Centro de Investigación en Matemáticas, A.C.

CIMAT

TESIS

para obtener el grado de

Maestría en Ciencias con Especialidad en Computación y Matemáticas Industriales

presentada el 23 de Noviembre de 2013

Vision-Based Locomotion for Humanoid Robots

Mauricio Josafat García Vázquez

Jurado

Dr. Jean-Bernard Hayet	Co-Asesor
Dr. Olivier Stasse	Co-Asesor
Dr. Héctor Becerra	Sinodal
Dr. Claudia Esteves	Sinodal

Agradecimientos

Quiero agradecer a Jean-Bernard, quien con su confianza e inspiración me impulsó a buscar grandes cosas, gracias por brindarme tanto apoyo durante estos años en CIMAT. Gracias Olivier que junto con Jean-Bernard me motivaron y aconsejaron, ustedes son las mentes detrás de este trabajo. Merci beaucoup Jean-Bernard et Olivier.

Gracias al resto del equipo de la Robocup en CIMAT, Claudia, Noé, Paco, que fueron un gran apoyo durante mi estancia. Gracias al CIMAT, que me abrió sus puertas y me hizo sentir parte de una gran familia. Gracias al LAAS-CNRS y al equipo Gepetto que me recibieron con mucha cordialidad durante mi estancia. Gracias a Oscar Ramos, Nicolas Mansard, Jean-Paul Laumond, Claire Dune, Philippe Soueres. Directa o indirectamente me ayudaron con contribuciones, consejos y discusiones.

Gracias a mi familia, a mi esposa Martha por tanto apoyo, por su paciencia, amor y compañía, gracias por estar conmigo en todo momento y brindarme tanta fortaleza. Gracias a mi madre, padre y hermanos, que con su cariño me motivaron a seguir adelante.

Contents

1

Introduction

This thesis was made with the spirit of contributing to the autonomy of humanoid robots; more precisely in the development of behaviors based on computer vision. Even with the tremendous expansion of humanoid robotics in the last years, the link between perception and motion generation is still not well developed. Hundreds of works cover both fields separately. In the perception side, computer vision is up to this day the main technique in humanoid robotics.

On the robotic side, there is an increasing interest in the development of humanoid robotics. Governments and companies such as Honda, Boston Dynamics and Aldebaran Robotics have spent a lot of money in these fields. As a result, amazing demonstrations in humanoid motion generation have been done recently. However, the interaction with the environment is limited in most of the systems.

The design of humanoid robots has been done thinking mainly in human-friendly environments, i.e. unstructured and dynamic environments, where objects move outside the robots control. Therefore, when specific tasks have to be completed, these robots have to be able to perceive and react to environmental

changes. Visual sensors can help them to reach this objective, by allowing to build local representations of the robot surroundings, and to adapt their behavior in consequence. Most of the existing humanoid platforms are equipped with video cameras, providing the robots with rich information (geometry, texture, color...) without adding so much weight and size, at a rather low cost. Moreover, the use of embedded cameras is attractive because it avoids equipping the environment with additional external sensors. Such an embedding increases greatly the level of autonomy of the robotic platform. However, to interpret the data generated from the camera of a humanoid robot is still problematic, with a huge amount of undergoing work in the computer vision community, as, in general, the image quality in humanoid robots is quite poor: blurring effects or vibrations due to the walk may make visual tasks such as localization and tracking really challenging.

Early works on humanoid locomotion have assumed that the robot path is defined before computing the current 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, given the previously computed CoM trajectory, and the sequence of footsteps to follow. This perception-decision-action loop has proven to be fast enough to realize impressive demonstrations for stair-climbing and obstacle avoidance. Our focus in this thesis 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.

A general diagram of the workflow in humanoid robots is depicted in Fig. 1.1. In general we use computer vision to extract information from the environment (mapping, localization, modeling). With this model, we can plan the motion using rules given by a cognitive process. With this plan, the robot performs the dynamic walking which leads to the whole body motion. Computer vision requires high computational load to extract high level information from the scene; motion generation on the other side, requires fast control loops to be dynamically stable.

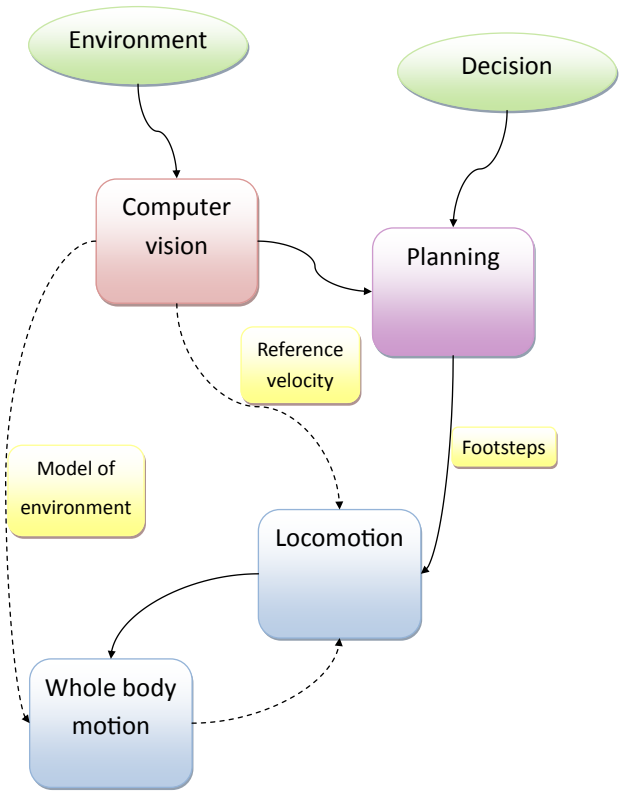


Figure 1.1: Workflow of a humanoid robot executing motion tasks using perception.

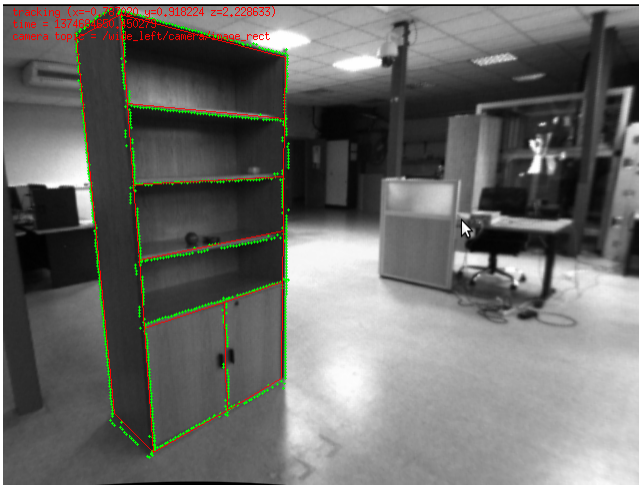
In this thesis we address the problem of the link between vision and motion generation, as illustrated in Fig 1.2. Building upon the work of several authors through the years, our main contributions are,

- The full integration of a visual servoing scheme within the walking motion generation using Linear Model Predictive Control (MPC);
- The integration of traditional motion planning approaches and on-line locomotion generation algorithms;
- The visual reconstruction of 3D models of the scene in front of the robot to

be used by an inverse-dynamics based approach to walk on rough terrain;

- And finally, the validation in simulation of the proposals.

(a)



(b)

Figure 1.2: Typical application of computer vision and control, visual servoing.

