Omnidirectional Humanoid Walking using Model Predictive Control

Marcos Maximo
Computer Science Division
Aeronautics Institute of Technology
Praça Marechal Eduardo Gomes, 50
Vila das Acácias, 12228-900
São José dos Campos, SP, Brazil
Email: mmaximo@ita.br

Rubens Afonso
Electronic Engineering Division
Aeronautics Institute of Technology
Praça Marechal Eduardo Gomes, 50
Vila das Acácias, 12228-900
São José dos Campos, SP, Brazil
Email: rubensjm@ita.br

Carlos Ribeiro
Computer Science Division
Aeronautics Institute of Technology
Praça Marechal Eduardo Gomes, 50
Vila das Acácias, 12228-900
São José dos Campos, SP, Brazil
Email: carlos@ita.br

***Abstract*—Humanoid walking is considered one of the hardest problems in Robotics. Current state-of-the-art humanoid robots are able to achieve high speeds on flat ground. However, they still exhibit agility, dexterity, robustness, flexibility and energy efficiency far below than a typical human does. In this work, we present an omnidirectional walking pattern generator for a humanoid robot. We follow an approach based on the Zero Moment Point (ZMP) concept, which provides an useful criterion for biped stability. To avoid dealing directly with the complex dynamics of a high degrees of freedom humanoid robot, we used the 3D Linear Inverted Pendulum Model (3D-LIPM) to approximate the robot dynamics. The resulting dynamics is linear, which allows us to use a linear Model Predictive Control (MPC) scheme. Finally, experiments results are shown to validate the method presented.**

1. Introduction

Humanoid robots are claimed to be more adequate than wheeled ones for locomotion in unstructured human environments that include floor discontinuities, ladders, rocks, debris, etc. However, walk control for humanoid robots (biped locomotion) involves characteristics acknowledged for making a control problem hard, such as high nonlinearity, underactuation, and many degrees of freedom. Current state-of-the-art humanoid robots are able to achieve high speeds on flat ground. Nevertheless, they are still outmatched by the typical human in terms of robustness and energy efficiency [1]. Therefore, humanoid walking is still considered an unsolved problem in Robotics.

Biped locomotion methods can be grouped into two main approaches: model-based and model-free. Model-based methods depend on analytic models of the dynamics of the robot. Due to the high dimensionality of a high degrees of freedom humanoid robot, approximate models are often used, e.g. "3D Linear Inverted Pendulum" [2]. Then, a controller is developed for the model assuming that the dynamics of the robot follows the model. On the other hand, model-free methods try to produce a controller directly, without a mathematical model for the dynamics of the robot. Two known model-free methods are the Central Pattern Generator (CPG) [3] and the Ballistic Walking [4].

A popular concept in the humanoid walking Literature is the Zero Moment Point (ZMP) [5]. The ZMP is often called the dynamic version of the center of mass (CoM) due to the following stability criterion: a biped is dynamically stable at a given time instant if the ZMP is inside the support polygon (the convex hull of all contact points on the ground). Therefore, a common approach is to generate a gait that maintains the ZMP inside the support polygon at all times.

In this work, we present an omnidirectional walking pattern generator for a humanoid robot. Our approach is based on the ZMP concept and uses the 3D-LIPM to approximate the robot dynamics. Then, a Model Predictive Control (MPC) formulation inspired by the work presented in [6] is used. Using MPC proved very useful due to the possibility to explicitly consider constraints in the gait generation process [7]. Moreover, footsteps positions are added as decision variables to the scheme, which allow the robot to select footsteps positions that will ensure its stability. We expect to use this walking algorithm in the humanoid robots of team ITAndroids, which competes in RoboCup's 3D Soccer Simulation and Humanoid Kid-Size leagues, so the feasibility of real-time execution is of concern.

2. 3D Linear Inverted Pendulum Model (3D-LIPM)

The 3D Linear Inverted Pendulum Model (3DLIPM) approximates the robot dynamics by considering that the whole robot mass is concentrated in its CoM. Additionally, Kajita et al. demonstrated that if the pendulum mass is constrained to move along a plane, then the resulting dynamics is linear. Further considering this constraint plane to be a horizontal plane, we arrive at the following uncoupled linear differential equations for the ZMP dynamics of the inverted pendulum:

$$z_x = x - \frac{h}{g}\ddot{x} \tag{1}$$

$$z_y = y - \frac{h}{g}\ddot{y} \tag{2}$$

where $[z_x, z_y]^T$ is the ZMP position, $[x, y]^T$ is the CoM position, h is the CoM height, and g is the acceleration of gravity. These equations allow us to predict the ZMP trajectory given a CoM trajectory, thus they play a major role in our walking engine.

