Vision-guided motion primitives for humanoid reactive walking: decoupled vs. coupled approaches

Mauricio Garcia[1,2] and Olivier Stasse[2] and Jean-Bernard Hayet[1]
Claire Dune[3] and Claudia Esteves[4] and Jean-Paul Laumond[2]

***Abstract*— This paper proposes a novel visual servoing approach to control the dynamic walk of a humanoid robot. Online visual information is given by an on-board camera. It is used to drive the robot towards a specific goal. Our work is built upon a recent reactive pattern generator that make use of Model Predictive Control (MPC) to modify footsteps, center of mass and center of pressure trajectories to track a reference velocity. The contribution of the paper is to formulate the MPC problem considering visual feedback. We compare our approach with a scheme decoupling visual servoing and walking gait generation. Such a decoupled scheme consists in first, computing a reference velocity from visual servoing; then, the reference velocity is the input of the pattern generator. Our MPC based approach allows to avoid a number of limitations that appears in decoupled methods. In particular visual constraints can be introduced directly inside the locomotion controller, while camera motions do not have to be accounted for separately. Both approaches are compared numerically and validated in simulation. Our MPC method shows a faster convergence.**

I. Introduction

Humanoid robots are meant to function in unstructured and dynamic environments, where objects may move outside the robots control. Therefore, to complete specific tasks, robots have to be able to perceive and react to environment changes. Visual sensors endow robots with this capability, allowing them to build local representations of their surroundings and to adapt their behavior according to these representations. Most of the existing humanoid platforms are equipped with video cameras. They constitute rich (geometry, texture, color, etc.) sources of information at rather low costs both in terms of price and additions to the weight and size of the robot. Furthermore, cameras embedded in the platform avoid the need of outfitting the environment with external sensors, greatly increasing the autonomy of the robot.

However, the image quality from video cameras embedded in humanoid robots is, in general, quite poor: blurring effects or vibrations due to the walk make interpretation of these images for visual tasks such as localization and tracking really challenging. This situation has prompted considerable effort in the computer vision comunity, with no as-of-yet generally accepted solution.

In this paper, we build upon recent advances in walking pattern generation (WPG) and visual servoing to construct a new monocular visual servoing scheme that is able to control the walk of a humanoid robotic platform towards an object of interest. An example setup can be seen in Fig. 1 where the robot has to walk towards the cupboard with the image of the cupboard as a goal reference. We use the information from the sensors as feedback in certain tasks, e.g. positioning. We find that this visual servoing control scheme is a very good candidate for closing the perception-decision-action loop, due in particular, to its robustness to model errors.

Moreover, as we show below, the use of a visual servoing control fits well in its traditional role as a "black box" of the WPG.

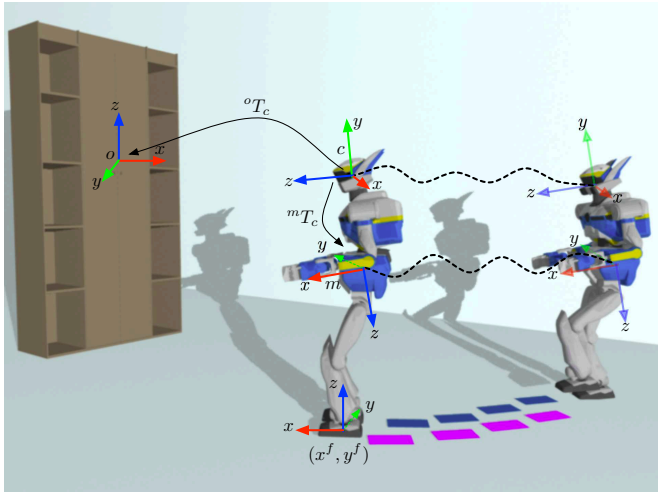

Fig. 1. An example setup for our approach: the robot has to walk towards a desired position with regard to an object viewpoint. Frames o, c and m refer to the *object*, *camera* and *CoM* respectively. ^oT_c is the transformation to view the camera points in the object reference frame. (x^f, y^f) is the current footprint position.

The main feature of our approach is the direct use of visual errors within the WPG, i.e. a visual predictive control where visual servoing and WPG, the generation of footsteps and center of mass trajectories, are completely coupled. We compare our approach to a recently-proposed decoupled approach, where visual servoing takes its traditional role as a provider for reference velocity for the camera, whose output

[1] Mauricio Garcia and Jean-Bernard Hayet are with CIMAT, A. C., Jalisco S/N Mineral de Valenciana, CP 36240 Guanajuato, Mexico {mjgarciav,jbhayet}@cimat.mx.
[2] Mauricio Garcia, Olivier Stasse and Jean-Paul Laumond are with LAAS-CNRS, 7 av. du Colonel Roche, F-31400, Toulouse, France, {mjgarcia,ostasse,jpl}@laas.fr.
[3] Claire Dune is with Université du Sud Toulon Var.
[4] Claudia Esteves is with Universidad de Guanajuato, Guanajuato, Mexico, cesteves@cimat.mx.

is then used as a reference velocity for the WPG. Our main claim is that, in contrast to other methods available in the literature, we do not need to model the particular motion of the camera induced by the robot walk, thereby gaining in robustness and flexibility. Also, our approach allows to fully take into account the robot constraints as well as visual constraints in a unified framework. Finally, our experimental results show that our approach achieves a better overall behavior of the whole control.

A. Visual servoing vs. (re-)planning

Two paradigms for vision-based navigation have been co-existing for a long time in mobile robotics. The first one tackles navigation more globally using a three-step sequence: (1) vision-based mapping and localization; (2) trajectory planning; and (3) control. With this approach, a robot can navigate towards a visual goal if this three-step sequence can be cycled on-board of the robot at a high frequency. The second paradigm deals with the visual navigation problem in a more reactive manner, using mostly visual servoing techniques. Here, planning is generally avoided and the visual perception output feeds directly the control. We believe that these two paradigms are in fact, more complementary than opposite: whereas planning may require higher computational costs than visual servoing, it can solve more complex problems of global path finding. Visual servoing is fast by nature but in general it is confined to solving local tasks.
Visual servoing is a useful approach for controlling precisely the robot position in certain tasks such as in human-robot interaction, e.g. to set the robot in front of a person, or to place the robot's end-effector in a given posture before starting a manipulation task. The advantage of visual servoing is that the positioning task is defined relatively to a specific target. Even more, the precise positioning may not be necessary. In any of the aforementioned applications, the higher-level navigation can be left to a planner, which would determine the sequence of landmarks, or human interactions, to reach consecutively.
However, as many advantages the second paradigm provides to the visual navigation of humanoid robots, it also generates some problems, which have to do precisely with the fact that the output of the vision system, serves as the input to the control of the robot WPG. Some of these are that: (1) the output of the vision system may introduce noise to the WPG of the robot and degrade its behavior; (2) as a control process driven by visual errors and designed for an exponential decay of these errors, the classical visual servoing approach also generates control laws exhibiting this exponential decay for the velocity controls. While the latter may be desirable for a wheeled robot, it may not be adequate to a humanoid dynamic walk. In this work we try to deal with both of these problems. Our claim on the first point is that the visual data can be easily filtered to avoid the propagation of perturbations to the walking controller. On the second point, even though we start by recalling a classic visual servoing scheme where the gain is not adapted, adaptive gains exist in the literature of this problem. Furthermore, in the approach we propose here, this problem is highly alleviated because the error terms are now mixed with other terms (e.g. regularizing the jerks) into the pattern generator. We will discuss more on this point in Section VI-D.

B. Contributions

Building upon the WPG initially proposed by Herdt et al [Herdt et al., 2010a], [Herdt et al., 2010b], we propose an integration of a visual servoing scheme within the WPG through linear MPC. We adapt the non-linear global model for visual servoing using MPC described by Allibert et al in [Allibert et al., 2010], by means of a linearization of the projection function. Here, we use the position-based version of [Allibert et al., 2010], that requires 3-D information of the object of interest, therefore, localization is needed. We claim that a simple, local linearization of the projection model predicts well the behavior of the system and that the performance of this linearization depends on the distance traveled in the time horizon, which depends itself on the velocity of the robot and the size of the time horizon. By coupling this linearization to a simplified dynamical model of the locomotion, we can introduce predictive visual control to the WPG in a straightforward way. To our knowledge of the existing literature, this is the first time that such a strategy is used right at the pattern generation level. We finally compare our approach both, qualitatively and quantitatively to the one proposed recently in [Dune et al., 2010].

C. Paper Overview

First, in Section II, some work related to our approach will be discussed. In Section III, we will recall the principles of the reactive WPG with automatic footstep placement, and in Section IV, we make a brief reminder about existing visual servoing schemes. Then, in Section V, we describe the approach proposed in [Dune et al., 2010], that uses the visual servoing output velocities as an input to the pattern generation. In Section VI, we present a new scheme that integrates visual servoing within the pattern generator. In Section VII, we present simulation results and comparisons between the decoupled and the coupled approaches.

First, a few guidelines about the notations used throughout this paper should be mentioned: Bold, capital letters are used to refer to matrices; normal, lower-case letters refer to scalars or vectors, whereas normal, capital letters are used for the stacks of quantities in a time window, which is typically used in MPC. As for the indices, we mainly use indices k or j to refer to time indices, whereas other letters are used as indices and exponents to indicate 3D frames in transformations or twist matrices. For example, ${}^{c}\mathbf{T}_{m}$ is a 3D rigid transform mapping points expressed in the "m" frame (the center of mass frame) to their coordinates in the "c" frame (the camera frame).

D. Notations

MPC Notations

k, j		Time indices.
q_k		Full robot state at time k.
u_k		Control variable at time k.
q_k $f(q_{k-1}, u_{k-1})$	$=$	Dynamic model of the robot.
$\mathbf{Q}_k$		Quadratic term of the cost function considered in this paper.
p_k		Linear term of the cost function considered in this paper.
U_k		Vector of control variables. Free variables of the MPC-based optimization problem solved in this paper.

WPG Notations

$x_k, \dot{x}_k, \ddot{x}_k$	Respectively the CoM position, velocity and acceleration in the x-axis at time k.
$y_k, \dot{y}_k, \ddot{y}_k$	Respectively the CoM position, velocity and acceleration in the y-axis at time k.
x_k^f	Position of the support foot at time k.
θ^f	Orientation of the support foot.
ξ_k^x	Center-of-Pressure (CoP) along the x-axis at time k.
$\hat{x}_k$	Vector stacking the CoM position, velocity and acceleration along the x-axis.
X_k	Sequence of CoM x-positions for a preview window of size N starting at k.
$\dddot{X}_k$	Sequence of CoM x-jerks for a preview window of size N starting at k.
X_k^f	Sequence of support foot x-positions for a preview window of size N starting at k.
Z_{k+1}^x	Sequence of CoP x-positions for a preview window of size N starting at k.
Z_{k+1}^{ref}	Sequence of reference CoM positions along the x-axis for a preview window of size N.
$\alpha_T, \beta_T, \gamma_T$	Weighting parameters for the translational component optimization.
$\alpha_R, \beta_R, \gamma_R$	Weighting parameters for the rotational component optimization.
$\mathbf{A}, \mathbf{B}, \mathbf{C}$	Matrices relating respectively $\hat{x}_k$ with $\hat{x}_{k+1}$, $\dddot{x}_k$ with $\hat{x}_{k+1}$, $\hat{x}_k$ with ξ_k^k, when considering the Linear Inverted Pendulum Model.
$(a_1^x(\theta^f), a_1^y(\theta^f))^T$	Vectors normal to the support foot edges.
$b_1(\theta^f)$	Position of the support foot edges along the normal defined by $a_1^x(\theta^f), a_1^y(\theta^f)$.