The rest of this manuscript is organized as follows. In chapter 2, we state the problem and briefly recall the work that has been done on it. In chapter 3, we introduce the techniques that most of the current walking motion generation schemes for humanoid robots are based on. The one proposed in Kajita et al. in 2003 [Kajita et al., 2003] that focuses in the trajectory of the center of mass to generate balanced and stable motions and the one proposed by Herdt et al. in

2010 [Herdt et al., 2010a] in which a reference velocity is tracked. In chapter 4 we present an approach to introduce visual information in the walking pattern generator for humanoid robots. We make use of visual servoing with Model Predictive Control (MPC), which is combined with the walking motion generator. Since visual servoing with MPC is in general a nonlinear optimization problem, we propose a linearization scheme in order to keep it as a Quadratic Program (QP) and introduce it within the pattern generator. In chapter 5, we propose a method for reactive walking allowing visual servoing and adaptation of footsteps trajectories in real-time. This is done by building upon recent advances in the fields of optimal control for a walking pattern generator [Herdt et al., 2010a] and planning for a nonholomic robot with field-of-view constraints [Salaris et al., 2010]. In chapter 6, we present a 3D reconstruction system of the ground in front of the robot. This model allows to know the ground structure where the swinging foot is going to step on to an inverse dynamics control scheme. Finally, in chapter 7, we present the final discussion and future work.

2
Related work

Recently, very efficient control systems for humanoid robots walking generation have been proposed. They use dynamical balance criteria such as the Center of Pressure (CoP) and may be very reactive since foot step placement can be computed online [Herdt et al., 2010a]. In this methodology, the CoM and CoP trajectories and the footstep positions are computed automatically. This result is fed to a whole-body controller to deal coherently with the three constraints. In that work, the notion of footsteps disappears, allowing the user to provide a reference velocity as input to the pattern generator. Moreover, because the range of footsteps is explored by a guided search in the space of the whole-body controller transitions, a large set of possible footsteps is available in real-time.

Most of humanoid walking techniques are based on Linear Model Predictive Control (LMPC). LMPC previews the behavior of the system if one applies a sequence of controls and it allows us to estimate the optimal control sequence for a given horizon. In the next iteration, one applies the first next optimal control. Remarkably, LMPC can be expressed as a Quadratic Program (QP), that is the minimization of quadratic errors subject to a set of linear equality and inequality

constraints. Handling explicitly the constraints in the QP is one of the main advantages of this formulation. Furthermore, there are very efficient techniques to solve QPs.

The next step would be to close the control loop and use sensors information as feedback, e.g. in positioning tasks. In general humanoid walking control assume that the robot foot steps are defined before computing the current joint control to realize it. They generally follow a perception-decision-action scheme: footsteps on time horizon are defined upon sensors information on the environment, from the footsteps center of mass trajectories (CoM) are computed to respect balance constraints. Legs control 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., 2007b; Michel et al., 2007; Gutmann et al., 2008].

On the other hand, for reactive positioning tasks, visual servoing techniques have proven to be useful [Chaumette and Hutchinson, 2006, 2007]. Dune et al. [Dune et al., 2010] proposed a visual control scheme for humanoid robots using walking pattern generation as a black box. The desired velocity of the Center of Mass (CoM) is computed using visual servoing and it is used as a reference in the pattern generator. The main disadvantage with this approach is that the visual servoing scheme and the pattern generator are completely decoupled so that the visual information is not directly feeding the pattern generator.

Visual servoing has been successfully applied in Model Predictive Control schemes [Allibert et al., 2010]. The main problem is that the dynamics of the camera and the projection itself are nonlinear functions for which non-linear programming are required. However, this may be time consuming and hence does not fit into an online walking pattern generator.

Parallely, research has been led towards planning trajectories that are appropriate for humanoid robots walking in cluttered environments, with different types of restrictions (e.g. [Chestnutt et al., 2005; Hayet et al., 2012]). From the seminal work of [Chestnutt et al., 2005], a main trend of research in footstep planning has been to consider a limited set of known actions (quite often footsteps) and transitions, and to find an optimal path over them. The use of a fixed-set-of-actions approach can be limiting, as it may produce unnecessary motions near the obstacles [Bourgeot et al., 2002], while not usually dealing with the problem

of robust perturbation rejection.

A vast amount of work exist to lessen the amount of movements generated around the obstacles. Chestnutt et al. proposed in [Chestnutt et al., 2007a] an adaptation mechanism to search around the set of transitions. More recently, Hornung et al. [Hornung and Bennewitz, 2012], proposed a method to deal with highly dynamical environments by keeping both, accurate short-term goals and rough long-term goals. As doing this may lead to local optimum, the authors propose a method to automatically adapt the set of actions according to the environment traversability characteristics.

Regarding the problem of robust perturbation rejection, several advances have also been done. They can be divided into three strategies: (1) ankle-foot stabilization, (2) whole-body stabilization, and (3) footstep generation. Using the capture-point or the Center-of-Pressure (CoP) as an indicator of stability, one can switch between a Finite-State-Machine strategy [Nishiwaki and Kagami, 2009] and a learned strategy [Yi et al., 2011].

From an application point of view, however, it is very difficult to decouple planning and control from each other. Planning is needed to avoid local optimum and control is needed to reject disturbances and adapt to modeling errors. Some planning methods try to account for the motion and control capabilities of the humanoid robot by using inverse kinematics [Kanoun et al., 2011]. Despite real-time implementation [Dang et al., 2011], this method suffers from local minima in planning footsteps.

Building upon the work of Herdt et al. [Herdt et al., 2010a] and Allibert et al. [Allibert et al., 2010], the first contribution of this thesis is the full integration of a visual servoing scheme within the walking motion generation, through linearization into LMPC. As in [Vahrenkamp et al., 2009], the second contribution of this thesis is to use a planning approach integrating a constraint given by a task (e.g. visual-servoing). However, here, we modify the walking controller in such a way to use the planner as a generator of a full vector field that provides new local solutions from any given configuration and not only as a provider of single reference trajectories. For the third contribution of this thesis, the dense reconstruction of the floor surface in front of the robot, we rely on a real-time approach similar to KinectFusion algorithm proposed in [Newcombe et al., 2011].

3 Locomotion

In this chapter we introduce the techniques most commonly used in biped walking generation, the Linearized Inverted Pendulum Model and Kajita's Preview Control. These techniques are widely used and they have become a standard in humanoid walking control. In the next chapters, we will work upon these techniques. The reader is kindly encouraged to read the original papers to have a deeper understanding of these models.

3.1 Models of Biped Walking and Kajita's Preview Control

Most of the works in biped walking are based on the simplified model proposed by Kajita et al. in 1992 [Kajita et al., 1992]. This model simplifies the multi-body nature of the robot considering it as a single mass moving as an inverted pendulum. This mass is moving in the x, y plane. It also constraints the trajectory of the CoM to a horizontal plane, as depicted in Fig. 3.1.