3. Gait Generation using Model Predictive Control

Model Predictive Control (MPC) techniques solve an optimization problem at each time considering predicted future system behavior to yield an optimal control sequence [7]. Our MPC formulation closely follows the one presented by Wieber et al. in [6] and has the following characteristics: linear model, discrete time implementation, quadratic cost function, use of receding horizon, and constraints handling.

3.1. Predicting the Future

For conciseness, the derivation presented here considers only x direction, however expanding it to the y direction is straightforward. We consider that we are able to directly influence the jerk (i.e. the derivative of acceleration) $\dddot{x}$ of the robot's center of mass. Therefore, the CoM evolution is bound by the following kinematic relations:

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\dddot{x} \tag{3}$$

Our MPC approach requires discrete time implementation, thus we may discretize these continuous time equations without approximating by considering that we hold the jerk command $\dddot{x}$ within a time step of length T:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\dddot{x}(k) \tag{4}$$

where:

$$\mathbf{x}(k) = \begin{bmatrix} x(k) \\ \dot{x}(k) \\ \ddot{x}(k) \end{bmatrix} \tag{5}$$

$$\mathbf{A} = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} T^3/6 \\ T^2/2 \\ T \end{bmatrix} \tag{6}$$

Hence, Equations (1) and (2) become:

$$z_x(k) = \mathbf{C}\mathbf{x}(\mathrm{k}) \tag{7}$$

where:

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & -h/g \end{bmatrix} \tag{8}$$

Finally, we may apply Equation (4) recursively to predict future system behavior over a prediction horizon of N time steps:

$$\mathbf{X}(k) = \begin{bmatrix} x(k+1|k) \\ \vdots \\ x(k+N|k) \end{bmatrix} = \mathbf{P}_{ps}\mathbf{x}(k) + \mathbf{P}_{pu}\dddot{\mathbf{X}}(k) \tag{9}$$

$$\dot{\mathbf{X}}(k) = \begin{bmatrix} \dot{x}(k+1|k) \\ \vdots \\ \dot{x}(k+N|k) \end{bmatrix} = \mathbf{P}_{vs}\mathbf{x}(k) + \mathbf{P}_{vu}\dddot{\mathbf{X}}(k) \tag{10}$$

$$\mathbf{Z}_x(k) = \begin{bmatrix} z_x(k+1|k) \\ \vdots \\ z_x(k+N|k) \end{bmatrix} = \mathbf{P}_{zs}\mathbf{x}(k) + \mathbf{P}_{zu}\dddot{\mathbf{X}}(k) \tag{11}$$

where:

$$\dddot{\mathbf{X}}(k) = \begin{bmatrix} \dddot{x}(k|k) \\ \vdots \\ \dddot{x}(k+N-1|k) \end{bmatrix} \tag{12}$$

3.2. Cost Function

As an optimal control approach, MPC requires a cost function which will be optimized by the controller. Wieber et al. consider a cost function that compromises between speed tracking and walking stability. One of the major advantages of using Wieber's approach is that it allows the controller to decide where to place the footsteps, instead of fixing them beforehand. An effective strategy humans use for balancing when faced with strong perturbations is to change footstep placement [8]. Therefore, leaving the decision of footsteps positions for the robot will allow it to automatically perform this stepping strategy if needed. To allow automatic footstep placement, let us first include the footsteps positions as decision variables:

$$\mathbf{u}(k) = \begin{bmatrix} \dddot{\mathbf{X}}(k) \\ \mathbf{X}_f(k) \\ \dddot{\mathbf{Y}}(k) \\ \mathbf{Y}_f(k) \end{bmatrix} \tag{13}$$

where $\mathbf{X}_f(k)$ and $\mathbf{Y}_f(k)$ contain the x and y coordinates of N_f footsteps positions to be decided, respectively. Then, the following cost function is considered:

$$\begin{aligned} J(\mathbf{u}(k)) = \frac{\alpha}{2}\left|\left|\dddot{\mathbf{X}}(k)\right|\right|^2 + \frac{\beta}{2}\left|\left|\dot{\mathbf{X}}(k) - \dot{\mathbf{X}}_r(k)\right|\right|^2 \\ + \frac{\gamma}{2}\left|\left|\mathbf{Z}_x(k) - \mathbf{Z}_{x,r}(k)\right|\right|^2 + \frac{\alpha}{2}\left|\left|\dddot{\mathbf{Y}}(k)\right|\right|^2 \\ + \frac{\beta}{2}\left|\left|\dot{\mathbf{X}}(k) - \dot{\mathbf{X}}_r(k)\right|\right|^2 + \frac{\gamma}{2}\left|\left|\mathbf{Z}_y(k) - \mathbf{Z}_{y,r}(k)\right|\right|^2 \end{aligned} \tag{14}$$

where are α, β, and γ are positive weights. Moreover, ZMP reference is placed always at the center of the support polygon:

$$\mathbf{Z}_{x,r} = \mathbf{U}_c(k)X_{fc}(k) + \mathbf{U}(k)\mathbf{X}_f(k) \tag{15}$$

$$\mathbf{Z}_{y,r} = \mathbf{U}_c(k)Y_{fc}(k) + \mathbf{U}(k)\mathbf{Y}_f(k) \tag{16}$$

where the matrices $\mathbf{U}_c(k)$ and $\mathbf{U}(k)$ are used to select which foot is the active one at each time step. By employing previous equations, the cost function may be rewritten using only the decision variables (constants terms are dropped here since they do no influence the optimization):

$$J(\mathbf{u}(k)) = \frac{1}{2}\mathbf{u}^T(k)\mathbf{H}(k)\mathbf{u}(k) + \mathbf{f}^T(k)\mathbf{u}(k) \tag{17}$$

where:

$$\mathbf{H}(k) = \begin{bmatrix} \mathbf{H}'(k) & \mathbf{0} \\ \mathbf{0} & \mathbf{H}'(k) \end{bmatrix} \tag{18}$$

$$\mathbf{H}'(k) = \begin{bmatrix} \alpha\mathbf{I} + \beta\mathbf{P}_{vu}^T\mathbf{P}_{vu} + \gamma\mathbf{P}_{zu}^T\mathbf{P}_{zu} & -\gamma\mathbf{P}_{zu}^T\mathbf{U}(k) \\ -\gamma\mathbf{U}^T(k)\mathbf{P}_{zu} & \gamma\mathbf{U}^T(k)\mathbf{U}(k) \end{bmatrix} \tag{19}$$

$$\mathbf{f}(k) = \begin{bmatrix} \beta\mathbf{P}_{vu}^T\left(\mathbf{P}_{vs}\mathbf{x} - \dot{\mathbf{X}}_r\right) + \gamma\mathbf{P}_{zu}^T\left(\mathbf{P}_{zs}^T\mathbf{x} - \mathbf{U}_c X_{fc}\right) \\ -\gamma\mathbf{U}^T\left(\mathbf{P}_{zs}\mathbf{x} - \mathbf{U}_c X_{fc}\right) \\ \beta\mathbf{P}_{vu}^T\left(\mathbf{P}_{vs}\mathbf{y} - \dot{\mathbf{Y}}_r\right) + \gamma\mathbf{P}_{zu}^T\left(\mathbf{P}_{zs}^T\mathbf{y} - \mathbf{U}_c Y_{fc}\right) \\ -\gamma\mathbf{U}^T\left(\mathbf{P}_{zs}\mathbf{y} - \mathbf{U}_c Y_{fc}\right) \end{bmatrix} \tag{20}$$

Note that the dependence on k was dropped for some terms in Equation (20) for conciseness.

3.3. Constraints

MPC allows directly reasing about constraints [7]. As stated in [5], an effective strategy to ensure walking stability is to have the ZMP inside the support polygon at all times. Hence, this constraint is explicitly considered:

$$\begin{bmatrix} \mathbf{d}_x(\theta_i) & \mathbf{d}_y(\theta_i) \end{bmatrix} \begin{bmatrix} z_x(k+i|k) - z_{x,r}(k+i) \\ z_y(k+i|k) - z_{y,r}(k+i) \end{bmatrix} \leq \mathbf{s}, \quad 1 \leq i \leq N \tag{21}$$