Visual Servoing Notations

λ	Visual features indices.
l	Landmarks indices (one landmark gives two visual features, its u and v coordinates).
$u(x, y, z)$	Projection of point (x, y, z) on the x-axis image plane.
$v(x, y, z)$	Projection of point (x, y, z) on the y-axis image plane.
v^c	Real camera velocity.
$\overline{\mathrm{v}}^c$	Ideal camera velocity.
${}^c\mathbf{T}_m$	Transform matrix relating the camera frame (index "c") to the CoM frame (index "m").
${}^c\mathbf{V}_m$	Twist matrix related to ${}^c\mathbf{T}_m$.
$s_{\lambda,k}$	Observation of feature λ at time k.
s_k^*	Reference feature vector at time k.
s_k^d	Desired feature vector at time k.
s_k^m	Predicted feature vector at time k.
$s_k^m = h(q_k)$	h is the observation model.
$S_{\lambda,k}^m$	Collection of the predicted positions for feature λ in the horizon starting at time k.
p_l^o	Position of the l'-th landmark in the object of interest reference frame (index o).
e	Visual servoing task.
$\mathbf{L}_e$	Interaction matrix of task e.
$\hat{\mathbf{L}}_e$	Approximation of the interaction matrix of task e.
$\hat{\mathbf{L}}_e^+$	Moore-Penrose pseudo-inverse of an approximation $\hat{\mathbf{L}}_e$.
$\mathbf{W}_j$	Weighting matrix j in the horizon.
$\prod$	Projection matrix.

II. State of the Art

Early works on humanoid locomotion have assumed that the robot path on the ground is completely defined before computing the actual joint control to realize it. This clearly puts limits on the capacities of reaction when changes occur in the environment. These works generally follow a perception-decision-action scheme, in the sense that a sensor first acquires data on the world and/or the robot state, then, suitable footsteps over a time horizon are decided, and finally the trajectories of the Center of Mass (CoM) and the Center of Pressure (CoP) are computed while respecting the stability constraints and avoiding collision with the environment. Finally, the control of the legs and other joints is computed by inverse kinematics. This perception-decision-action loop has proven to be fast enough to realize impressive demonstrations for stair-climbing and obstacle avoidance [Lorch et al., 2002], [Chestnutt et al., 2007], [Michel et al., 2007], [Gutmann et al., 2008]. Our focus in this paper will be set only on one of the sub-problems necessary to implement this approach: the generation of footsteps and trajectories of the CoM and CoP. We stress that we will not address here the whole body control.

In the generation of the footsteps and the trajectory of the CoM or CoP, for a long time most of the works did not consider their online modification from an initial plan. The work presented in [Morisawa et al., 2007] considered online adaptation after showing that the modification of the next landing position of the flying foot might impose a new CoP trajectory going out of the support polygon and jeopardize the equilibrium of the robot. To solve the problem, the stepping period may be modified to reduce this instability [Morisawa et al., 2007], at the cost of slowing down the robot. A recent method proposes to modify the footsteps according to a perturbation applied to the CoP [Nishiwaki and Kagami, 2009].

Later on, very efficient and much more flexible control systems for humanoid robots walking generation have been proposed. They are dynamically stable and may be very reactive since footstep placement can be computed online [Herdt et al., 2010a]. Moreover, they differ from the aforementioned works by the fact that the CoP is authorized to move freely inside the support polygon, leaving much more flexibility to the motion. Most of these techniques such as in [Herdt et al., 2010a] are based on linear MPC. MPC previews the behavior of the system within a time window in the future by applying a given virtual sequence of controls. In most approaches, the controls are encoded as the supposedly constant third derivatives (jerks) of the CoM position during single time intervals. MPC allows to estimate the optimal control sequence at some horizon, even if, in the next iteration, one simply applies the first control of the computed optimal control sequence, and starts again in a similar way for the next one. At the end, MPC can be expressed as a Quadratic Program (QP), i.e., the minimization of quadratic errors subject to a set of linear constraints (equalities and

inequalities). Handling explicitly the constraints in the QP is one of the main advantages of the MPC. Furthermore there are very efficient techniques proposed to solve such a QP. The most important point is that, in [Herdt et al., 2010a], the only required input, besides the characteristics of the robot, is a reference velocity for the CoM. Hence, the MPC can be seen as a "black box" taking as an input a reference velocity to follow, and generating the corresponding CoM trajectory.

For reactive positioning tasks, visual servoing techniques have proven to be useful [Chaumette and Hutchinson, 2006], [Chaumette and Hutchinson, 2007]. In the humanoids community, visual servoing has been successful for grasping tasks while standing [Coelho et al., 2001], [Taylor and Kleeman, 2001] or for walking humanoids [Courty et al., 2001], [Mansard et al., 2007]. In [Courty et al., 2001], visual servoing has been used to control a humanoid avatar navigating along landmarks. The upper body is approximated by the kinematic chain that links an on-board camera to the CoM. The lower body is controlled by adding two translational degrees of freedom to the CoM. The translational velocity of the CoM is sent to a kinematic locomotion module which controls the legs motion. In [Mansard et al., 2007], a whole body visual servoing scheme based on a hierarchical stack of task is introduced. However, the footsteps are defined beforehand. The leg motion is set to be the task of higher priority, and visual servoing in this context is projected in the null-space of the pre-defined walking path.

More recently, in [Dune et al., 2010], which is the method we will compare ours to, the reference velocity of the CoM is computed based on a visual servoing based controller, associated to the "black box" pattern generator of [Herdt et al., 2010a]. The reference velocity computed from the visual servoing is used by the MPC to adapt the footsteps and the CoM trajectories, while ensuring at the same time walking stability constraints at time intervals of constant length. The main disadvantage of this approach is that the visual servoing scheme and the pattern generator are completely decoupled. This means that the visual information is not directly feeding the pattern generator. Instead, it simply gives a "trend" velocity to follow, and no visual constraints can be introduced inside the MPC problem. Moreover, it needs to handle the sway motion by virtually removing the motion of the camera due to the sway, based on the control inputs and outputs, which we do not. Another downside of this method is that the reference velocity introduced in the pattern generator is only taken as a reference, i.e. it is not *exactly* tracked.

Here, we adopt a similar approach as the ones mentioned above, in the sense that the footsteps are changed dynamically as in [Herdt et al., 2010a]. However, the main difference with the aforementioned work is that we do not introduce explicitly the reference velocity computed elsewhere by the visual servoing. Instead, we let the pattern generator decide for it by itself, within the optimization scheme.

One of the main motivations in this work is to tighten the link between the WPG and the visual servoing control. Basically, we aim at integrating more directly the visual servoing errors within the MPC. Outside the humanoid robots literature, Allibert et al. have successfully applied visual servoing within a MPC scheme [Allibert et al., 2010]. The main problem in doing so is that the dynamics of the camera and the involved perspective projection are nonlinear functions so that non-linear programming would be required to be combined with the aforementioned QP problem. Hence, a direct application of a MPC visual servoing scheme for the WPG would be time consuming and not adapted to an online walking pattern generator.

III. MPC-BASED WALKING PATTERN GENERATION

Most of the current walking motion generation schemes for humanoid robots are based on the model initially proposed in [Kajita et al., 2003], which focuses on determining the trajectory of the CoM to generate balanced and stable motions. The underlying dynamic model operating for each direction is the one of a cart-table system, that corresponds well to the distribution of mass in the humanoid robot. We will not be exhaustive here in describing the whole dynamic system, but let us revise its main components.

A. Dynamical system

First, we suppose that the CoM has periodic piece-wise constant jerks (third derivatives) on each time interval of duration T. This is because the accelerations of the CoM have to be continuous enough to be done without damage to the robot actuators. Let us discretize the time into these intervals of length T, and let us use the index k to refer to one such interval. Then, by writing (1) that the position at $k+1$ results from the position at k and the integration of the constant jerks, and (2) that (by definition of the CoP) the sum of moments at the CoP cancels, we can express the CoM dynamics on the x-axis as a *linear* system

$$\left\{ \begin{array}{rcl} \hat{x}_{k+1} & = & \mathbf{A}\hat{x}_k + \mathbf{B}\dddot{x}_k \\ \xi_k^x & = & \mathbf{C}\hat{x}_k, \end{array} \right. \tag{1}$$

where $\hat{x}_k \stackrel{\text{def}}{=} (x_k, \dot{x}_k, \ddot{x}_k)^\top$ stacks the $x-$position, $x-$velocity and $x-$acceleration of the robot CoM at time k into a 3×1 vector, and where the three matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ are given by

$$\mathbf{A} \stackrel{\text{def}}{=} \begin{pmatrix} 1 & T & \frac{1}{2}T^2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{pmatrix}, \mathbf{B} \stackrel{\text{def}}{=} \begin{pmatrix} T^3/6 \\ T^2/2 \\ T \end{pmatrix},$$

and

$$\mathbf{C} \stackrel{\text{def}}{=} \begin{pmatrix} 1 & 0 & \frac{c_z}{g} \end{pmatrix}.$$

In these equations, ξ_k^x is the CoP x-position at time k and c_z is the CoM height, which is considered as a *constant* during all the following. Similar expressions can be determined for the y and z components.

Then, starting from the time index k, we can apply recursively the dynamics of Eq. 1 N times, and express a full trajectory of the CoM in a horizon of N time intervals, in function of the initial position, the initial velocity and the initial acceleration (at k), stacked into the vector $\hat{x}_k$,

and in terms of the sequence of jerks applied during these intervals. We recall that this sequence will be considered as our *controls* to drive the robot steps. Let us denote the sequence of the CoM $x-$positions starting from k and ending at $k+N-1$ as

$$X_k \stackrel{\text{def}}{=} (x_k, x_{k+1}, ..., x_{k+N-1})^\top,$$

and, similarly, the sequence of jerks applied from k as

$$\dddot{X}_k \stackrel{\text{def}}{=} (\dddot{x}_k, \dddot{x}_{k+1}, ..., \dddot{x}_{k+N-1})^\top.$$

Then, by applying the dynamics N times from k (the current time index), we get another linear system of the form

$$X_{k+1} = \mathbf{P}_x \hat{x}_k + \mathbf{P}_u \dddot{X}_k, \tag{2}$$

where $\mathbf{P}_x$ and $\mathbf{P}_u$ are respectively $N \times 3$ and $N \times N$ matrices that are easily deduced from the previous dynamics equations. Similar expressions can be obtained for the vectors stacking the velocities ($\dot{X}_{k+1}$) and accelerations ($\ddot{X}_{k+1}$) of the CoM during the considered horizon time window.

In the original approach [Kajita et al., 2003], the positions of the center of pressure (CoP) had to be predefined, so that preliminary footstep planning was necessary. Later on, an interesting re-formulation was proposed to handle automatic footstep placement [Herdt et al., 2010a]. This approach is two-step: (1) it first determines the sequence of robot trunk orientations to be followed and (2) given the computed orientations, it determines the x, y trajectory of the CoM. Let us describe these two steps.

B. Determination of the x, y CoM trajectory

Given the CoM orientations, this step makes only use of a reference velocity $(\dot{X}^{ref}_{k+1}, \dot{Y}^{ref}_{k+1})$, given as an input, in such a way that the determination of the optimal controls can be written as the following constrained optimization problem:

$$\begin{aligned} \min_{U_k} \quad & \frac{\alpha_T}{2} \left\| \dddot{X}_k \right\|^2 + \frac{\alpha_T}{2} \left\| \dddot{Y}_k \right\|^2 \\ & + \frac{\beta_T}{2} \left\| \dot{X}_{k+1} - \dot{X}^{ref}_{k+1} \right\|^2 + \frac{\beta_T}{2} \left\| \dot{Y}_{k+1} - \dot{Y}^{ref}_{k+1} \right\|^2 \\ & + \frac{\gamma_T}{2} \left\| Z^x_{k+1} - Z^{x_{ref}}_{k+1} \right\|^2 + \frac{\gamma_T}{2} \left\| Z^y_{k+1} - Z^{y_{ref}}_{k+1} \right\|^2 \end{aligned} \tag{3}$$

where $U_k \stackrel{\text{def}}{=} \left(\dddot{X}_k^\top, (X^f_k)^\top, \dddot{Y}_k^\top, (Y^f_k)^\top \right)^\top$ contains the variables to be optimized, and $(\alpha_T, \beta_T, \gamma_T)$ are weighting parameters. The sequence of CoP reference positions $Z^{x_{ref}}_{k+1}, Z^{y_{ref}}_{k+1}$ are the centers of the support polygons in single support at each iteration and they depend *linearly* on the variables X^f_k, Y^f_k, which are the sequence of positions of the next footsteps in the horizon. The linear relationship has the form

$$Z^{x_{ref}}_{k+1} = \mathbf{V}^c_{k+1} x^f_k + \mathbf{V}_{k+1} X^f_k$$

where x^f_k is the (known) $x-$position of the *current* support foot, X^f_k the sequence of (relative) steps to be done (i.e. our variable to optimize), and $\mathbf{V}^c_{k+1}, \mathbf{V}_{k+1}$ two constant selection matrices.