The dynamics of this model is given by,

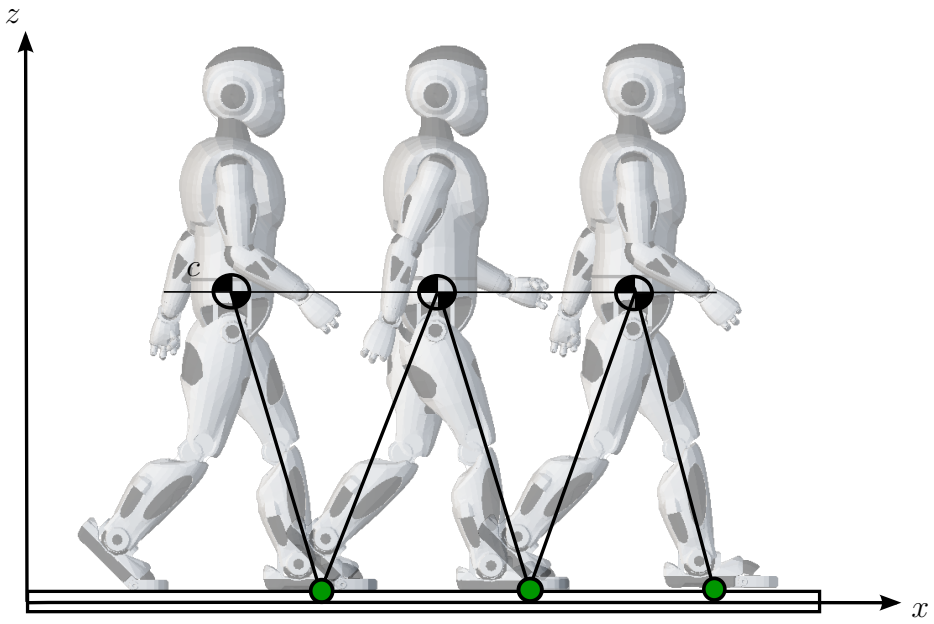


Figure 3.1: Humanoid walking modeled as an inverted pendulum. The center of mass is constrained to a horizontal plane. The ZMP corresponds to the point in the sole in which the momentum generated by the inertia and gravity is countered with the momentum generated by the contact forces of the foot with the ground.

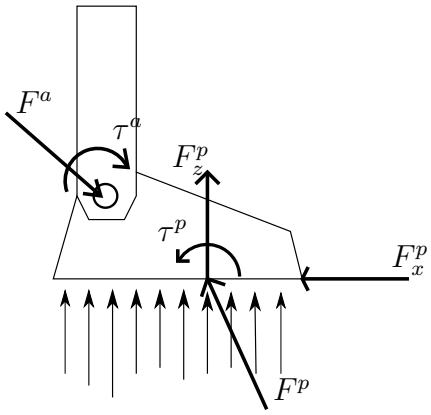


Figure 3.2: The resultant forces and torques of the robot in the foot are F^a and τ^a respectively. It corresponds the reactions in the contact with the floor. The resultant of those reactions is F^p and its corresponding torque τ^p. The z component of this resultant F_z^p corresponds to the vertical reaction of the floor. The x and y components correspond to friction forces and they avoid sliding.

$$\tau_x^a = mgc_y - mc_z\ddot{c}_y, \tag{3.1}$$
$$\tau_y^a = mgc_x - mc_z\ddot{c}_x \tag{3.2}$$

where (c_x, c_y, c_z) is the 3D position of the CoM, m its mass, and τ_x^a, τ_y^a the torques produced by the inertia and gravity (g) in the base of the pendulum. We can note that the equation in the y axis is completely equivalent to the x axis.

In the contact of the sole with the ground, there exists the countering force F^p. The x, y components of this force correspond to friction and they avoid sliding. The z component supports the weight of the pendulum and is equal to $F_z^p = mg$, as depicted in Fig. 3.2.

The total torques in the base of the pendulum are given by,

$$\tau_x^{total} = \tau_x^p + \tau_x^a = F_z^p z_x^p + \tau_x^a,$$
$$\tau_y^{total} = \tau_y^p + \tau_y^a = F_z^p z_y^p + \tau_y^a$$

where (z_x^p, z_y^p) is the acting point of F^p.

The locomotion is dynamically balanced if the contact forces of the feet

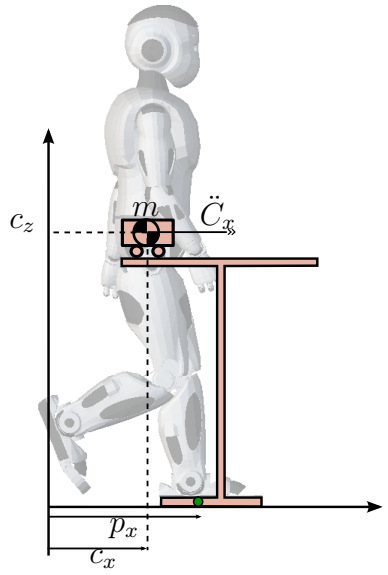


Figure 3.3: Equivalent model for humanoid walking. Statically the table is not balanced, dynamically inertia forces balance the table.

with the ground counter the forces due to the inertia and gravity. The Zero Moment Point (ZMP) (z_x, z_y) corresponds to the point on the ground where the total moment in the base of the pendulum is equal to zero [Vukobratovic and Stepanenko, 1972; Vukobratovic and Borovac, 2004],

$$0 = mgz_x + \tau_x^a, \tag{3.3}$$
$$0 = mgz_y + \tau_y^a. \tag{3.4}$$

Substituting the former equations in equation 3.1, we derive the ZMP equations,

$$Z = \begin{bmatrix} z_x \\ z_y \end{bmatrix} = \begin{bmatrix} c_x - \frac{c_z}{g}\ddot{c}_x \\ c_y - \frac{c_z}{g}\ddot{c}_y \end{bmatrix}. \tag{3.5}$$

This model is equivalent to the cart-table model. See Fig. 3.3. This model simulates the dynamics of the robot as a running cart above a table. The cart is in a position where it is not statically stable, however, dynamically, if the cart accelerates in an appropriate way, inertia forces will keep it in balance.

We take the control variables as the time derivative of the horizontal accelerations of the CoM, i.e. the jerks,

$$u_x \stackrel{\text{def}}{=} \frac{d}{dt}\ddot{c}_x = \dddot{c}_x.$$

From the work of [Kajita et al., 2003], if we suppose that the trajectory has periodic piece-wise constant jerks on a time interval T, for discrete time k, we can express the CoM dynamics in the x-axis as,

$$c_x(k+1) = c_x(k) + \dot{c}_x(k)T + \ddot{c}_x(k)T^2/2 + \dddot{c}_x(k)T^3/6.$$

Let us define

$$\hat{c}_x(k) \equiv \begin{bmatrix} c_x(k) \\ \dot{c}_x(k) \\ \ddot{c}_x(k) \end{bmatrix}$$

the state of the robot, defined by its position, velocity and acceleration at time k.

We can express the state of the robot at time $k+1$ in terms of the state k plus the control variable,

$$\begin{pmatrix} c_x(k+1) \\ \dot{c}_x(k+1) \\ \ddot{c}_x(k+1) \end{pmatrix} = \begin{pmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_x(k) \\ \dot{c}_x(k) \\ \ddot{c}_x(k) \end{pmatrix} + \begin{pmatrix} T^3/6 \\ T^2/2 \\ T \end{pmatrix} u_x(k). \tag{3.6}$$

Finally, using Eq. 3.5 and Eq. 3.6 we have the basic equations of biped locomotion,

$$\begin{cases} \hat{c}_x(k+1) &= A\hat{c}_x(k) + Bu_x(k) \\ z_x(k) &= C\hat{c}_x(k) \end{cases}, \tag{3.7}$$

with,

$$A = \begin{pmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{pmatrix}, B = \begin{pmatrix} T^3/6 \\ T^2/2 \\ T \end{pmatrix} \text{ and } C = \begin{pmatrix} 1\ 0\ \dfrac{c_z}{g} \end{pmatrix}.$$

To solve this control system efficiently, Kajita et al. also proposed to use Model Predictive Control (MPC). MPC takes into account future information by

previewing the behavior of the system for a given horizon. Then, an optimization problem is formulated using a performance index,

$$\min_{U(k)} \sum_{i=k}^{k+N-1} \frac{1}{2} Q(z_x(i+1) - z_x^{ref}(i+1))^2 + \frac{1}{2} R \dddot{c}_x^2(i), \tag{3.8}$$

where $U(k) = [u(k) \dots u(k+N-1)]$ is the sequence of the next N controls, $z_x^{ref}(i)$ is a reference ZMP given by a previous footstep planning for instance, Q and R are constants and N is the size of the horizon. The first term of the optimization problem contributes to a minimization of the squared error of the ZMP and the reference ZMP. The second term minimizes the jerks, so we can have smooth trajectories of the CoM.