where $[\mathbf{d}_x(\theta) \;\; \mathbf{d}_y(\theta)]$ stacks the vectors that are normal to the edges of the support polygon, and $\mathbf{s}$ the maximum ZMP error allowed by the support polygon in each direction. In our case, we model the robot's foot as rectangular and use:

$$\mathbf{d}_x(\theta) = \begin{bmatrix} \cos\theta & -\cos\theta & -\sin\theta & \sin\theta \end{bmatrix}^T \tag{22}$$

$$\mathbf{d}_y(\theta) = \begin{bmatrix} \sin\theta & -\sin\theta & \cos\theta & -\cos\theta \end{bmatrix}^T \tag{23}$$

$$\mathbf{s} = \begin{bmatrix} l/2 & l/2 & w/2 & w/2 \end{bmatrix} \tag{24}$$

where l and w are the length and width of the foot, respectively.

Walking speed is ultimately limited by maximum joint speed. Since we have simplified the humanoid dynamics by its CoM dynamics, our control scheme is unable to directly reason about joint saturation. Nevertheless, Wieber et al. argue that joint speed limit may be adequately captured by bounding the position of the next foot step depending on the current position of the swing foot and how much time is left before the swing foot touches the ground:

$$\begin{bmatrix} \mathbf{d}_x(\theta_c) & \mathbf{d}_y(\theta_c) \end{bmatrix} \begin{bmatrix} x_{f,1} - x_a \\ y_{f,1} - y_a \end{bmatrix} \leq t_{touch} \begin{bmatrix} v_{x,max} \\ v_{x,max} \\ v_{y,max} \\ v_{y,max} \end{bmatrix} \tag{25}$$

where $[x_{f,1}, y_{f,1}]^T$ is the first footstep position, $[x_a, y_a]^T$ is the current position of the swing foot, and $v_{x,max}$ and $v_{y,max}$ are the maximum speeds in x and y directions, respectively.

To avoid feet collision, we add a last group of constraints to ensure that the feet always stay at a minimum lateral distance of each other:

$$(-1)^b \begin{bmatrix} -\sin\theta_c & \cos\theta_c \end{bmatrix} \begin{bmatrix} x_{f,1} - x_c \\ y_{f,1} - y_c \end{bmatrix} \leq -y_{sep} \tag{26}$$

$$(-1)^{j+b} \begin{bmatrix} -\sin\theta_j & \cos\theta_j \end{bmatrix} \begin{bmatrix} x_{f,j+1} - x_{f,j} \\ y_{f,j+1} - y_{f,j} \end{bmatrix} \leq -y_{sep}, \quad 1 \leq j \leq N_f - 1 \tag{27}$$

where:

$$b = \begin{cases} 1, & \text{left foot is the swing foot} \\ 0, & \text{otherwise} \end{cases} \tag{28}$$

All the constraints described here are linear, thus they may be grouped as simply:

$$\mathbf{A}_{ineq}\mathbf{u}(k) \leq \mathbf{b}_{ineq} \tag{29}$$

3.4. Computing the Optimal Control

The optimal control is the solution to the following quadratic program (QP) [9]:

$$\begin{aligned} \mathbf{u}^*(k) = \arg\min_{\mathbf{u}(k)} & \frac{1}{2}\mathbf{u}^T(k)\mathbf{H}(k)\mathbf{u}(k) + \mathbf{f}^T(k)\mathbf{u}(k) \\ & \text{subjected to } \mathbf{A}_{ineq}\mathbf{u}(k) \leq \mathbf{b}_{ineq} \end{aligned} \tag{30}$$

whose solution may be efficiently found by state-of-the-art QP solvers, such as CPLEX [10].

4. Gait Generation Experiments

To verify the gait generation approach described in Section 3, we implemented the underlying algorithm in MATLAB. We used CPLEX [10] as our QP solver, which is one of the fastest solvers available for QP. We considered the following parameters: $T = 0.1s$, $N = 30$, $h = 0.24m$, $y_{sep} = 0.1m$, $v_{x,max} = 1m/s$, $v_{y,max} = 1m/s$, $l = 0.06m$, and $w = 0.03m$. Moreover, the step duration is $T_{step} = 0.5s$ and the double support ratio is $f_{ds} = 0.2$. Thus, the robot needs to stay exactly one time step in double support, which frees us from having to model the support polygon when the robot is in double support.