The variable $Z^x_{k+1} = [\xi^x_{k+1} \ \cdots \ \xi^x_{k+N}]$ contains the sequence of CoP positions along the time window. Again, they can be derived as a linear function of U_k from the second equation of the system 1.

All reunited, the optimization problem of Eq. 3 can be finally written as a canonical Quadratic Program (QP)

$$\min_{U_k} \frac{1}{2} U_k^\top \mathbf{Q}_k U_k + p_k^\top U_k.$$

Now, of course, this optimization problem comes with a set of constraints described hereafter.

C. Constraints on the CoP and on the foot placement

Because the robot feet can only push on the ground, the CoP has to lie inside the support polygon, i.e. the convex hull of the contact points between the feet and the ground [Wieber, 2002]. We assume that the foot on the ground has a known polygonal shape, so that the CoP constraint at time j can be expressed as a set of inequalities on the position of the CoP (ξ^x_j, ξ^y_j)

$$\begin{bmatrix} a^x_1\left(\theta^f_j\right) & a^y_1\left(\theta^f_j\right) \end{bmatrix} \begin{bmatrix} \xi^x_j - x^f_j \\ \xi^y_j - y^f_j \end{bmatrix} \leq b_1\left(\theta^f_j\right).$$

where the vector a^x_1 (resp. a^y_1) contains the x (resp. y) coordinates of the vectors normal to the feet edges and b_1 the positioning of these edges along this normal. These equations are linear with respect to the position of the foot (x^f_j, y^f_j) on the ground, each of which is an element of the aforementioned $Z^{x_{ref}}_{k+1}$ and $Z^{y_{ref}}_{k+1}$ for $k+1 \leq j \leq k+N$. However, it is nonlinear with respect to the foot orientation θ^f_j. This is the explanation of the two-phase approach: since the orientations are computed beforehand, then the inequalities do not depend on them and are all linear in the problem variables.

As for the next foot placement, several constraints arise from the joints limits, the prevention of self-collision, maximum leg lengths, etc. and most of them are highly non-linear in the variables of the problem. Here, we use the approach of [Perrin et al., 2010] to approximate these constraints by a series of *linear* constraints on the variables of interest, here the feet positions at consecutive time indexes,

$$\begin{bmatrix} a^x_2 & a^y_2 \end{bmatrix} \begin{bmatrix} x^f_j - x^f_{j-1} \\ y^f_j - y^f_{j-1} \end{bmatrix} \leq b_2.$$

D. Determination of the θ trajectory.

As explained above, the robot trunk orientation is determined *beforehand* to avoid non-linearities in Eq. III-C and keep the optimization problem as a QP. Several schemes have been proposed to determine these orientations. The one in [Herdt et al., 2010a] sets up another QP problem where the cost function includes the minimization of the difference between the feet orientation and the θ angle, and the tracking of a reference angular velocity. For more details, the reader is kindly invited to refer to [Herdt et al., 2010a].

We will propose a variant of this method in Section VI-B.

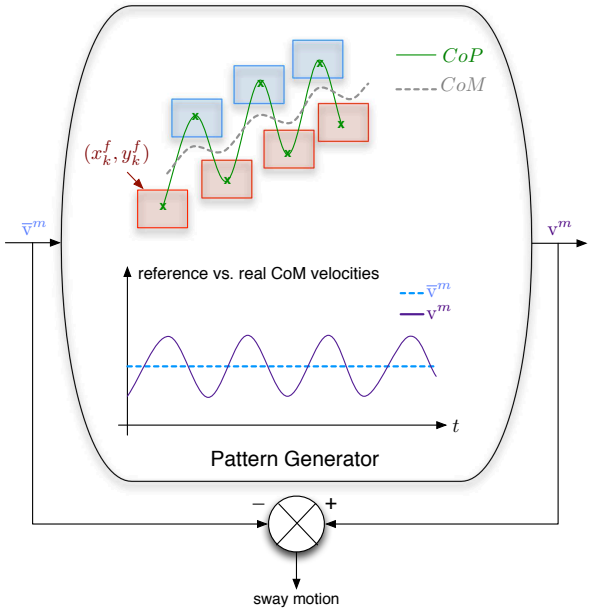

Fig. 2. The pattern generator ensures that the input velocity is tracked on average on the preview horizon. The output of the Model Predictive Control is the first control computed on the preview horizon. The difference between the reference velocity and the real velocity is mostly a sway motion due to the stepping.

E. Sway motion

As for any pattern generator, the stepping motion induces a sway motion, which can be read as the difference existing between the reference velocity used in the Eq. 3 of the Model Predictive Control and the real velocity effectively attained from the first control, as illustrated in Fig. 2. This sway motion is necessary for a proper walk, but it obviously generates non-desired effects on the robot visual perception. As stated in [Dune et al., 2010], the result of this sway motion on the real camera velocity v^c can be modeled as:

$$\mathrm{v}^c = \overline{\mathrm{v}}^c + {}^c\mathbf{V}_m \mathrm{v}^b \tag{4}$$

where $\overline{\mathrm{v}}^c$ is the "ideal" camera velocity that would exist without sway, and v^b is the part of the CoM velocity that is induced by the sway. The matrix ${}^c\mathbf{V}_m$ is a twist matrix relating the camera frame (index "c") to the CoM frame (index "m") through the transform ${}^c\mathbf{T}_m$.

We will now describe two approaches for introducing visual control in the described pattern generation: the first one (in Section V) is a decoupled one, where the visual servoing is used to determine the reference velocity in Eq. 3; the second one (Section VI) modifies Eq. 3 to replace the reference velocity term by a new term directly related to the decrease of the visual errors.

IV. Visual Servoing schemes

Visual servoing aims at controlling the motion of a robot equipped with a camera, by minimizing the errors between observed features (denoted in the following as s) and their corresponding reference, desired features (denoted as s^*) [Chaumette and Hutchinson, 2006]. The nature of the features differentiates schemes of visual servoing: Image based visual servoing (IBVS) uses only image features, i.e. s may be a vector of image points coordinates; Position based visual servoing (PBVS) uses the 3-D pose(s) of object(s) of interest. In any of these schemes, one may use a velocity controller based on these errors [Chaumette and Hutchinson, 2006],

$$\mathrm{v}^c = -\lambda \widehat{\mathbf{L}_e}^+ e, \tag{5}$$

where $e = s - s^*$ is the vector of errors, v^c is the velocity of the camera and $\widehat{\mathbf{L}}_e^+$ is the Moore-Penrose pseudo-inverse of an approximation $\widehat{\mathbf{L}}_e$ of the interaction matrix $\mathbf{L}_e$ related to e. That is, the matrix relating the velocity of the features and the velocity of the camera.

In subsection IV-A, we briefly describe the visual tracking system that allows us to track the visual features and estimate the pose of the camera with respect to the features; then in subsection IV-A, we describe the classical approach for IBVS used in [Dune et al., 2010]. Finally, in IV-C, we recall how IBVS has been previously expressed as a Model Predictive Control scheme coined as *Visual Predictive Control*.

A. Robust Model Based Tracking

In the case of the humanoid robots, one difficulty is that when the robot walks, the stepping causes the head to shake and oscillate. At each foot step, the impact propagates to the camera. Moreover, the inherent sway motion of the CoM translates into camera motion. Both causes generate blur and shift in the image. Hence, the extraction of the visual features, by image processing, is not that easy, e.g. compared to the case of wheeled robots. This explains in part why 3D information is more robust to be used in that case than only 2D, i.e., PBVS instead of IBVS.

To face the image processing problem, we use the robust approach presented in [Comport et al., 2006] for the tracking of geometrical shapes (lines, cylinders, ellipsoids, etc.). This algorithm estimates the 3D position of a known object frame (index "o") in the camera frame $\widehat{{}^c\mathbf{T}_o}$, based on two steps: (1) tracking of the contour points along the tracked shape and (2) optimization for the pose estimation. Basically, starting from a predicted pose, the lines of the geometric object model are projected on the image and sampled regularly. Then, at each of these sample points, the maximum gradient magnitude point p_i is searched along the normal to the projected line. These locally registered points are used in the second step to optimize the camera-object transform

$$\widehat{{}^c\mathbf{T}_o} = \arg\min_{{}^c\mathbf{T}_o} \sum_i \mathcal{C}(d_\perp(p_i, l_i({}^c\mathbf{T}_o))) \tag{6}$$

where $\mathcal{C}$ is a robust function that allows to handle outliers in an M-Estimator way. The distance $d_\perp$ is the orthogonal distance from a point to a line. p_i are the extracted points in the current image and l_i are the projections of the model lines. As the points p_i should belong to the lines l_i in the

image plane, ${}^{\widehat{c}}\widehat{\mathbf{T}_o}$ is searched as the one minimizing these point-line distances.

Hence, to implement satisfactorily the visual processing part of our approach, we rely on the algorithm presented in [Comport et al., 2006] which assumes that the target, i.e. the entire set of visual features, is a polygon, with edges prominent enough for the aforementioned tracking approach to be successful. We also suppose that the possibly rough initialization is done manually before the first iteration of this visual tracking.

B. Elements for a Classical Visual Servoing Approach

To use an Image-Based Visual Servoing (IBVS) setup for the humanoid robot, we first define the current and the desired features s and s^* in terms of the (u, v) coordinates of visual features in the image.

With that design choice, the approximated interaction matrix $\widehat{\mathbf{L}_e}$ associated to the error for just one feature in this system, i.e., regarding two lines in the visual features error vector e, can then be determined as [Chaumette and Hutchinson, 2006]

$$\widehat{\mathbf{L}_e} = \begin{pmatrix} -1/z^c & 0 & u/z^c & uv & -(1+u^2) & v \\ 0 & -1/z^c & v/z^c & (1+v^2) & -uv & -u \end{pmatrix}, \tag{7}$$

where z^c is the z coordinate (depth) of the single visual feature. It can be estimated from the transform ${}^{\widehat{c}}\widehat{\mathbf{T}_o}$, for example, which estimation is described above.

Then, the velocity control for the camera ensuring that e decreases exponentially is defined as in Eq. 5. This translates into a separate control for the translational and rotational velocities [Chaumette and Hutchinson, 2006].

From this objective velocity computed for the camera, we can then deduce the CoM reference speed $\overline{\mathrm{v}}^m$ by making use of the twist matrix ${}^m\mathbf{V}_c$:

$$\overline{\mathrm{v}}^m = -\lambda {}^m\mathbf{V}_c \widehat{\mathbf{L}_e}^+ e. \tag{8}$$

The only problem in this approach is the one mentioned just before: The handling of sway motion. We will see in Section V how this has been handled in [Dune et al., 2010].

C. Elements for an MPC-based Visual Servoing Approach

Now, in WPG (see Section III), we saw that MPC is used to estimate a sequence of optimal controls at some horizon, because of the step-based nature of walking. Hence, we may want to orient the optimization of the foot placement by taking into account the expected evolution of the visual servoing (VS) errors so that, instead of minimizing the VS errors at current time k, one would like to foresee its evolution at some horizon $[k+1, k+N]$. In [Allibert et al., 2010], outside of the humanoid context, such a time horizon-aware scheme has been proposed for Image-Based Visual Servoing, and we will show that it can be applied to our own scheme. The visual predictive control (VPC) has been introduced in general terms as:

$$\min_{U_k} \quad \sum_{j=k+1}^{k+N} [s_j^d - s_j^m]^\top \mathbf{W}_j [s_j^d - s_j^m], \tag{9}$$

$$\text{subject to} \quad s_j^d = s_j^* - \epsilon_j, \tag{10}$$

$$q_j = f(q_{j-1}, u_{j-1}), \tag{11}$$

$$s_j^m = h(q_j). \tag{12}$$

In Eq. 9, $U_k = u_{k:k+N-1}$ are the series of controls to be applied to the camera, j refers to time indices in the future, q_j is the state. The feature vectors s_j^*, s_j^d and s_j^m are respectively the reference, desired and predicted positions of the visual features. The terms ϵ_j are the errors $s_j - s_j^m$ between real and predicted feature positions.

Allibert et al. assume ϵ_j constant over the prediction horizon, equal to $\epsilon_k = s_k - s_k^m$, i. e. the error at the current time k, because by definition the s_j are not known for $j > k$. Since our landmarks are static, $s_j^* \overset{\text{def}}{=} s^*$, and since the prediction errors are constant on the horizon window, $s_j^d = s_k^d = s^* - \epsilon_k$ are constant in the prediction horizon.