The optimal controller is given by,

$$\dddot{c}_x(k) = -K_1 \sum_{i=0}^{k} e(i) - K_2 c_x(k) - \sum_{j=1}^{N} K_p(j) z_x^{ref}(k+j) \tag{3.9}$$

with $e(i) = z_x(i) - z_x^{ref}(i)$ being the ZMP error w.r.t. the reference ZMP, and K_1, K_2 and $K_p(j)$ are gains.

Kajita et al. also proposed a second step to correct the effects of the inverted pendulum simplification. This second step takes into account the multi-body dynamics and is re-injected to have better CoM and ZMP trajectories.

3.2 Quadratic Programming and Automatic footstep placement

Wieber in [Wieber, 2006] proposed to reformulate the preview control problem as a Quadratic Programming Problem. By applying recursively the dynamics of Eqs. 3.7, we can express the position, velocity and acceleration of the CoM in terms of the initial state $\hat{c}_x(k)$ and the sequence of jerks $\dddot{C}_x(k) \stackrel{\text{def}}{=} [\dddot{c}_x(k), \dddot{c}_x(k+1), ..., \dddot{c}_x(k+N-1)]^\top$,

$$C_x(k+1) \stackrel{\text{def}}{=} \begin{pmatrix} c_x(k+1) \\ \vdots \\ c_x(k+N-1) \end{pmatrix} = S_p \hat{c}_x(k) + U_p \dddot{C}_x(k), \tag{3.10}$$

$$\dot{C}_x(k+1) \stackrel{\text{def}}{=} \begin{pmatrix} \dot{c}_x(k+1) \\ \vdots \\ \dot{c}_x(k+N-1) \end{pmatrix} = S_v \hat{c}_x(k) + U_v \dddot{C}_x(k), \tag{3.11}$$

similar expressions can be obtained for the y component. We can also express the ZMP trajectory,

$$Z_x(k+1) \stackrel{\text{def}}{=} \begin{pmatrix} z_x(k+1) \\ \vdots \\ z_x(k+N-1) \end{pmatrix} = S_z \hat{c}_x(k) + U_z \dddot{C}_x(k). \tag{3.12}$$

With the matrices $S_p, S_v, S_z \in \mathbb{R}^{N\times 3}$ and $U_p, U_v, U_z \in \mathbb{R}^{N\times N}$ defined as,

$$S_p = \begin{bmatrix} 1 & T & T^2/2 \\ \vdots & \vdots & \vdots \\ 1 & NT & N^2T^2 \end{bmatrix}, \qquad U_p = \begin{bmatrix} T^3/6 & 0 & 0 \\ \vdots & \ddots & 0 \\ (1+3N+3N^2)T^3/6 & \cdots & T^3/6 \end{bmatrix},$$

$$S_v = \begin{bmatrix} 0 & 1 & T \\ \vdots & \vdots & \vdots \\ 0 & 1 & NT \end{bmatrix}, \qquad U_v = \begin{bmatrix} T^2/2 & 0 & 0 \\ \vdots & \ddots & 0 \\ (1+2N)T^2/2 & \cdots & T^2/2 \end{bmatrix},$$

$$S_z = \begin{bmatrix} 1 & T & \frac{T^2}{2} - \frac{z^c}{g} \\ \vdots & \vdots & \vdots \\ 1 & NT & \frac{N^2T^2}{2} - \frac{z^c}{g} \end{bmatrix},$$

$$U_z = \begin{bmatrix} \frac{T^3}{6} - \frac{Tz^c}{g} & 0 & 0 \\ \vdots & \ddots & \vdots \\ [1+3(N-1)+3(N-1)^2]\frac{T^3}{6} - \frac{Tz^c}{g} & \cdots & \frac{T^3}{6} - \frac{Tz^c}{g} \end{bmatrix}.$$

This allows us to rewrite the optimization problem as,

$$\min_{U(k)} \ \frac{1}{2} R \|\dddot{C}(k)\|^2 + \frac{1}{2} Q \|Z_x(k+1) - Z_x^{ref}(k+1)\|^2,$$

with the same interpretations as in Eq. 3.8.

This optimization problem has the analytical solution,

$$\dddot{C}_x(k) = -(U_z^\top U_z + \frac{R}{Q} I_{N\times N})^{-1} U_z^\top (S_z \hat{c}_x(k) - Z_x^{ref}(k+1)). \tag{3.13}$$

Wieber showed that with this proposal the system is able to reject strong perturbations.

As an evolution of this work, where the footsteps positions (and, correspondingly, the ZMP reference) are fed to the pattern generation, the work of [Herdt et al., 2010a] introduced automatic footstep placement, i.e. managed the footsteps as free variables in the optimization problem and not as inputs. This reduced the necessary input to a simple stack of reference velocities $(\dot{C}_x^{ref}(k+1), \dot{C}_y^{ref}(k+1))$. This leads to the optimization problem,

$$\begin{aligned} \min_{U(k)} \quad & \frac{\alpha}{2} \left\|\dddot{C}_x(k)\right\|^2 + \frac{\alpha}{2} \left\|\dddot{C}_y(k)\right\|^2 \\ & + \frac{\beta}{2} \left\|\dot{C}_x(k+1) - \dot{C}_x^{ref}(k+1)\right\|^2 + \frac{\beta}{2} \left\|\dot{C}_y(k+1) - \dot{C}_y^{ref}(k+1)\right\|^2 \\ & + \frac{\gamma}{2} \left\|Z_x(k+1) - Z_x^{ref}(k+1)\right\|^2 + \frac{\gamma}{2} \left\|Z_y(k+1) - Z_y^{ref}(k+1)\right\|^2 \end{aligned} \tag{3.14}$$

with α, β, γ being constants that indicate the weight of each term in the optimization problem. The first two terms correspond to the minimization of the jerk, the next two terms correspond to a tracking of a reference velocity and the final ones correspond to a tracking of a reference ZMP.

The reference ZMP is defined as,

$$\begin{aligned} Z_x^{ref}(k+1) &= V_c \hat{F}_x(k) + V F_x(k) \\ Z_y^{ref}(k+1) &= V_c \hat{F}_y(k) + V F_y(k), \end{aligned} \tag{3.15}$$

with $\hat{F}_x(k)$, $\hat{F}_y(k)$ being the current position of the foot on the ground.

The variables to optimize are

$$U(k) \stackrel{\text{def}}{=} \begin{pmatrix} \dddot{C}_x(k) \\ F_x(k) \\ \dddot{C}_y(k) \\ F_y(k) \end{pmatrix},$$

in which $F_x(k)$ and $F_y(k)$ are the next footstep positions in the horizon. In Eq. 3.15 we set the reference ZMP to the middle of the support foot. This way, the reference ZMP is not fixed in advanced but is permanently recomputed from the feet position decided by the algorithm. Finally

$$V_c = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad V = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \\ 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \\ 0 & 1 & 0 & \cdots & 0 \end{bmatrix},$$

are selection matrices that indicate which sampling time falls in which step.