Then, we asked the algorithm to generate four different gait patterns, which differ by the commanded speed $\mathbf{v} = \begin{bmatrix} \dot{x}_r, \dot{y}_r, \dot{\theta}_r \end{bmatrix}^T$ and by the cost functions weights α, β, and γ. The results of these experiments are shown in figures 1, 2, 3, and 4. Since we want good speed tracking, we used $\alpha = 10^{-6}$ and $\beta = 1$ in all experiments. Moreover, we used $\gamma = 10^{-6}$ for experiments 1, 3, and 4. In experiment 2, we used $\gamma = 10^3$ to evaluate how this would impact the compromise between speed tracking and stability.

In experiment 1, we requested the following velocity: $\dot{x}_r = 0.2m/s$, $\dot{y}_r = 0m/s$, and $\dot{\theta}_r = 0rad/s$. In the resulting gait shown in Figure 1, it is noteworthy how the algorithm maintains the ZMP inside the support polygon

at all times. Furthermore, the solution found by the optimization exploits the constraints to their limits by keeping the ZMP at the edges of the support polygon most of the time, which helps to reduce jerk usage and deviation from the requested speed to a minimum. Since $\dot{y}_r = 0m/s$, the algorithm tries to minimize the natural torso sway in y direction. Additionally, the minimum separation of y_{sep} between the feet is satisfied.

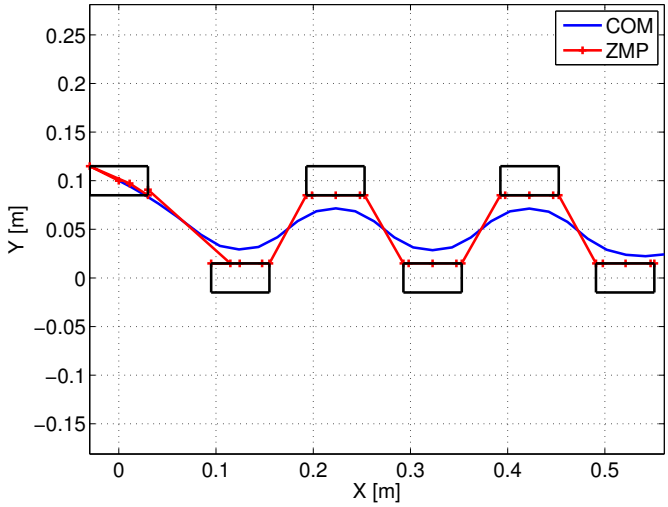

Figure 1. Gait generation experiment 1: $\dot{x}_r = 0.2m/s$, $\dot{y}_r = 0m/s$, $\dot{\theta}_r = 0rad/s$, $\alpha = 10^{-6}$, $\beta = 1$, and $\gamma = 10^{-6}$.

In experiment 2, we request the same velocity command than experiment 1, but the weight for ZMP error is increased to $\gamma = 10^3$. Hence, we expect the resulting ZMP trajectory to be kept close to the center of the feet and this is what happens, as shown in Figure 2.

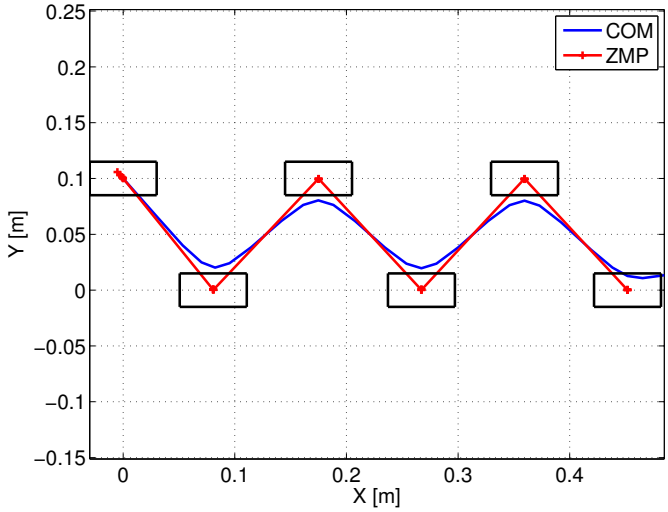

Figure 2. Gait generation experiment 2: $\dot{x}_r = 0.2m/s$, $\dot{y}_r = 0m/s$, $\dot{\theta}_r = 0rad/s$, $\alpha = 10^{-6}$, $\beta = 1$, and $\gamma = 10^3$.