Eq. 11 is the dynamic model, that estimates the new state given the last state/control pair. In general, this function f is non-linear. We will see how to deal with this non-linearity.

Eq. 12 is also a non-linear function h that estimates the output of the model s_j^m, given the current state q_j. In practice, this equation implements the pinhole camera projection model.

The matrix $\mathbf{W}_j$ in Eq. 9 is a positive definite matrix used to weight errors in the prediction horizon. As suggested in [Allibert et al., 2010], we consider equal weights for all features errors, $\mathbf{W}_j = diag(w_j)$.

Here, s_j^m is the collection of all the predicted features at time j, that is, for M features $s_j^m = (s_{1,j}^m, s_{2,j}^m, \ldots, s_{M,j}^m)^\top$. Eq. 9 can be rewritten as

$$\min_{U_k} \sum_{\lambda=0}^{M} [S_\lambda^d - S_{\lambda,k}^m]^\top \mathbf{W} [S_\lambda^d - S_{\lambda,k}^m], \tag{13}$$

where $\lambda \in [1, M]$ is the visual feature index, $S_{\lambda,k}^m$ stacks the positions of feature λ in the horizon:

$$S_{\lambda,k}^m = \left(s_{\lambda,k+1}^m, s_{\lambda,k+2}^m, \ldots, s_{\lambda,k+N}^m\right)^\top,$$

S_λ^d stacks the corresponding desired positions, and, finally, $\mathbf{W} = diag(w_{k+1}, w_{k+2}, ..., w_{k+N})$.

This approach is completely different from the one seen just before in IV-B, and we will see in Section VI how to use it in a MPC scheme for WPG.

V. Using the visual servoing output as the WPG reference velocity

In this section, we recall how in [Dune et al., 2010], a classical visual servoing scheme such as the one described in IV-B is adapted to be used for a humanoid robot. The idea, basically, is to use the output velocity derived from the visual errors through Eq. 8 as a reference velocity for Eq. 3. However, as we have pointed it out, the humanoid motion has

an intrinsic sway component, that makes the visual features have an evolution in the image that do not correspond to the desired trajectory.

As mentioned above, the features motion in the image can be decomposed into a component due to the sway motion, and a component due to the "average" sway-less motion of the robot,

$$\dot{s} = \widehat{\mathbf{L}_e}\overline{\mathrm{v}}^c + \widehat{\mathbf{L}_e}\,{}^c\mathbf{V}_m \mathrm{v}^b. \tag{14}$$

Then, the idea developed in [Dune et al., 2010] is (1) to use a virtual camera that corresponds to the position of the camera if we suppose that there is no sway motion applying to the robot, (2) to control this virtual camera based on the regulation of visual errors, and (3) to use its controlled velocity ($\overline{\mathrm{v}}^c$) as a reference velocity in the reactive WPG.

Dune et al. have shown that the relationship between the observed features $s(t)$ and the "sway-less" features $\overline{s}(t)$ can be written as

$$s(t) = \overline{s}(t) + \int_0^t \widehat{\mathbf{L}_e}\,{}^c\mathbf{V}_m \mathrm{v}^b d\tau - E, \tag{15}$$

where $E \stackrel{\text{def}}{=} s(0) - \overline{s}(0)$. From this, the corrected visual error to be used for the computation of the reference velocity is deduced as

$$\overline{e}(t) = \overline{s}(t) - s^* = e(t) - (\int_0^t \widehat{\mathbf{L}_e}\,{}^c\mathbf{V}_m \mathrm{v}^b d\tau - E). \tag{16}$$

Under this model, the virtual error $\overline{e}(t)$ is regulated to zero, and the real error $e(t)$ is oscillating around zero, with a period T. The shift E is re-estimated regularly over one period of time (e.g., the last period) by

$$E = \frac{1}{T}\int_{t-T}^{t}\int_0^t \widehat{\mathbf{L}_e}\,{}^c\mathbf{V}_m \mathrm{v}^b d\tau d\tau',$$

and finally, it is used in the control law for the CoM (e.g., the reference velocities $\dot{X}^{ref}_{k+1}, \dot{Y}^{ref}_{k+1}$ for Eq. 3)

$$\overline{\mathrm{v}}^m = -\lambda {}^c\mathbf{V}_m \widehat{\mathbf{L}_e}^{+}(e - (\int_0^t \widehat{\mathbf{L}_e}\,{}^c\mathbf{V}_m \mathrm{v}^b d\tau - E)). \tag{17}$$

Note that the discrete form of the involved integrals are used in practice. More details can be found in [Dune et al., 2010]. Please note that, as in any classical visual servoing approach, to apply this control, we need the actual measured errors e, an estimate of the sway motion period noted T, which is deduced from the stepping period, and a point-wise estimate for the sway motion at the level of the CoM, i.e. v^b.

VI. Integrating the visual servoing to the walking motion generator

In this section, we describe our MPC-based approach for plugging the visual servoing error regulation within the dynamic walk control.

As already mentioned, the basic idea is to use the visual errors between the observed features and the desired ones inside the pattern generator, with an MPC approach. However, it should be clear that if we introduce directly Eq. 13 as a new term for Eq. 3, we will not have a QP formulation anymore, due to the non-linear constraints, namely Eqs. 11 and 12. We can avoid the first non-linearity (Eq. 11) by using the dynamic model in Eq. 2. In this case, there is no rotation, but we will see that we can introduce it in a separate optimization process without losing the QP formulation.

A. Linearization of the observation model

As already mentioned, Eq. 12 implements the classical pinhole camera model. Let $p^o_l = (x^o_l, y^o_l, z^o_l)^\top$ be the position of the l-th landmark in the object of interest reference frame (index "o"). At time j, one can compute the projection of this landmark onto the image plane by first transforming the landmark position to the camera frame with the homogeneous transform ${}^c\mathbf{T}_m\,{}^m\mathbf{T}_{o,j}$ and then by applying the perspective projection to the coordinates in the camera frame $(x^c_{l'}, y^c_{l'}, z^c_{l'})^\top$. Assuming a camera with canonical parameters, this takes the form

$$\begin{pmatrix} u_{l,j} \\ v_{l,j} \end{pmatrix} = \begin{pmatrix} u(x^c_{l,j}, y^c_{l,j}, z^c_{l,j}) \\ v(x^c_{l,j}, y^c_{l,j}, z^c_{l,j}) \end{pmatrix} = \begin{pmatrix} x^c_{l,j}/z^c_{l,j} \\ y^c_{l,j}/z^c_{l,j} \end{pmatrix}. \tag{18}$$

At this point, we recall for clarity that the exponent "c" stands for the camera frame, the index "l'" for the landmark id, and "j" for the time index. Also note that ${}^m\mathbf{T}_{o,j}$ is the transformation at time j from the object of interest frame (index "o") to the CoM frame (index "m") and ${}^c\mathbf{T}_m$ is the transformation from the CoM frame to the camera frame, which is considered as constant in our approach. Hence, there is no index j associated to it. Observe that

$${}^m\mathbf{T}_{o,j} = ({}^o\mathbf{T}_{m,j})^{-1} = \begin{pmatrix} ({}^o\mathbf{R}_{m,j})^{-1} & -({}^o\mathbf{R}_{m,j})^{-1}\,{}^o t_{m,j} \\ 0_{1\times 3} & 1 \end{pmatrix}$$

where ${}^o t_{m,j}$ is the position of the CoM in the object of interest frame at time j, which depends directly in our control variables through Eq. 2. The rotation matrix ${}^o\mathbf{R}_{m,j}$ is the direction of the robot waist according to the object of interest frame at time j. In our current formulation, there is no free variable modifying this quantity, because it would make the problem non-linear. This argument is a similar one as in [Herdt et al., 2010a], mentioned in Section III, which justifies the use of separate optimization processes for translation and rotation. More details about this problem are given in Section VI-B. For the moment, consider ${}^o\mathbf{R}_{m,j}$ and its inverse as constants.

If we use directly the non-linear equation Eq. 18 in ${}^o t_{m,j}$, we will lose the QP formulation. We know that Eq. 18 is a projection $h : \mathbb{R}^3 \to \mathbb{R}^2$,

$$h(x, y, z) = \begin{pmatrix} u(x, y, z) \\ v(x, y, z) \end{pmatrix} = \begin{pmatrix} x/z \\ y/z \end{pmatrix}.$$

We also know that in the MPC formulation, the prediction is done over a *finite* horizon, corresponding to relatively small distances. Typically, a few dozen centimeters are considered. So it might be enough to use a first order approximation of h for small variations (dx, dy, dz) of the features position in the camera frame, so that we can maintain the QP form.

By using a Taylor series for $u(x, y, z)$ around some linearization point (x_0, y_0, z_0) and substituting the derivatives,

$$\left\{ \begin{array}{l} u(x_0+dx, y_0+dy, z_0+dz) \approx \frac{x_0}{z_0} + \frac{dx}{z_0} - \frac{x_0 dz}{z_0^2}, \\ v(x_0+dx, y_0+dy, z_0+dz) \approx \frac{y_0}{z_0} + \frac{dy}{z_0} - \frac{y_0 dz}{z_0^2}. \end{array} \right. ,$$

with $dx = x - x_0$, $dy = y - y_0$ and $dz = z - z_0$.

We propose to apply such a linearization of Eq. 18 for the whole horizon, around the first position $(j = k)$ of the l-th landmark, i.e. at the linearization point $(x^c_{l,k}, y^c_{l,k}, z^c_{l,k})$.

This way, we can express the predicted position of the l-th landmark, at time $j > k$ in the horizon, in a linear way:

$$\begin{pmatrix} u_{l,j} \\ v_{l,j} \end{pmatrix} = \begin{pmatrix} \pi^{11}_{l,k} x^c_{l,j} + \pi^{13}_{l,k} z^c_{l,j} + u_{l,k} \\ \pi^{22}_{l,k} y^c_{l,j} + \pi^{23}_{l,k} z^c_{l,j} + v_{l,k} \end{pmatrix},$$

where $u_{l,k} = x^c_{l,k}/z^c_{l,k}$ and $v_{l,k} = y^c_{l,k}/z^c_{l,k}$ are the initial positions of the landmarks in the horizon and the coefficients $\pi^{ij}_{l,k}$ are the elements of the following matrix

$$\mathbf{\Pi}_{l,k} = \begin{pmatrix} 1/z^c_{l,k} & 0 & -u_{l,k}/z^c_{l,k} \\ 0 & 1/z^c_{l,k} & -v_{l,k}/z^c_{l,k} \end{pmatrix},$$

which is the classical interaction matrix in Image-Based Visual Servoing. Finally, we can express the projection of the l-th landmark (constraint 12) as:

$$\begin{pmatrix} u_{l,j} \\ v_{l,j} \end{pmatrix} = \left[\begin{array}{cc} \mathbf{\Pi}_{l,k} & \begin{array}{c} u_{l,k} \\ v_{l,k} \end{array} \end{array} \right] {}^c\mathbf{T}_m \, {}^m\mathbf{T}_{o,j} \begin{pmatrix} p^o_l \\ 1 \end{pmatrix}, \quad (19)$$

so that we can now introduce the visual errors regulation in the pattern generator. By expanding the first row in Eq. 19 and by referring to the first row of matrix $\mathbf{\Pi}_{l,k}$ as $\mathbf{\Pi}^u_{l,k}$,

$$u_{l,j} = \mathbf{\Pi}^u_{l,k} ({}^c\mathbf{R}_o \, p^o_l + {}^c\mathbf{R}_o \, {}^o t_{m,j} + {}^c t_m) + u_{l,k}. \quad (20)$$

Since ${}^c\mathbf{R}_o \, {}^o t_{m,j} = {}^c\mathbf{R}_o(1) x_j + {}^c\mathbf{R}_o(2) y_j + {}^c\mathbf{R}_o(3) z_j$, where ${}^c\mathbf{R}_o(i)$ is the i-th column of ${}^c\mathbf{R}_o$ [1] and x_j, y_j, z_j the position of the CoM in the object of interest frame at time j (see Section III), we can rewrite Eq. 20 in the simpler form

$$u_{l,j} = a^u_{l,k} x_j + b^u_{l,k} y_j + c^u_{l,k}, \quad (21)$$

with

$$\left\{ \begin{array}{lcl} a^u_{l,k} & = & \mathbf{\Pi}^u_{l,k} \, {}^c\mathbf{R}_o(1) \\ b^u_{l,k} & = & \mathbf{\Pi}^u_{l,k} \, {}^c\mathbf{R}_o(2) \\ c^u_{l,k} & = & \mathbf{\Pi}^u_{l,k} \, ({}^c\mathbf{R}_o p^o_l + {}^c t_m + {}^c\mathbf{R}_o(3) z_j) + u_{l,k}. \end{array} \right.$$