As the terms in the optimization problem defined in Eqs. 3.10-3.12 and 3.15 are linear in the variables to optimize, the problem can be written as a Quadratic Program (QP),

$$\min_{U(k)} \frac{1}{2} U(k)^\top Q(k) U(k) + p(k)^\top U(k), \tag{3.16}$$

under linear constraints arising, among others, from the inclusion of the reference ZMP inside the support polygon [Herdt et al., 2010a].

For instance, during the single support phase (in which we have a support foot and a flying one), the constraint ensuring that the ZMP remains inside the support polygon is expressed as:

$$\begin{bmatrix} d_x(\theta) & d_y(\theta) \end{bmatrix} \begin{bmatrix} z_x - f_x \\ z_y - f_y \end{bmatrix} \leq b(\theta) \tag{3.17}$$

where (f_x, f_y) is the foot position, θ is its orientation, $d_x(\theta)$, $d_y(\theta)$ are column vectors containing the x, y coordinates of the normal vectors to the feet edges, and $b(\theta)$ is the column vector containing their position with a security margin. For the double support phase, Herdt et al. chose to satisfy the constraint of the reference ZMP at the sampling time kT, and given that the double support phase is chosen to be T long (0.1s for the double support and 0.7 for the single support), no samplings fall strictly in the double support, so they just consider single support constraint in the reference ZMP. This assumption appears to be reasonable enough to generate stable motions as they showed in their experiments.

4

Vision-Based Control of Humanoid Walking

We have introduced the mathematical foundations behind humanoid walking. In this chapter we will discuss the applicability of controlling the robot trajectory using visual servoing, as depicted in Fig. 4.1. We present a novel approach for using a visual servoing scheme to control the dynamic walk of a humanoid robot. Here, the online information given by an on-board camera is used to drive the robot towards a specific goal, in a visual servoing scheme. This closed loop approach allows the system to react to changes in its environment and to adapt to modeling errors. Our work is built upon the reactive pattern generator of [Herdt et al., 2010a] that we presented at the end of Chapter 3, which modifies footsteps, center of mass and center of pressure trajectories to track a reference velocity. We compare our method to another which does not use Model Predictive Control in the visual servoing. This alternative method, proposed by Dune et al. [Dune et al., 2010] outputs, in a first stage, a reference velocity directly given by the visual servoing control law and then, in a second stage, this reference velocity is

introduced in the pattern generator to produce the appropriate walking motion (hence, our term of "decoupled"). The coupled approach proposed here uses a Model Predictive Control scheme to introduce the visual error terms directly within the pattern generator. This allows to avoid a number of limitations (e.g. that visual constraints cannot be introduced directly inside the locomotion controller or that the camera motion has to be accounted separately) that appear in the decoupled method. In this work, both approaches are compared numerically and validated in simulation.

4.1 Visual servoing

Visual servoing aims at controlling the motion of a robot equipped with a camera, by minimizing errors between observed features and their corresponding reference features, [Chaumette and Hutchinson, 2006, 2007]. The nature of the features differentiates schemes of visual servoing: Image based visual servoing (IBVS) uses only image features; Position based visual servoing (PBVS) uses the 3-D pose(s) of object(s) of interest. In any of these cases of feedback, one may use a velocity controller such as

$$\mathrm{v}^c = -\lambda L_e^+ e,$$

where $e = s - s^*$ is the vector of errors between the current features s and the desired ones s^*, v^c is the velocity of the camera (the control variable), and L_e^+ is the Moore-Penrose pseudo-inverse of the interaction matrix L_e, that is, the matrix relating the velocity of the features and the velocity of the camera.

We will now describe two approaches for introducing visual control in the described pattern generation: the first one (in Section 4.2) is a decoupled one, proposed by Dune et al. [Dune et al., 2010], where the visual servoing is used to determine the reference velocity in Eq. 3.14; the second one (Section 4.3) modifies Eq. 3.14 to replace the reference velocity term by a new term directly related to the decreasing of the visual errors.

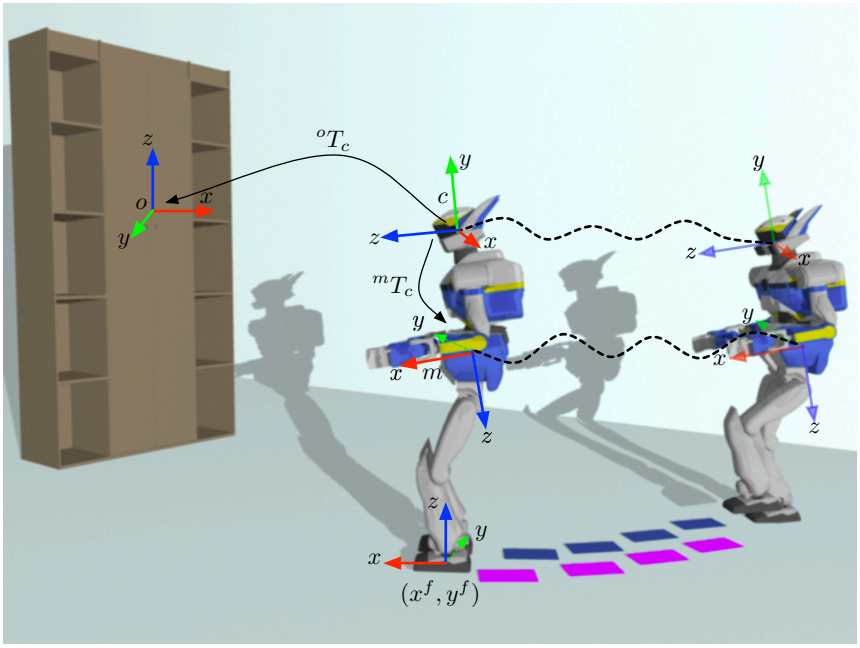


Figure 4.1: An example setup for our approach: the robot has to walk towards a desired position with regards to an object viewpoint. Frames o, c and m refer to the *object*, *camera* and *CoM* respectively. ${}^oT_c$ is the transformation to view the camera points in the object reference frame. (x^f, y^f) is the current footprint position.

4.2 Visual Control Using a Reference Velocity

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. 3.14 of the Model Predictive Control and the real velocity effectively attained from the first control. 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:

$$\mathrm{v}^c = \overline{\mathrm{v}}^c + {}^cV_m \mathrm{v}^b \tag{4.1}$$

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 ${}^cV_m$ is a 6×6 twist matrix relating the camera frame (index "c") to the CoM frame (index "m") through the transform ${}^cT_m$.

We recall how in [Dune et al., 2010], a classical visual servoing scheme is adapted to be used for a humanoid robot. The idea, basically, is to use the output velocity from Eq. 4.5 as a reference velocity for Eq. 3.14. 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 not corresponding 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,

$$\dot{s} = L_e \overline{\mathrm{v}}^c + L_e \, {}^cV_m \mathrm{v}^b. \tag{4.2}$$

Then, the idea is to use a virtual camera that would correspond to the position of the camera if we supposed that there were no sway motion affecting the robot motion, to control this virtual camera, and use its controlled velocity ($\overline{\mathrm{v}}^c$) as a reference velocity in the reactive PG.

Integrating equation 4.2 between 0 and t, Dune et al. have shown that the relationship between the real observed features $s(t)$ and the "sway-less" features $\overline{s}(t)$ can be written as

$$s(t) = \overline{s}(t) + \int_0^t \widehat{L_e} \, {}^cV_m \, \mathrm{v}^b dt - E, \tag{4.3}$$

where $E \stackrel{\text{def}}{=} s(0) - \overline{s}(0)$. From the previous equation, the corrected visual error between a reference features vector and the sway-less features vector, to be used for the computation of the reference velocity, is deduced as

$$\overline{e}(t) = \overline{s}(t) - s^* = e(t) - (\int_0^t \widehat{L_e} \, {}^cV_m \, \mathrm{v}^b dt - E). \tag{4.4}$$

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, because of the sway.