In experiment 3, the robot is requested to move forward and sideways simultaneously with speeds $\dot{x}_r = 0.1m/s$ and $\dot{y}_r = -0.2m/s$. As may be seen in Figure 3, the constraints are satisfied, however the speed in y direction is slower than expected since we are regulating instantaneous speed. This problem may easily solved by regulating a mean speed instead of instantaneous speed, as discussed in [6].

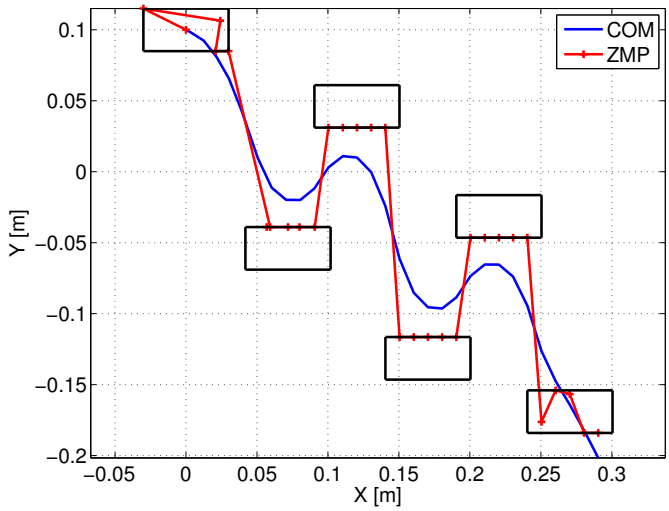

Figure 3. Gait generation experiment 3: $\dot{x}_r = 0.1m/s$, $\dot{y}_r = -0.2m/s$, $\dot{\theta}_r = 0rad/s$, $\alpha = 10^{-6}$, $\beta = 1$, and $\gamma = 10^{-6}$.

Experiment 4 involves forward and rotation movements simultaneously and its result is presented in Figure 4. Again, the ZMP constraints are exploited to their limits to yield maximum performance, as the ZMP is kept in the edges of the feet most of the time. Note that the rotation of the feet is not a result of the MPC scheme and needed to be predefined by a higher-level layer.

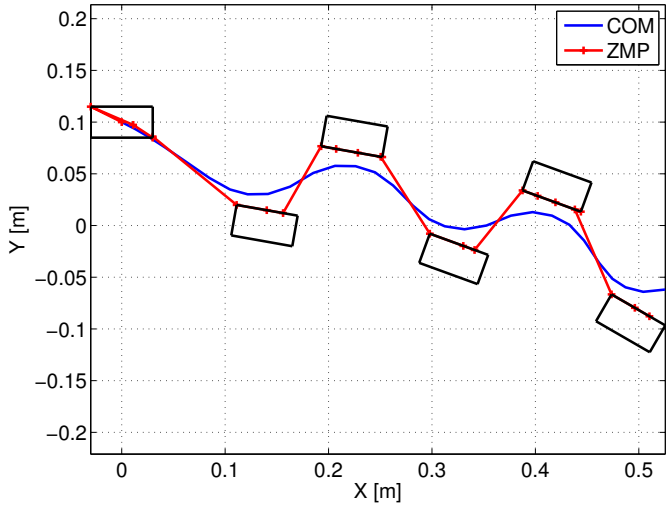

Figure 4. Gait generation experiment 4: $\dot{x}_r = 0.2m/s$, $\dot{y}_r = 0m/s$, $\dot{\theta}_r = 10\pi/180rad/s$, $\alpha = 10^{-6}$, $\beta = 1$, and $\gamma = 10^{-6}$.

Despite being a powerful control technique, MPC schemes are often too computationally expensive for real-time control, especially when fast dynamics are involved.

TABLE 1. SETUP TIME STATISTICS USING 1000 SAMPLES FOR EACH GAIT GENERATION EXPERIMENT.

Experiment	Mean (ms)	Standard Deviation (ms)
1	7.1	1.1
2	7.1	1.7
3	6	1.3
4	6	1.3

TABLE 2. SOLVER TIME STATISTICS USING 1000 SAMPLES FOR EACH GAIT GENERATION EXPERIMENT.