Note that $z_j = c_z$ is assumed constant here (see Section III). Similarly to the previous equation, we also get

$$v_{l,j} = a^v_{l,k} x_j + b^v_{l,k} y_j + c^v_{l,k}. \quad (22)$$

By stacking the features $u_{l,j}$ and the CoM positions for the whole horizon, and by using Eq. 2, we get a vector $S^{u,m}_{l,k}$ similar to the one introduced in Eq. 13. We recall that $S^{u,m}_{l,k} = \left(s^{u,m}_{l,k+1}, s^{u,m}_{l,k+2}, \ldots, s^{u,m}_{l,k+N} \right)^\top$ stacks the l-th landmark u-values over the time window. It can now be expressed as,

$$S^{u,m}_{l,k} = \mathbf{A}^u_{l,k} X_{k+1} + \mathbf{B}^u_{l,k} Y_{k+1} + \mathbf{C}^u_{l,k},$$

with

$$\begin{array}{lcl} \mathbf{A}^u_{l,k} & = & a^u_{l,k} \mathbf{I}_{N \times N}, \\ \mathbf{B}^u_{l,k} & = & b^u_{l,k} \mathbf{I}_{N \times N}, \\ \mathbf{C}^u_{l,k} & = & c^u_{l,k} (1, 1, ..., 1)^\top. \end{array}$$

This corresponds to the sequence of predicted coordinates u of the l-th landmark in the horizon. The equivalent equations for the v coordinates of the same landmark are straightforward.

Every projected landmark provides two coordinates (u, v) and we treat each one as an individual feature. This means that our λ-th feature is the u (resp. v) image coordinate of the landmark $l = \lfloor \lambda/2 \rfloor$ for λ even (resp. odd). In other terms, for any time index j,

$$\left\{ \begin{array}{lcl} s^m_{\lambda,j} & \stackrel{\text{def}}{=} & u_{\lfloor \frac{\lambda}{2} \rfloor, j} \text{ for } \lambda \text{ even} \\ s^m_{\lambda,j} & \stackrel{\text{def}}{=} & v_{\lfloor \frac{\lambda}{2} \rfloor, j} \text{ for } \lambda \text{ odd.} \end{array} \right.$$

Generalizing to all features, we have:

$$S^m_{\lambda,k} = \mathbf{A}_{\lambda,k} X_{k+1} + \mathbf{B}_{\lambda,k} Y_{k+1} + \mathbf{C}_{\lambda,k}, \quad (23)$$

with $\mathbf{A}_{\lambda,k} = \mathbf{A}^u_{l,k}$ for λ even and $\mathbf{A}_{\lambda,k} = \mathbf{A}^v_{l,k}$ for λ odd, where $l = \left\lfloor \frac{\lambda}{2} \right\rfloor$. The same holds for $\mathbf{B}_{\lambda,k}$ and $\mathbf{C}_{\lambda,k}$.

Finally, we introduce visual servoing in the walking generation with the QP:

$$\begin{array}{ll} \min\limits_{U_k} & \frac{\alpha_T}{2} \left\| \dddot{X}_k \right\|^2 + \frac{\alpha_T}{2} \left\| \dddot{Y}_k \right\|^2 \\ & + \frac{\beta_T}{2} \sum\limits_{\lambda=0}^{M} [S^d_\lambda - S^m_{\lambda,k}]^\top \mathbf{W} [S^d_\lambda - S^m_{\lambda,k}] \\ & + \frac{\gamma_T}{2} \left\| Z^x_{k+1} - Z^{x_{ref}}_{k+1} \right\|^2 + \frac{\gamma_T}{2} \left\| Z^y_{k+1} - Z^{y_{ref}}_{k+1} \right\|^2, \end{array}$$

with the set of weights $(\alpha_T, \beta_T, \gamma_T)$. We can rewrite it as a canonical QP:

$$\min_{U_k} \frac{1}{2} U_k^\top \mathbf{Q}_k U_k + p_k^\top U_k$$

with

$$\mathbf{Q}_k = \begin{pmatrix} \mathbf{Q}'_k & 0 \\ 0 & \mathbf{Q}'_k \end{pmatrix} + \hat{\mathbf{Q}}_k,$$

$$\mathbf{Q}'_k = \begin{pmatrix} \alpha_T \mathbf{I} + \gamma_T \mathbf{P}^\top_{zu} \mathbf{P}_{zu} & -\gamma_T \mathbf{P}^\top_{zu} \mathbf{V}_{k+1} \\ -\gamma_T \mathbf{V}^\top_{k+1} \mathbf{P}_{zu} & \gamma_T \mathbf{V}^\top_{k+1} \mathbf{V}_{k+1} \end{pmatrix},$$

[1] To simplify the notations we dropped the time index j.

$$\hat{\mathbf{Q}}_k = \begin{pmatrix} \beta_T \sum_\lambda \mathbf{P}_{pu}^\top \mathbf{A}_{\lambda,k}^\top \mathbf{W} \mathbf{A}_{\lambda,k} \mathbf{P}_{pu} & 0 & \beta_T \sum_\lambda \mathbf{P}_{pu}^\top \mathbf{A}_{\lambda,k}^\top \mathbf{W} \mathbf{B}_{\lambda,k} \mathbf{P}_{pu} & 0 \\ 0 & 0 & 0 & 0 \\ \beta_T \sum_\lambda \mathbf{P}_{pu}^\top \mathbf{B}_{\lambda,k}^\top \mathbf{W} \mathbf{A}_{\lambda,k} \mathbf{P}_{pu} & 0 & \beta_T \sum_\lambda \mathbf{P}_{pu}^\top \mathbf{B}_{\lambda,k}^\top \mathbf{W} \mathbf{B}_{\lambda,k} \mathbf{P}_{pu} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

and $p_k = p'_k + \hat{p}_k$,

$$p'_k = \begin{pmatrix} \gamma \mathbf{P}_{zu}^\top (\mathbf{P}_{zs} \hat{x}_k - \mathbf{V}_{k+1}^c x_k^f) \\ -\gamma \mathbf{V}_{k+1}^\top (\mathbf{P}_{zs} \hat{x}_k - \mathbf{V}_{k+1}^c x_k^f) \\ \gamma \mathbf{P}_{zu}^\top (\mathbf{P}_{zs} \hat{y}_k - \mathbf{V}_{k+1}^c y_k^f) \\ -\gamma \mathbf{V}_{k+1}^\top (\mathbf{P}_{zs} \hat{y}_k - \mathbf{V}_{k+1}^c y_k^f) \end{pmatrix},$$

where (x_k^f, y_k^f) is the foot position at the beginning of the time window, $\hat{x}_k \stackrel{\text{def}}{=} (x_k, \dot{x}_k, \ddot{x}_k)^\top$, $\hat{y}_k \stackrel{\text{def}}{=} (y_k, \dot{y}_k, \ddot{y}_k)^\top$ (see Section III), and finally

$$\hat{p}_k = \begin{pmatrix} \beta_T \sum_\lambda \mathbf{P}_{pu}^\top \mathbf{A}_{\lambda,k}^\top \mathbf{W} [\mathbf{A}_{\lambda,k} \mathbf{P}_{ps} \hat{x}_k + \mathbf{B}_{\lambda,k} \mathbf{P}_{ps} \hat{y}_k + \mathbf{C}_{\lambda,k} - S_\lambda^d] \\ 0 \\ \beta_T \sum_\lambda \mathbf{P}_{pu}^\top B_{\lambda,k}^\top \mathbf{W} [\mathbf{A}_{\lambda,k} \mathbf{P}_{ps} \hat{x}_k + \mathbf{B}_{\lambda,k} \mathbf{P}_{ps} \hat{y}_k + \mathbf{C}_{\lambda,k} - S_\lambda^d] \\ 0 \end{pmatrix}.$$

For details on matrices $\mathbf{P}_{ps}$, $\mathbf{P}_{zs}$, $\mathbf{P}_{pu}$, $\mathbf{P}_{zu}$, $\mathbf{V}_{k+1}$ and $\mathbf{V}_{k+1}^c$ see [Herdt et al., 2010a].

To summarize, our approach, together with the elements described hereafter (control of the rotation angle in VI.C and handling of the visual constraints in VI.D), follows a Position Based scheme, as depicted in Fig. 3.

B. Control of the rotation angle

So far, we have proposed a scheme to control the trajectory of the center of mass in the xy plane. However, introducing the rotation angle in the minimization problem is not straightforward without losing linearity. Furthermore, the rotation angle plays a very important role here since sometimes most of the error between the desired s^d and the predicted s^m may be due to the angle between the robot and the features.

An extension of the original linear MPC scheme with automatic footstep placement that deals with a reference angular velocity has been proposed in [Herdt et al., 2010b], as mentioned in III-D. The approach is a two-fold optimization process that first estimates the optimal rotation angles and then introduces these values as known in the main QP (here, as the matrix ${}^o\mathbf{R}_{m,j}$). This scheme should not affect the stability of the walking since inertial effects are not taken into account.

The same methodology is used in this approach. Hence, in a first stage, we optimize the orientations in the MPC time window by

$$\min_{\dddot{\Theta}_k, \dddot{\Theta}_k^f} \quad \frac{\alpha_R}{2} \left\| \dddot{\Theta}_k \right\|^2 + \frac{\alpha_R}{2} \left\| \dddot{\Theta}_k^f \right\|^2 + \frac{\beta_R}{2} \left\| \Theta_{k+1} - \Theta^0 \right\|^2 + \frac{\gamma_R}{2} \left\| \Theta_{k+1}^f - \Theta^0 \right\|^2, \tag{24}$$

where, with the same notations as for $\dddot{X}_k$ and $\dddot{Y}_k$, $\dddot{\Theta}_k$ is the sequence of N jerk values to be applied, and Θ_{k+1} is the sequence of N predicted θ values, i.e. the orientations of the trunk in the horizon,

$$\Theta_{k+1} \stackrel{\text{def}}{=} (\theta_{k+1}, ..., \theta_{k+N})^\top,$$

and similarly for Θ_{k+1}^f, the feet orientations. A reference orientation Θ^0 is defined once for all at the starting configuration as a target feet orientation. Several conventions exist but in this paper, the trunk orientation Θ_k is trying to follow the flying foot orientation Θ_k^f. The flying foot is the only one which can move during the single support phase. Indeed obviously the support foot is fixed, and both feet are fixed during the double support phase. Finally, zero speed, and zero acceleration are required at the beginning and the end of the trajectories.

Then in a second stage, we introduce these computed angles as constant in the main QP (Eq. 3). This approach gives us the advantage of introducing constraints like the maximum rotation between both feet, between the feet and the trunk, and also a rotation limit to keep the visibility of the landmarks.

C. Visual constraints

The visual constraints can be introduced by using Eq. 21 and Eq. 22. Any linear constraint in the image plane (u, v), can be expressed as a linear constraint in the variables U_k.

Furthermore, a convex polytope can be expressed under a linear form. It means that we can have time- and landmark-varying constraints. Commonly, we want all landmarks to follow trajectories inside some convex polytope. Hence, the constraints become constant in time and for all landmarks. This can be written as:

$$\begin{aligned} \mathbf{A}' \begin{pmatrix} \mathbf{A}_{l',k}^u X_{k+1} + \mathbf{B}_{l',k}^u Y_{k+1} + \mathbf{C}_{l',k}^u \\ \mathbf{A}_{l',k}^v X_{k+1} + \mathbf{B}_{l',k}^v Y_{k+1} + \mathbf{C}_{l',k}^v \end{pmatrix} &\leq \mathbf{b}' \\ \text{and then, } \mathbf{A}'' U_k &\leq \mathbf{b}'', \end{aligned} \tag{25}$$

where the matrix $\mathbf{A}'$ and the vector $\mathbf{b}'$ are related to the image constraints. For example, bound constraints in the (u, v) coordinates like visibility constraints are easily expressed in terms of Eq. 25 and are introduced directly in the QP.