The shift E is re-estimated regularly over one period of time (e.g., the last period) by

$$E = \frac{1}{T^{sway}} \int_{t-T^{sway}}^{t} \int_{0}^{t} \widehat{L_e}^{c} V_m \ \mathrm{v}^b dt dt,$$

and finally, it is used in the control law for the CoM (e.g., the reference velocity for Eq. 3.14)

$$\overline{\mathrm{v}}^m = -\lambda^c V_m \widehat{L_e}^{+} (e - (\int_{0}^{t} \widehat{L_e}^{c} V_m \ \mathrm{v}^b dt - E)). \tag{4.5}$$

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

This reference velocity can be directly injected in the Velocity Controlled Pattern Generator introduced in previous section. However, this approach has several drawbacks. First, the control law will never be executed as is, because of dynamical and physical limitations of the robot. Second, to get equation 4.5, they make several assumptions, which sometimes are not true. And third, since the PG and the servo controller are totally decoupled, the visual feedback has to wait for the next iteration to be introduced in the PG. In order to solve this problem, we will present in the next section a Visual Servo controller which will be directly introduced in the Pattern Generator.

4.3 Visual servoing and MPC

As we know, in walking pattern generation, MPC is used to estimate a sequence of optimal controls at some horizon, because of the step-based nature of walking. Hence, we 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 this chapter, in order to simplify the notation, we will use the form s_k to express the time reference k of $s(k)$ and alike.

In [Allibert et al., 2010], such a time horizon-aware scheme has been proposed.

The visual predictive control (VPC) is introduced as:

$$\min_{U_k} \quad \sum_{j=k+1}^{k+N} [s_j^d - s_j^m]^\top W_j [s_j^d - s_j^m], \tag{4.6}$$

$$\text{subject to} \quad s_j^d = s_j^* - \epsilon_j, \tag{4.7}$$

$$q_j = f(q_{j-1}, u_{j-1}), \tag{4.8}$$

$$s_j^m = h(q_j). \tag{4.9}$$

In Eq. 4.6, $U_k = u_{k:k+N-1}$ are the series of controls to be applied, q_j is the state, and 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 current time k, because by definition the s_j are not known for $j > k$. Since our landmarks are static, $s_j^* \stackrel{\text{def}}{=} s^*$, 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. 4.8 is the dynamic model, that estimates the new state given the last state/control pair. In general, this function is non-linear. We will see how to deal with this non-linearity.

Eq. 4.9 is also a non-linear function that estimates the output of the model s_j^m, given the current state q_j. This equation implements the pinhole camera projection model.

Matrix W_j in Eq. 4.6 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, $W_j = diag(w_j)$.

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. 4.6 can be rewritten as

$$\min_{U_k} \sum_{l=0}^{M} [S_l^d - S_{l,k}^m]^\top W [S_l^d - S_{l,k}^m], \tag{4.10}$$

with $S_{l,k}^m = \left(s_{l,k+1}^m, s_{l,k+2}^m, \ldots, s_{l,k+N}^m\right)^\top$ stacking the features positions in the horizon, S_l^d stacking the corresponding desired positions, and the weights matrix $W = diag(w_{k+1}, w_{k+2}, ..., w_{k+N})$.

Allibert et al. solved directly the non-linear programming problem Eqs. 4.6-4.9 to compute the controls in their visual servoing scheme.

4.4 Integrating the visual servoing to the walking motion generator

In the following we use superscripts w, c, m to refer the world, the camera, and the CoM reference frame, and ${}^{c}T_{m}$ means transformation from c to m frame. If we introduce directly Eq. 4.10 in Eq. 3.14, we will not have a QP anymore due to the non-linear constraints, namely Eqs. 4.8 and 4.9. We can directly solve the first problem (Eq. 4.8) by using the dynamic model in Eq. 3.10. In this case there is no rotation, but we will see that we can introduce it in a decoupled way without losing the QP formulation.

4.4.1 Linearization of the observation model

As we said, Eq. 4.9 implements the pinhole camera model. Let ${}^{w}p_{l'} = [{}^{w}x_{l'}, {}^{w}y_{l'}, {}^{w}z_{l'}]^{\top}$ be the position of the l'-th landmark in the world reference frame. At time j, one can compute the projection to the image plane by first transforming the landmark position to the camera frame with the homogeneous transform ${}^{m}T_{c}\,{}^{w}T_{m,j}$ and then applying the projection,

$$\begin{pmatrix} u_{l',j} \\ v_{l',j} \end{pmatrix} = \begin{pmatrix} u({}^{c}x_{l',j}, {}^{c}y_{l',j}, {}^{c}z_{l',j}) \\ v({}^{c}x_{l',j}, {}^{c}y_{l',j}, {}^{c}z_{l',j}) \end{pmatrix} = \begin{pmatrix} {}^{c}x_{l',j}/{}^{c}z_{l',j} \\ {}^{c}y_{l',j}/{}^{c}z_{l',j} \end{pmatrix}, \tag{4.11}$$

where ${}^{w}T_{m,j}$ is the transformation from the world frame to the CoM frame at time j and ${}^{m}T_{c}$ is the transformation from the CoM frame to the camera frame, which is not variable in our approach. Note that

$${}^{w}T_{m,j} = ({}^{m}T_{w,j})^{-1} = \begin{pmatrix} ({}^{m}R_{w,j})^{-1} & -({}^{m}R_{w,j})^{-1}{}^{m}t_{w,j} \\ 0_{1\times 3} & 1 \end{pmatrix}$$

where ${}^{m}t_{w,j}$ is the position of the CoM in the world frame at time j, which depends in our control variables through Eq. 3.10. ${}^{m}R_{w,j}$ is the direction of the robot waist according to the world reference frame at time j. In our current formulation there

is no free variables modifying this quantity, because it would make the problem non-linear. More details about this problem are given in Section 4.4.2.

If we use directly Eq. 4.11 (non-linear), we will lose the QP formulation. We know that Eq. 4.11 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 from MPC that prediction is done over a finite horizon. So it might be enough to use a first order approximation of h for small (dx, dy, dz) so that we can maintain the QP form.

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

$$\begin{cases} u(x_0 + dx, y_0 + dy, z_0 + dz) \approx \dfrac{x_0}{z_0} + \dfrac{dx}{z_0} - \dfrac{x_0 dz}{z_0^2}, \\ \\ v(x_0 + dx, y_0 + dy, z_0 + dz) \approx \dfrac{y_0}{z_0} + \dfrac{dy}{z_0} - \dfrac{y_0 dz}{z_0^2}. \end{cases}$$

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

We propose to apply such a linearization of Eq. 4.11 for the whole horizon, around the first position ($j = k$) of landmark l', i.e. at the linearization point $({}^c x_{l',k}, {}^c y_{l',k}, {}^c z_{l',k})$, that is, the point we are actually watching in the CoM frame.