Experiment	Mean (ms)	Standard Deviation (ms)
1	14.4	1.3
2	7.6	0.9
3	16.2	1.5
4	28.1	2.1

Therefore, we decided to measure how much time the algorithm takes to generate a gait pattern in each experiment case. Note that some of the matrices shown in Section 3 are not static and depend on information only present at time step k, so some matrix manipulation is required at each time step for assembling the final optimization matrices, which requires what we called "setup time". Then, we called "solver time" the time required to solve the QP itself.

Statistics based on 1000 samples for setup time and solver time are shown in Table 1 and Table 2, respectively. These tests were run in a computer with a Core i5-3320M CPU @ 2.60 GHz processor with 4 GB of RAM. An interesting fact is that setup time is almost constant between experiments, while solver time differs greatly. Finally, note that if the gait generation is called at each time step of duration $T = 0.1s$, then the algorithm looks suited for real-time usage. However, limited processing power embedded in small humanoid robots may pose a problem, especially if this processing power must be shared with other tasks: computer vision, decision making etc.

5. Conclusion

This work presented an omnidirectional walking engine for a humanoid robot. This walking algorithm is based on the Zero Moment Point (ZMP) concept, since it provides a useful criterion for stability. Furthermore, the "3D Linear Inverted Pendulum" (3D-LIPM) model is used since it provides a simple linear dynamics for the robot dynamics while being precise enough for walking pattern generation. Then, the algorithm attempts to generate a gait where the ZMP is maintained inside the support polygon at all time instants by using a Model Predictive Control (MPC) scheme that explicitly considers the ZMP criterion as constraints in the associated optimization problem. This MPC formulation is inspired by the one presented in [6].

By tuning the weights used in the cost function, the designer may chose the desired compromise between control effort (jerk) usage, reference speed tracking and stability. Beyond constraints to keep the ZMP inside the support polygon, additional constraints are considered to avoid feet collision and exceeding maximum joint speed. Moreover, footsteps positions are considered as decision variables in the underlying optimization problem, allowing the robot to autonomously select footsteps positions in order to keep balance in case of strong perturbations.

The gait generation algorithm was implemented in MATLAB and the results show that interesting characteristics arise, such as the robot automatically exploiting the ZMP constraints to their limits by keeping the ZMP close to the edges of the feet. Despite MPC being known for its high computational cost, measurements of the processing time indicate that the scheme may be able to run in real-time.

For future research, we expect to test this walking algorithm in realistic simulations and in real humanoid robots. Furthermore, we are looking for ways to expand the MPC formulation described here. One idea is to allow the robot to also select footstep duration, which may be possible using mixed integer programming approaches.

Acknowledgment

Marcos Maximo is a member of team ITAndroids and acknowledges the team's support. Moreover, he thanks the company Radix for sponsoring ITAndroids.

References

[1] S. Collins, A. Ruina, R. Tedrake, and M. Wisse, "Efficient Bipedal Robots Based on Passive-Dynamic Walkers," *Science*, vol. 307, no. 5712, pp. 1082–1085, February 2005.

[2] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3D Linear Inverted Pendulum Mode: a Simple Modeling for a Biped Walking Pattern Generator," in *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001, pp. 239–246.

[3] F. Hackenberger, "Balancing Central Pattern Generator based Humanoid Robot Gait using Reinforcement Learning," Master's thesis, Theoretical Computer Science, 2007.

[4] S. Mochon and T. McMahon, "Ballistic Walking," *J. Biomech.*, pp. 49–57, 1980.

[5] M. Vukobratović and B. Borovac, "Zero-moment point—thirty five years of its life," *International Journal of Humanoid Robotics*, vol. 1, no. 01, pp. 157–173, 2004.

[6] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online walking motion generation with automatic foot step placement," *Advanced Robotics*, vol. 24, no. 5–6, 2010.

[7] J. Maciejowski, *Predictive Control with Constraints*. Prentice Hall, 2000.

[8] S.-J. Yi, B.-T. Zhang, D. Hong, and D. D. Lee, "Online learning of a full body push recovery controller for omnidirectional walking," in *Proceedings of the 11th IEEE-RAS International Conference on Humanoid Robots*, 2011.

[9] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. Academic Press, 1981.

[10] IBM, "IBM ILOG CPLEX Optimization Studio – CPLEX User's Manual," http://www-01.ibm.com/support/knowledgecenter/SSSA5P_12.6.1/ilog.odms.studio.help/pdf/usrcplex.pdf.