D. Qualitative comparison with the classical approach

A first advantage of plugging the visual errors term in the Pattern Generator MPC and of avoiding the decoupled approach is that, with a pure visual servoing approach, the expected behavior of the controls to be applied would correspond to an exponentially decreasing velocity. Indeed, in classical visual servoing, when the goal is close, errors tend to zero, and the velocity controls requested to the robot get smaller and smaller. This is not a problem, e.g. with robotic arms, since in that case we just send rotational velocities to the motors, and these velocities can be as small as requested without consequence on the safety of the robot. In humanoid robots, having very small reference velocities is much more a problem, since it would involve more steps. As stepping involves balance, and as every step could break it, we must avoid unnecessary motion and reach the goal as

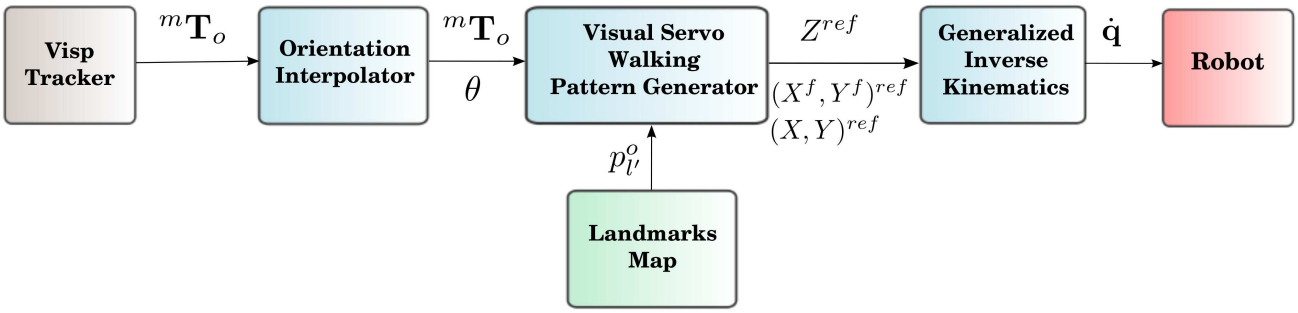

Fig. 3. In this scheme, the pose of the object in the camera frame ${}^m\mathbf{T}_o$ is sent to our control scheme. First, the orientation is found as described in section VI-B. Then, by the coordinates of the landmarks specified in the object reference frame (p^o_l) the visual servo based WPG provides a CoP reference trajectory Z^{ref}, a set of footprints $(X^f, Y^f)^{ref}$ and a Center-Of-Mass reference trajectory $(X, Y)^{ref}$. The reference trajectories are then followed by a Generalized Inverse Kinematics scheme which computes at each time step a velocity $\dot{\mathbf{q}}$ for the robot actuators. The vision part (gray box) is usually running at 30 to 60 Hz. The control part (blue boxes) is typically running at 200 Hz in the HRP-2 robot. The landmarks are constant (green box) and the control part using $\dot{\mathbf{q}}$ of the HRP-2 robot is typically running at 200 Hz.

soon and efficiently as possible. Here, with our approach, because the visual errors term is only one term in the QP problem, this exponential decay is strongly attenuated by the regularizing effect of other terms such as the jerks. This will be illustrated clearly in the experimental results section.

Another advantage of our approach is that the constraint on velocities such as maximal velocities are naturally handled, as inequality constraints in a QP problem.

Finally, since the velocity reference we set as an input to the pattern generator is not truly performed, due to the physical constraints of the robot, we have to re-inject this difference in the next iteration. In the coupled approach, those problems are handled intrinsically within the MPC. The sway motion is naturally filtered since we minimize the errors within a full cycle (the horizon in the MPC). Finally, in the MPC-based coupled approach, we minimize errors as long as the stability criteria permits it, so we always request and apply feasible controls and the error is instantaneously taken into account.

VII. Experimental results

A. Simulation results on the MPC-based approach only

We first tested our own approach (Section VI) in a simulated environment and we comment these results hereafter. We assume that no noise or modeling errors have been introduced. For all the tests, the initial position is $(0, 0)$ and the desired features are set in the position $(2, 1)$. The parameters α_T, β_T and γ_T used for the translational component control are by default respectively set to the values 0.001 , 0.001, and 1. The parameters α_R, β_R and γ_R used for the rotational component control are by default respectively set to the values 0.1 , 1, and 1. These default values were determined manually.

First, we perform an experiment with a desired final position that does not imply rotation, so that the robot has just to control the x and y velocities through the corresponding jerks, which are the variables in the QP. With the default parameters values mentioned above, we obtain the walk depicted in Fig. 4. As it can be seen, the visual errors (evaluated at each of the quadrilateron corners) converge to zero. We can note an offset in the oscillatory velocity in x (in blue) and y (in red) due to the features errors.

Now, in a second experiment, we evaluate the trajectory with rotation. We recall that the rotation velocity control is done in a separate process from the x and y velocities control. It works as follows: The rotation velocity controller sets the angular position while the main controller (QP) adapts the x, y velocities in terms of these computed angular positions the visual errors, the footsteps centering and the jerks minimization. The results of a first simulation involving rotation is shown in Fig. 5. One can observe that the dynamical balance is kept, and that most of the correction related to the rotation is done at the beginning of the trajectory. Also, in Fig. 6, we can see that it is robust to perturbations: The same trajectory is followed as in Fig. 5 but a strong perturbation in the CoM position has been introduced. It is simulating an external force applying to the CoM or a strong error in the position estimation, inducing peaks in the velocities. However, the perturbation is recovered quasi-instantly.

We have already explained how a local linearization is made to maintain the QP formulation. The performance of this linearization depends of the distance traveled inside the horizon, which depends on the velocity of the robot and the size of the horizon. In Fig. 7, we can see the linearized and real features trajectories for a given CoM trajectory. As expected, close to the beginning (the linearization point) the trajectories are quite similar, while the final positions differ more. This is an extreme situation, since usual metric displacements in the horizon are much smaller than this one. In any case, horizon displacements are bigger when the visual errors are bigger. i.e. the robot is far from the desired position, in which case the robot just needs a tendency. But when the errors are getting smaller, the robot needs more precision. In this case, the displacements in the horizon becomes smaller so that the difference between the real model and the linearized one becomes negligible.

Due to the walking nature, we have oscillations in the features trajectories. One of the main advantages of using

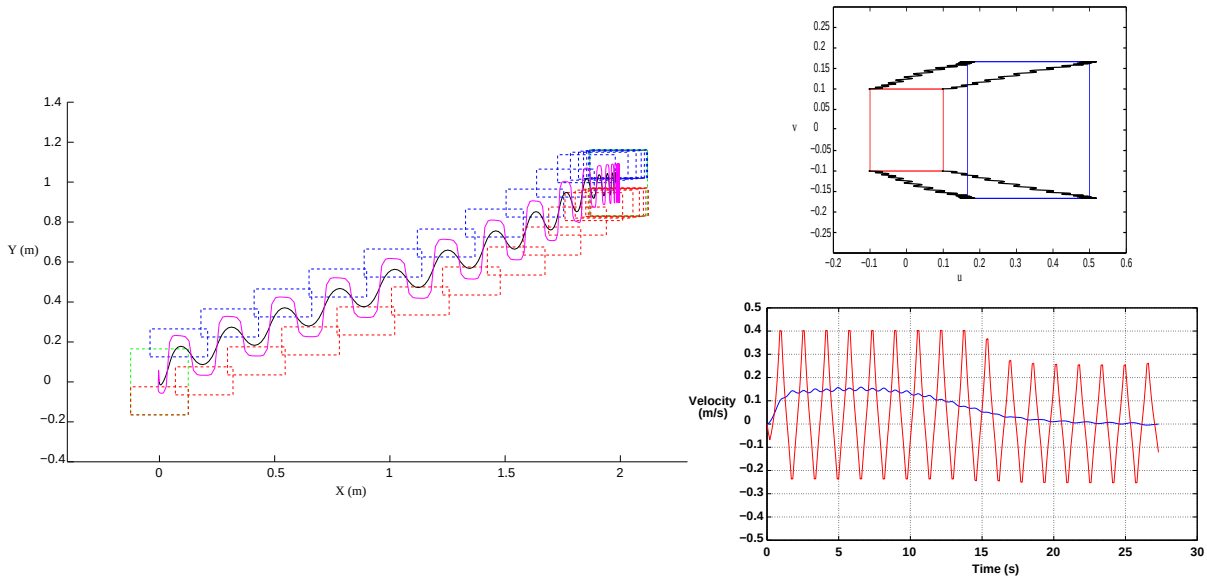

Fig. 4. On the left, we show the trajectory of the robot in the x, y plane, driven by our MPC-based coupled approach. The initial and final double stance phases appear in green. The single stance support feet appear in red (resp. blue) for the right (resp. left) foot. In pink, we depicted the CoP trajectory, which can be observed to remain safely in the support polygon, and in black the CoM trajectory. On the right, top, we depict (in black) the trajectory of the features in the image, with the initial positions in red and the desired positions in blue, for the first simulation. Finally, the evolution of the velocities is shown in the bottom. It is interesting to note the offset of the oscillatory velocity in x (in blue) and y (in red) due to the features errors.

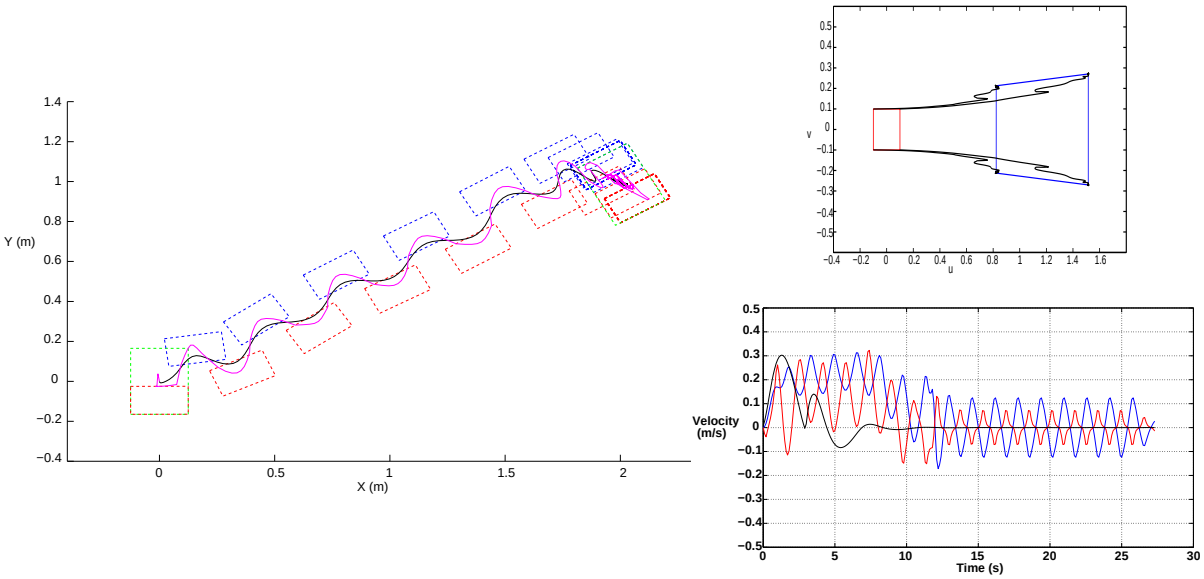

Fig. 5. The behavior of our system in the second simulation, involving rotation, with the same graphical conventions as in Fig. 4. In the velocities graph (rigth, bottom) we added the rotation velocity in black. The robot is walking in the sagittal direction most of the time, after an initial rotation. The amplitude of the oscillation of the CoM and subsequently of the features are smaller than in the first simulation. However, we can see a non-negligible component of velocity in the positive y direction, since the angle is not fully compensated. When the angle is almost fully compensated, this component disappears.

MPC is that it naturally filters out these oscillations because we minize the errors in a full cycle. It is remarkable that, in comparison with the decoupled approach [Dune et al., 2010] we do not need to model explicitly the sway motion of the robot and the resulting motion of the visual features. The system could oscillate inside the horizon, and it does, but at the end, the optimal control is taken without oscillations (Fig. 8). In Fig. 8, we only show three features errors evolution, the u component of each left (black) and right (red) lower side corners, since the upper ones are, by symmetry, the same. And we complete with a single v component (blue) for all the corners, with the same symmetry argument.