This way, we can express the predicted position of the landmark l' linearly at time $j > k$ in the horizon:

$$\begin{pmatrix} u_{l',j} \\ v_{l',j} \end{pmatrix} = \begin{pmatrix} \pi^{11}_{l',k} \, {}^c x_{l',j} + \pi^{13}_{l',k} \, {}^c z_{l',j} + u_{l',k} \\ \pi^{22}_{l',k} \, {}^c y_{l',j} + \pi^{23}_{l',k} \, {}^c z_{l',j} + v_{l',k} \end{pmatrix},$$

where $u_{l',k} = {}^c x_{l',k} / {}^c z_{l',k}$ and $v_{l',k} = {}^c y_{l',k} / {}^c z_{l',k}$ are the initial image positions of the landmarks in the horizon and the coefficients $\pi^{ij}_{l',k}$ the elements of the matrix

$$\Pi_{l',k} = \begin{pmatrix} 1/{}^c z_{l',k} & 0 & -u_{l',k}/{}^c z_{l',k} \\ 0 & 1/{}^c z_{l',k} & -v_{l',k}/{}^c z_{l',k} \end{pmatrix},$$

which is the classical definition of the interaction matrix in Image-Based Visual Servoing.

Finally, we can express the projection of the $l' - th$ landmark (constraint 4.9) as:

$$\begin{pmatrix} u_{l',j} \\ v_{l',j} \end{pmatrix} = \begin{bmatrix} \Pi_{l',k} & \begin{matrix} u_{l',k} \\ v_{l',k} \end{matrix} \end{bmatrix} {}^{m}T_{c}\ {}^{w}T_{m,j} \begin{pmatrix} {}^{w}p_{l'} \\ 1 \end{pmatrix}, \tag{4.12}$$

so that we can now introduce the visual features in the pattern generator. Expanding terms in Eq. 4.12 for the first row and setting $\Pi^{u}_{l',k}$ as the first row of matrix $\Pi_{l',k}$,

$$u_{l',j} = \Pi^{u}_{l',k}({}^{w}R_{c}\,{}^{w}p_{l'} + {}^{w}R_{c}\,{}^{m}t_{w,j} + {}^{m}t_{c}) + u_{l',k}. \tag{4.13}$$

Since ${}^{w}R_{c}\,{}^{m}t_{w,j} = {}^{w}R_{c1}x_j + {}^{w}R_{c2}y_j + {}^{w}R_{c3}z_j$, where ${}^{w}R_{ci}$ is the i-th column of ${}^{w}R_{c}$ [1] and x_j, y_j, z_j the position of the CoM at time j (see Chapter 3). We can rewrite Eq. 4.13

$$u_{l',j} = a^{u}_{l',k}x_j + b^{u}_{l',k}y_j + c^{u}_{l',k}, \tag{4.14}$$

with

$$\begin{cases} a^{u}_{l',k} &= \Pi^{u}_{l',k}\,{}^{w}R_{c1} \\ b^{u}_{l',k} &= \Pi^{u}_{l',k}\,{}^{w}R_{c2} \\ c^{u}_{l',k} &= \Pi^{u}_{l',k}({}^{w}R_{c}\,{}^{w}p_{l'} + {}^{m}t_{c} + {}^{w}R_{c3}z_j) + u_{l',k}. \end{cases}$$

Equivalently,

$$v_{l',j} = a^{v}_{l',k}x_j + b^{v}_{l',k}y_j + c^{v}_{l',k}. \tag{4.15}$$

Stacking the features $u_{l',j}$ and the CoM positions for the whole horizon and using Eq. 3.10, we get a vector $S^{m}_{l,k}$ similar to the one introduced in Eq. 4.10

$$S^{m}_{l',k} = A^{u}_{l',k}C_x(k+1) + B^{u}_{l',k}C_y(k+1) + C^{u}_{l',k},$$

with

$$\begin{aligned} A^{u}_{l',k} &= a^{u}_{l',k}I_{N\times N} \\ B^{u}_{l',k} &= b^{u}_{l',k}I_{N\times N} \\ C^{u}_{l',k} &= c^{u}_{l',k}(1,1,...,1)^{\top} \end{aligned}$$

[1] To simplify the notations ${}^{w}R_{c} = {}^{m}R_{c}\,{}^{w}R_{m,j}$

which corresponds to the predicted coordinates u of the landmark l'-th 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 the l-th feature is the u coordinate in the image of the landmark $l' = \lfloor l/2 \rfloor$ for l even, and the v coordinate for l odd.

Generalizing to all features, we have:

$$S^m_{l,k} = A_{l,k} C_x(k+1) + B_{l,k} C_y(k+1) + C_{l,k}, \tag{4.16}$$

with $A_{l,k} = A^u_{l',k}$ for l even and $A_{l,k} = A^v_{l',k}$ for l odd. The same holds for $B_{l,k}$ and $C_{l,k}$.

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

$$\begin{aligned} \min_{U(k)} \quad & \frac{\alpha}{2} \left\| \dddot{C}_x(k) \right\|^2 + \frac{\gamma}{2} \left\| Z_x(k+1) - Z^{ref}_x(k+1) \right\|^2 \\ & + \frac{\alpha}{2} \left\| \dddot{C}_y(k) \right\|^2 + \frac{\gamma}{2} \left\| Z_y(k+1) - Z^{ref}_y(k+1) \right\|^2 \\ & + \frac{\beta}{2} \sum_{l=0}^{M} [S^d_l - S^m_{l,k}]^\top W [S^d_l - S^m_{l,k}], \end{aligned}$$

and as a canonical QP:

$$\min_{U(k)} \frac{1}{2} U(k)^\top Q(k) U(k) + p(k)^\top U(k)$$

with

$$Q(k) = \begin{pmatrix} Q(k)' & 0 \\ 0 & Q(k)' \end{pmatrix} + \hat{Q}(k),$$

$$Q(k)' = \begin{pmatrix} \alpha I + \gamma U_z^\top U_z & -\gamma U_z^\top V \\ -\gamma V^\top U_z & \gamma V^\top V \end{pmatrix},$$

$$\hat{Q}(k) = \begin{pmatrix} \beta \sum_l U_p^\top A_{l,k}^\top W A_{l,k} U_p & 0 & \beta \sum_l U_p^\top A_{l,k}^\top W B_{l,k} U_p & 0 \\ 0 & 0 & 0 & 0 \\ \beta \sum_l U_p^\top B_{l,k}^\top W A_{l,k} U_p & 0 & \beta \sum_l U_p^\top B_{l,k}^\top W B_{l,k} U_p & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

and $p(k) = p(k)' + \hat{p}(k)$,

$$p'(k) = \begin{pmatrix} \gamma U_z^\top (S_z \hat{c}_x(k) - V_c \hat{F}_x(k) \\ -\gamma V^\top (S_z \hat{c}_x(k) - V_c \hat{F}_x(k)) \\ \gamma U_z^\top (S_z \hat{c}_y(k) - V_c \hat{F}_y(k)) \\ -\gamma V^\top (S_z \hat{c}_y(k) - V_c \hat{F}_y(k)) \end{pmatrix},$$

$$\hat{p}(k) = \begin{pmatrix} \beta \sum_l U_u^\top A_{l,k}^\top W [A_{l,k} U_s \hat{c}_x(k) + B_{l,k} U_s \hat{c}_y(k) + C_{l,k} - S_l^d] \\ 0 \\ \beta \sum_l U_p^\top B_{l,k}^\top W [A_{l,k} U_s \hat{c}_x(k) + B_{l,k} U_s \hat{c}_y(k) + C_{l,k} - S_l^d] \\ 0 \end{pmatrix}.$$

One must note in Eq. 4.17, unlike the velocity controlled PG, this QP does not include the reference velocity tracking, instead, it includes the minimization of the visual errors. It also includes the terms of the minimization of the jerks and the ZMP centering.

4.4.2 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 features s^d and the predicted features 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]. The approach is a decoupled solution, that is, it estimates first the optimal rotation angles and afterwards introduces these values as known in the

main QP (as the matrix ${}^{m}R_{w,j}$). This scheme should not affect the stability of the walking since inertial effects are not taken into account.

