 6. 1st column: forward walk in the simulator. 2nd column: rotation on the spot.

VIII. Conclusion on Possible extensions

This paper has presented walking pattern details used in simulation for making humanoids move in all directions, and adapt their path to identified targets. The patterns were used with success in the 3D simulation league by triggering walking requests in real time to go to the ball, push it, and score goals. Motion speed was not the highest among teams. Nevertheless it is possible to improve the current locomotion with the following extensions:

- In the current walk, step length is constant, but it can be useful to decrease it at the beginning and the end of walk, this is to avoid too high acceleration and deceleration at the beginning and at the end. More generally it is possible to vary step length during the walk, this can be useful to adapt velocity to the desired behavior.
- Currently translation patterns do not accept any rotation component. But it is possible to add one. However this rotation component must be limited to avoid the ZMP to go outside the foot support polygon. In addition, the rotation component cannot be introduced into every walking primitive, this is to prevent the robot from stepping on its feet. As a matter of fact, the introduction of a left rotation while going forward must only start to be active when the left foot is lift off, and vice-versa.
- The actual connection between walking primitives of different nature, e.g. rotation and translation, is made by passing though the halt position where both feet are rigorously parallel next to each other. However it is possible to save time by connecting walking primitives directly without double sideways move required by the halt posture. This demands a specific design of connections, which must monitor the robot's stability margin before actually triggering the transition between both walking patterns.
- The walking patterns developed so far are open-loop. The addition of sensor feedback will be useful to monitor balance, and to improve it to prevent falls. The first step will be to detect loss of balance accurately, if possible thanks to the estimation of ZMP offset in both horizontal directions. Reactions to loss of balance can be implemented with reflex walking primitives or with reflex moves if walking primitives are not enough.

References

[1] S. Behnke, Online trajectory generation for omnidirectional biped walking, in: Int. Conf. on Robotics and Automation, 2006, pp. 1597–1603.

[2] C. Graf, T. Röefer, A closed-loop 3d-lipm gait for the robocup standard platform league humanoid, in: Fourth Workshop on Humanoid Soccer Robots in conjunction with the IEEE-RAS International Conference on Humanoid Robots, 2010.

[3] C. Graf, T. Röefer, A center of mass observing 3D-LIPM gait for the robocup standard platform league humanoid, in: RoboCup 2011: Robot Soccer World Cup XV, Lecture Notes in Artificial Intelligence, Vol. 7416, Springer, Heidelberg, 2012, pp. 101–112.

[4] P. MacAlpine, D. Urieli, S. Barrett, S. Kalyanakrishnan, F. Barrera, A. Lopez-Mobilia, N. Stiurca, V. Vu, P. Stone, UT Austin Villa 2011, 3D Simulation Team Report.

[5] S. Kajita, Humanoid Robot, Ohmsha Ltd, 3-1 Kanda Nishikicho, Chiyodaku, Tokyo, Japan, 2005.

[6] F. Xue, X. Chen, J. Liu, D. Nardi, Real time biped walking gait pattern generator for a real robot, in: RoboCup Int. Symposium, 2011.

[7] A. Schmitz, M. Missura, S. Behnke, Real-time trajectory generation by online footstep planning for a humanoid soccer robot, in: RoboCup Int. Symposium, 2011.

[8] M. Vukobratovic, B. Borovac, Zero-moment point - thirty five years of its life., in: Int. J. of Humanoid Robotics, Vol. 1(1), 2004, pp. 157–173.

[9] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, B. Maisonnier, Mechatronic design of NAO humanoid, in: Int. Conf. on Robotics and Automation, 2009, pp. 769–774.

[10] W. Khalil, J.-F. Kleinfinger, A new geometric notation for open and closed-loop robots, in: Int. Conf. on Robotics and Automation, 1986, pp. 1174–1180.

[11] V. Hugel, R. Hackert, A. Abourachid, Kinematic modeling of bird locomotion from experimental data, in: IEEE Transactions on Robotics, Vol. 27(2), 2011, pp. 185–200.

[12] Http://simspark.sourceforge.net/wiki/index.php/Models.