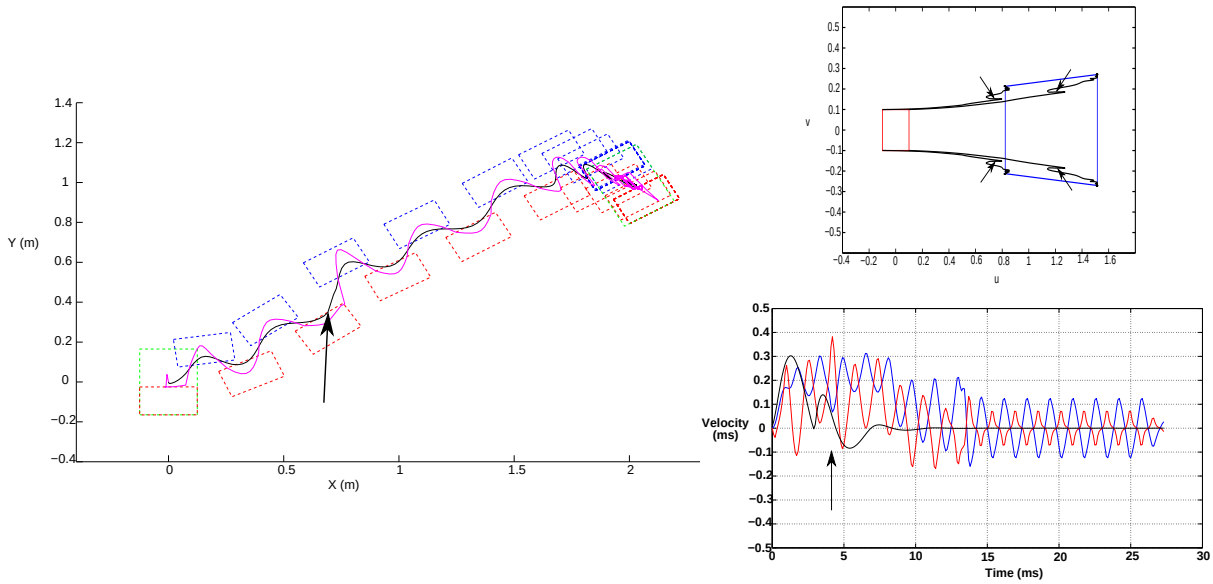

Fig. 6. The behavior of our system in the third simulation, with the same graphical conventions as in Fig. 5. The robot is following a trajectory similar to the one of Fig. 5. After a few footsteps, a strong perturbation is applied to the CoM, inducing a peak in the velocities graph (rigth, bottom). This perturbation is small in distance metric, so it is not visible in the features trajectories. The perturbation is quasi-instantly recovered.

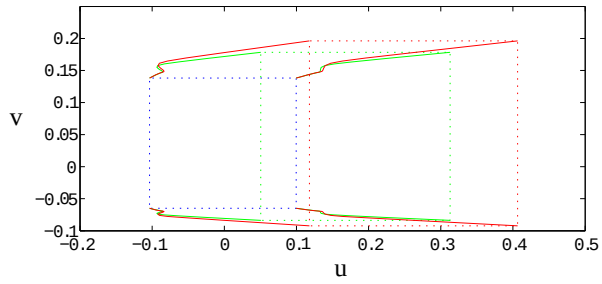

Fig. 7. In this figure, we show the trajectory of the visual features in one iteration of the QP, and compare the evolution of the features obtained by the linearization model (green lines) and features obtained by the exact non-linear model (red lines).

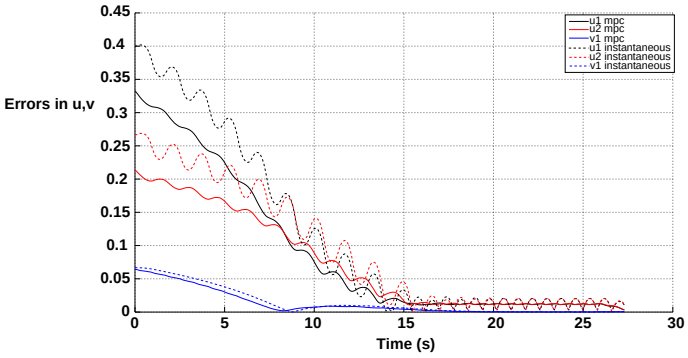

Fig. 8. Evolution of the errors for the u, v components of each feature, in the second simulation. In dashed line, we depict the instantaneous errors along time, and in solid line, we depict the errors estimated in the horizon (i.e. individual terms of Eq. 13, normalized by the size of the horizon). Observe that the sway motion of the robot induces oscillations of the u components in the instantaneous errors, and that these oscillations are not present in the errors estimated in the horizon window.

B. Influence of the weighting parameters

Depending of the weights of the QP (α_T, β_T, γ_T), we obtain different trajectories corresponding to the different priorities conveyed by each choice of parameters. For example, refer to 9. It depicts a simulation with the same objective as in Fig. 4. The difference between these two simulations is that we increased the β_T parameter, with the other parameters fixed. In Fig. 4, $\beta_T = 0.001$, the default value, and in 9, $\beta_T = 0.005$. The result is that the robot minimizes first the visual features errors, disregarding the jerks regularization term, which produces higher velocities and a globally less smooth trajectory.

Similarly, we illustrate the effect of parameter γ_T on the obtained trajectories in Fig. 10. This parameter is the weight given to the CoP centering term, making the CoP as close as possible to the footstep center. It is critical in the sense that if it takes a too low value, then the optimization may take configurations where in single support phase, the CoP is very close to the boundary of its admissible space (the support footprint), and may not converge at all. Experimentally, we have seen that for values of $\gamma_T < 0.3$, the convergence is compromised. In Fig. 10, we depict four experiments with the same goal as in Fig. 4, with increasing values of γ_T. What can be observed along these figures is the evolution of the CoP trajectories (in pink): with larger values of γ_T, the CoP trajectory tends to have shorter periods of time away from the center of the footprint. As a consequence, the steps are shorter, but the resulting trajectories are safer to be done.

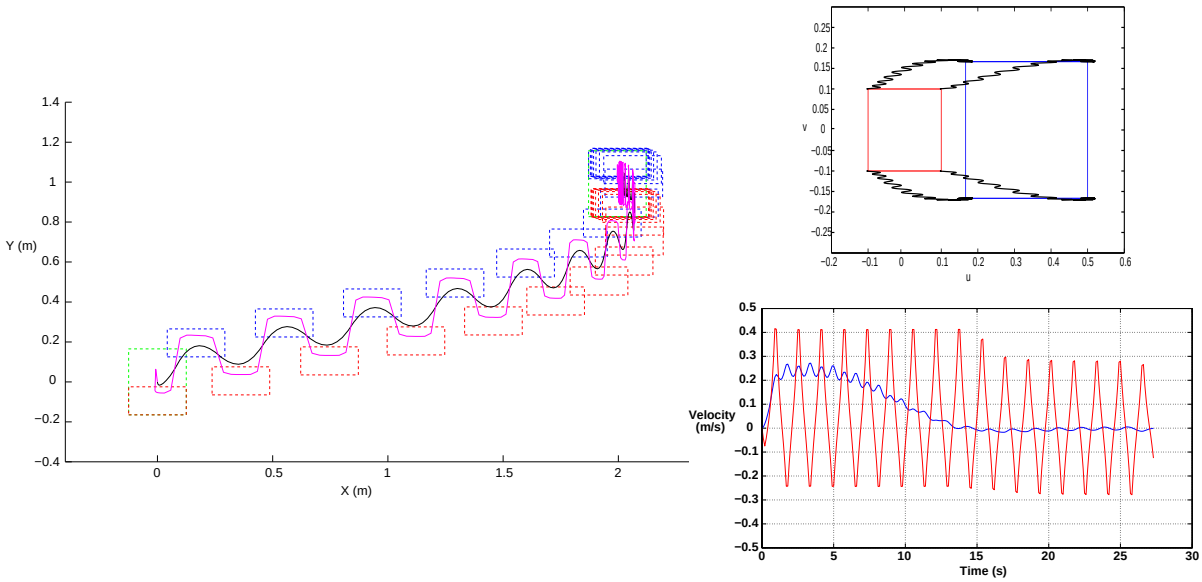

Fig. 9. The behavior of our system in a variant of the first simulation, with β_T is 5 times higher, and with the same graphical conventions as in Fig. 4. The robot is following a quite distinct trajectory, with a backwards part close to the end. As the priority is to minimize features errors, disregarding the jerks, high velocities are taken.

C. Comparisons: coupled vs. decoupled approaches

We conducted a simple experiment to illustrate better the differences between the two approaches of Section V and Section VI, and to have quantitative comparisons between them. In this experiment, the robot has simply to go three meters forward. In Fig. 11, we depict the robot trajectories and footsteps (left column) and the features trajectories (right column) for each approach. The coupled one is depicted in the upper line, the decoupled one is depicted in the lower line. We can clearly appreciate that the coupled approach converges faster. Indeed, with the coupled approach, it took 15 steps, including double supports, to reach the goal. However, with the decoupled approach, after 30 steps, the goal has still not been reached and keeps converging slowly. Here, the convergence criterion is the norm of the errors between the current features positions and the desired ones. Moreover, one can observe that, close to the goal, the features positions follow a smoother trajectory in the case of the visual predictive control than with the decoupled approach, where as a result of the immediate stepping, oscillations are visible.

In Fig. 12 left, we present the profiles of velocities performed by the robot, in both cases. It is interesting to note that in the coupled approach the amplitude of the oscillations is smaller, which is desirable. Also, when the error gets small, the classical approach slows down to have a slow convergence rate. This is a normal feature of classical visual servoing, i.e. the error evolves with an exponential decay. In the decoupled approach, this behavior is clearly present. However, in the coupled approach combining the MPC WPG with visual predictive control, this is not true anymore, see Fig. 12 right, in particular, because we take into account future information, so we converge faster. In the errors evolution we use the same symmetry argument as in Fig. 8. We should note that the u components in the image plane are theoretically the oscillatory ones from the stepping motion. In Fig. 12, we can see that the v component converges faster in the coupled approach. With the u components, there remains a small residual of the oscillation in both the coupled and decoupled approaches, which explains that the u components converge slower in the coupled approach.

VIII. Conclusion

Since the original proposal for walking generation proposed in [Kajita et al., 2003], most of the efforts in the literature have focused in dynamical balance and stability. In this paper, we have proposed a novel approach to close the robot navigation control loop within the visual servoing paradigm, by coupling tightly visual information to the Model Predictive Control formulation for walking pattern generation. This way, our online pattern generator integrates the regulation of the relative pose of 3D image features while simultaneous ensuring safety and stability for the robot, and enforcing useful visual servoing constraints, such as the visibility of the features, maximal velocities, etc. In order to keep the optimization formulation as a Quadratic Program, the perspective projection equations have been linearized around the features positions at the beginning of each pattern generator cycle. For the moment, our current approach uses 3-D information (Position-Based Visual Servoing) that strongly depends on the localization. As a future work, we wish to drop the need of 3-D information by predicting the image positions of the landmarks in terms on the velocity of the robot in the horizon (IBVS).

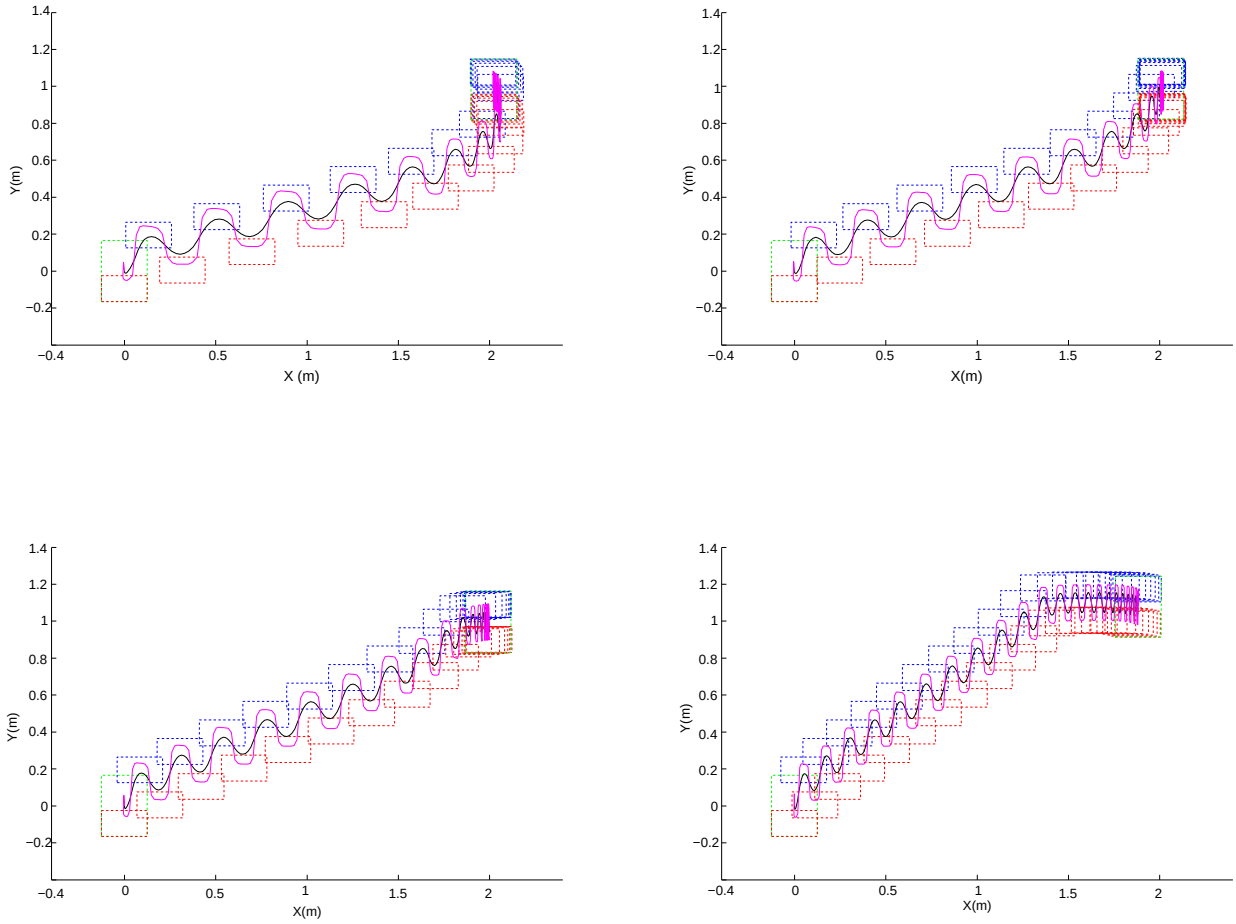

Fig. 10. Variants of the first simulation in Fig. 4, with different values of the γ_T parameter, controlling the weight given to the CoP centering term, making the CoP as close as possible to the footstep center. From left to right, and top to bottom, $\gamma_T = 0.55$, $\gamma_T = 0.75$, $\gamma_T = 1$, and $\gamma_T = 2$.

ACKNOWLEDGMENTS

The first author is supported by the grant 263150 from the Mexican National Council of Science and Technology (CONACYT). For this work the second author was supported by a grant from the RBLINK Project, Contrat ANR-08-JCJC-0075-01, from the OSEO ROMEO Project, and from the FP7-ICT-2013-10/611909 KOROIBOT project. For this work the fourth author was supported by Grant-in-Aid from the Japanese Society for the Promotion of Science (JSPS) Fellows P-09721. The authors want to thank especially Pierre-Brice Wieber and Alexander Sherikov. Special thanks to E. Marchand and F. Chaumette for providing us with Lagadic model-tracker and for their precious advice and discussion.

REFERENCES

[Allibert et al., 2010] Allibert, G., Courtial, E., and Chaumette, F. (2010). Visual servoing via nonlinear predictive control. In Chesi, G. and Hashimoto, K., editors, *Visual Servoing via Advanced Numerical Methods*, volume 401 of *Lecture Notes in Control and Information Sciences*, pages 375–393. Springer London.

[Chaumette and Hutchinson, 2006] Chaumette, F. and Hutchinson, S. (2006). Visual servo control, part i: Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4):82–90.

[Chaumette and Hutchinson, 2007] Chaumette, F. and Hutchinson, S. (2007). Visual servo control, part ii: Advanced approaches. *IEEE Robotics and Automation Magazine*, 14(1):109–118.

[Chestnutt et al., 2007] Chestnutt, J., Michel, P., Kuffner, J., and Kanade, T. (2007). Locomotion among dynamic obstacles for the honda asimo. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2572–2573.

[Coelho et al., 2001] Coelho, J., Piater, J., and Grupen, R. (2001). Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot. *Robotics and Autonomous Systems*, 37(2-3):195–217.

[Comport et al., 2006] Comport, A., Marchand, E., Pressigout, M., and Chaumette, F. (2006). Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *IEEE Trans. on Visualization and Computer Graphics*, 12(4):615–628.

[Courty et al., 2001] Courty, N., Marchand, E., and Arnaldi, B. (2001). Through-the-eyes control of a virtual humanoïd. In Ko, H.-S., editor, *IEEE Computer Animation 2001*, pages 74–83, Seoul, Korea.

[Dune et al., 2010] Dune, C., Herdt, A., O., S., P.-B., W., E., Y., and K., Y. (2010). Cancelling the sway motion of dynamic walking in visual servoing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS10)*.

[Gutmann et al., 2008] Gutmann, J.-S., Fukuchi, M., and Fujita, M. (2008).

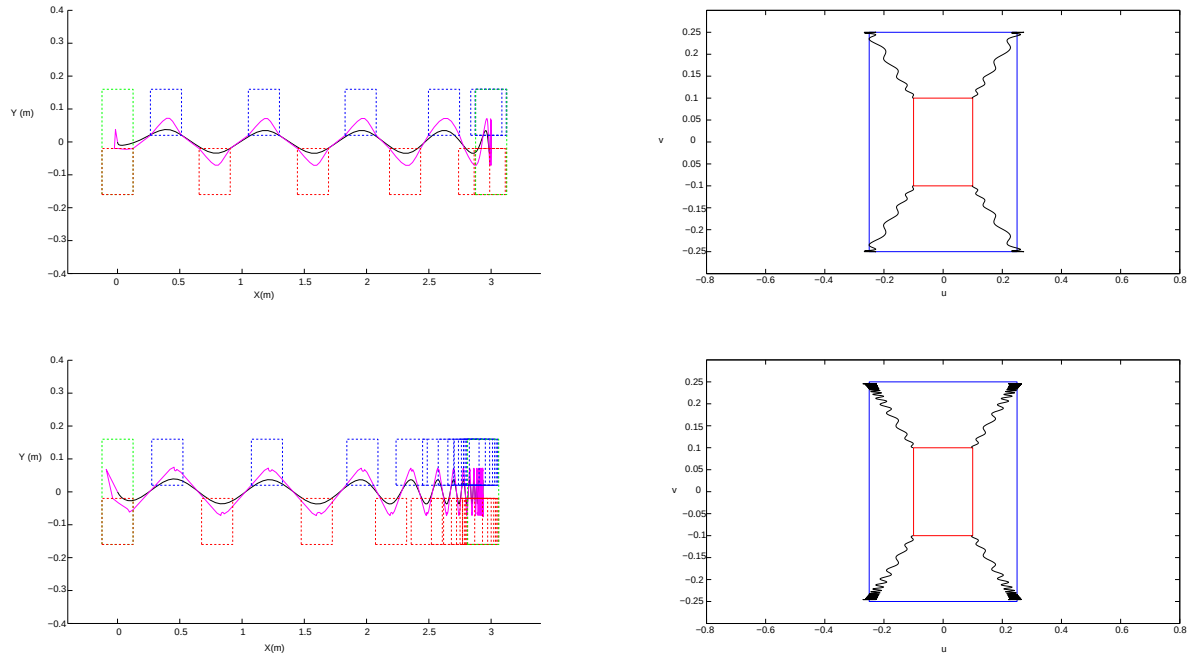

Fig. 11. Comparison between the coupled (upper part) and decoupled approaches (lower part). On the left side, we depict the robot trajectory with the footsteps (red and blue), and the CoM (black) and CoP (magenta) trajectories. On the right side, we depict the evolution of the instantaneous features positions during the experiment (from the red rectangle, at the beginning of the simulation, to the blue one, at the end).

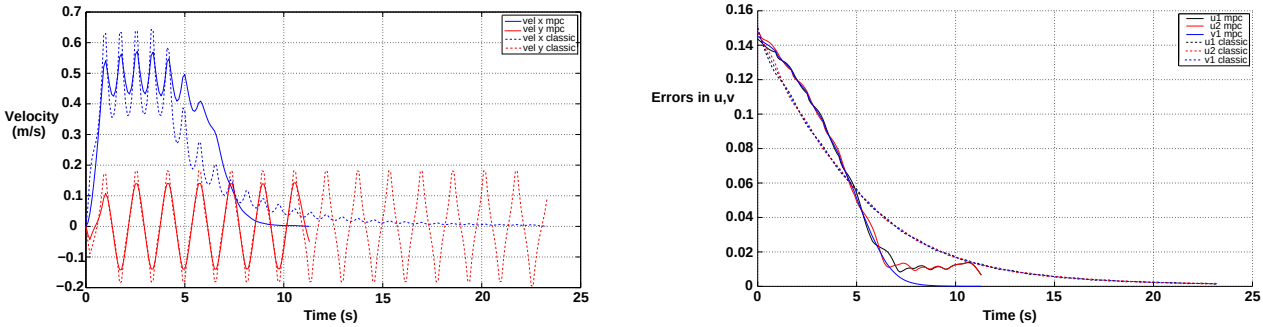

Fig. 12. Left, comparison of velocity profiles in the x (blue) and y (red) axes, for the decoupled (dashed line) and coupled (solid line) approaches. Right, comparison of the features errors evolution.

3d perception and environment map generation for humanoid robot navigation. *Int. Journal of Robotics Research*, 27(10):1117–1134.

[Herdt et al., 2010a] Herdt, A., Holger, D., Wieber, P., Dimitrov, D., Mombaur, K., and Moritz, D. (2010a). Online walking motion generation with automatic foot step placement. *Advanced Robotics*, 24(5-6):719–737.

[Herdt et al., 2010b] Herdt, A., Perrin, N., and Wieber, P.-B. (2010b). Walking without thinking about it. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS10)*.

[Kajita et al., 2003] Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., and Yokoi, K. (2003). Biped walking pattern generation by using preview control of zero-moment point. In *IEEE/RAS International Conference on Robotics and Automation*, pages 1620–1626.

[Lorch et al., 2002] Lorch, O., Albert, A., Denk, J., Gerecke, M., Cupec, R., Seara, J., Gerth, W., and Schmidt, G. (2002). Experiments in vision-guided biped walking. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2484–2490.

[Mansard et al., 2007] Mansard, N., Stasse, O., Chaumette, F., and Yokoi, K. (2007). Visually-guided grasping while walking on a humanoid robot. In *IEEE Int. Conf. on Robotics and Automation (ICRA07)*, pages 3041–3047, Roma, Italy.

[Michel et al., 2007] Michel, P., Chestnutt, J., Kagami, S., Nishiwaki, K., Kuffner, J., and Kanade, T. (2007). Gpu-accelerated real-time 3d tracking for humanoid locomotion and stair climbing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 463–469.

[Morisawa et al., 2007] Morisawa, M., Harada, K., Kajita, S., Nakaoka, S., Fujiwara, K., Kanehiro, F., Kaneko, K., and Hirukawa, H. (2007). Experimentation of humanoid walking allowing immediate modification of foot place based on analytical solution. In *IEEE Int. Conf. on Robotics and Automation*, pages 3989–3994.

[Nishiwaki and Kagami, 2009] Nishiwaki, K. and Kagami, S. (2009). Online walking control system for humanoids with short cycle pattern generation. *Int. Journal of Robotics Research (IJRR)*, 28:729–742.

[Perrin et al., 2010] Perrin, N., Stasse, S., Lamiraux, F., and Yoshida, E. (2010). Approximation of feasibility tests for reactive walk on hrp-2. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 4243–4248.

[Taylor and Kleeman, 2001] Taylor, G. and Kleeman, L. (2001). Flexible self-calibrated visual servoing for a humanoid robot. In *Australian Conf. on Robotics and Automation (ACRA2001)*, pages 79–84.

[Wieber, 2002] Wieber, P.-B. (2002). On the stability of walking systems. In *Int. Workshop on Humanoid and Human Friendly Robotics (2002)*.