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

$$\min_{\dddot{C}_\theta(k),\dddot{F}_\theta(k)} \quad \frac{\beta}{2}\left\|C_\theta(k+1)-\Theta^0\right\|^2+\frac{\gamma}{2}\left\|F_\theta(k+1)-\Theta^0\right\|^2 \\ +\frac{\alpha}{2}\left\|\dddot{C}_\theta(k)\right\|^2+\frac{\alpha}{2}\left\|\dddot{F}_\theta(k)\right\|^2, \tag{4.17}$$

with the same notations as for $\dddot{C}_x(k)$ and $\dddot{C}_y(k)$, $\dddot{C}_\theta(k)$ is the sequence of N jerk values to be applied, and $C_\theta(k+1)$ is the sequence of predicted θ values, i.e. the orientations of the trunk,

$$C_\theta(k+1) \stackrel{\text{def}}{=} (c_\theta(k+1),...,c_\theta(k+N))^\top,$$

and similarly for $F_\theta(k+1)$, the feet orientations. Finally, as a reference orientation Θ^0 is defined once for all at the starting configuration as a target feet orientation. Several conventions exist, in this paper, the trunk orientation $C_\theta(k)$ is trying to follow the flying foot orientation $F_\theta(k)$. The flying foot is the only one which can move during the single support phase. Certainly, the support foot is fixed, and both feet are fixed during the double support phase. Finally, zero speed, and zero acceleration are specified at the beginning and the end of the trajectories.

Then in a second stage, we introduce these angles as constant in the main QP (Eq. 3.14). This approach gives us the advantage of introducing the following constraints,

$$|C_\theta(k+1)-F_\theta(k+1)|<\Theta^{FT}_{max} \tag{4.18}$$

$$|C^{f,L}_\theta(k+1)-C^{f,R}_\theta(k+1)|<\Theta^{LR}_{max} \tag{4.19}$$

$$|C_\theta(k+1)|<\Theta^{visibility}. \tag{4.20}$$

Eq. 4.18 constraint the maximum rotation between feet and trunk, Eq. 4.19 between both feet, and Eq. 4.20 sets a rotation limit of the trunk to keep the visibility of the landmarks.

4.4.3 Visual constraints

The introduction of visual constraints can be done by using Eq. 4.14 and Eq. 4.15. 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 only want all landmarks to follow trajectories inside of some convex polytope. Hence, constraints become constant in time and for all landmarks. This can be written as:

$$\begin{aligned} A' \begin{pmatrix} A^u_{l',k} C_x(k+1) + B^u_{l',k} C_y(k+1) + C^u_{l',k} \\ A^v_{l',k} C_x(k+1) + B^v_{l',k} C_y(k+1) + C^v_{l',k} \end{pmatrix} &\leq b' \\ \text{and then, } A'' U_k &\leq b'', \end{aligned} \tag{4.21}$$

where matrix A' and vector b' are related with the image constraints. For example, bound constraints in the (u, v) coordinates like visibility constraints are easily expressed in terms of Eq. 4.21 and are introduced directly in the QP.

4.4.4 Qualitative comparison with the classical approach

A first advantage of introducing 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, with velocities that eventually could not be realized by the humanoid robot. On the opposite, with our approach, because the visual errors term is only one term in the QP problem (1) the exponential decay is attenuated by the regularizing effect of other terms such as the jerks and (2) the constraint on velocities (maximal velocities) are naturally handled.

Moreover, since the velocity reference we set as an input to the pattern generator is not truly performed (due to 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 Model Predictive Control. The sway motion is naturally filtered since we minimize the errors within a full

cycle (the horizon in the Model Predictive Control). 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.

We know classical visual servoing control laws have very good performance in robotic arms. However, stepping is a highly dynamic process. We can not ignore the stability and limitations in the design of our control laws. For instance, in classical visual servoing, while reaching the goal, the velocity controls requested to the robot are getting smaller, and null velocities are theoretically reached at infinite time, due to the exponential decay. Clearly we stop the motion when some convergence criteria is reached. This is not a problem with robotic arms since we just send rotational velocities to the motors, so this velocity can be very small. On the other side, the motion can take long time, and the stability of the robot arm system is not jeopardized. Stepping involves balance, and every step could break it. We must avoid unnecessary motion and reach the goal as soon and efficient as possible.

4.5 Simulation results

4.5.1 Simulation results on the MPC-based approach only

We first tested our own approach (Section 4.3) in a simulated environment with the HRP-2 robot model 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 $(2, 1)$. First, we try with a desired final position that does not imply rotation, so that the robot has just to control the x and y velocities, which are the variables in the QP. Depending of the weights of the QP, we obtain different trajectories, such as in Figs. 4.2 and 4.3. The difference between these two simulations is that we increased the β parameter ($\beta = 0.001$ in the first, $\beta = 0.005$ in the second). 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.

Now, in a second experiment, we evaluate the trajectory with rotation. We

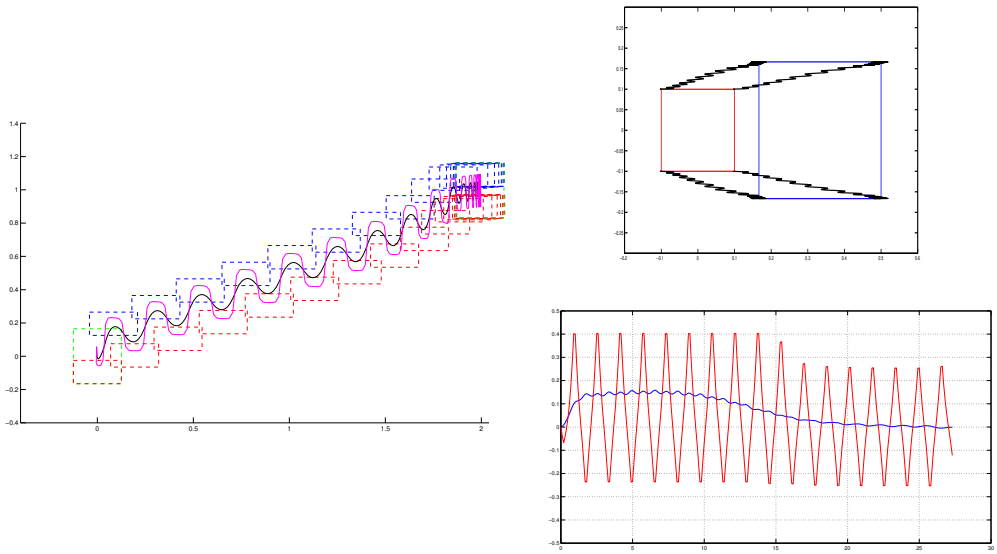


Figure 4.2: On the left, we show the trajectory of the robot in the x, y plane, driven by our MPC-based coupled approach. The initial and final double stance phases appear in green. The single stance support feet appear in red (resp. blue) for the right (resp. left) foot. In pink, we depicted the CoP 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.

recall that the rotation velocity control is decoupled from the x and y velocities control. It works as follows: The rotation velocity controller sets (in a decoupled way) the angular position while the main controller (QP) adapts the x, y velocities in terms of these computed angular positions (which are known in the horizon), the visual errors, the footsteps centering and the jerks minimization. The results of a first simulation involving rotation is shown in Fig. 4.4. 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. 4.5, we can see that it is robust to perturbations: The same trajectory is followed as in Fig. 4.4 but a strong perturbation in the CoM position has been introduced (simulating an external force applying to the CoM or a string error in the position estimation), inducing visible peaks in the velocities. However, the perturbation is recovered quasi-instantly.

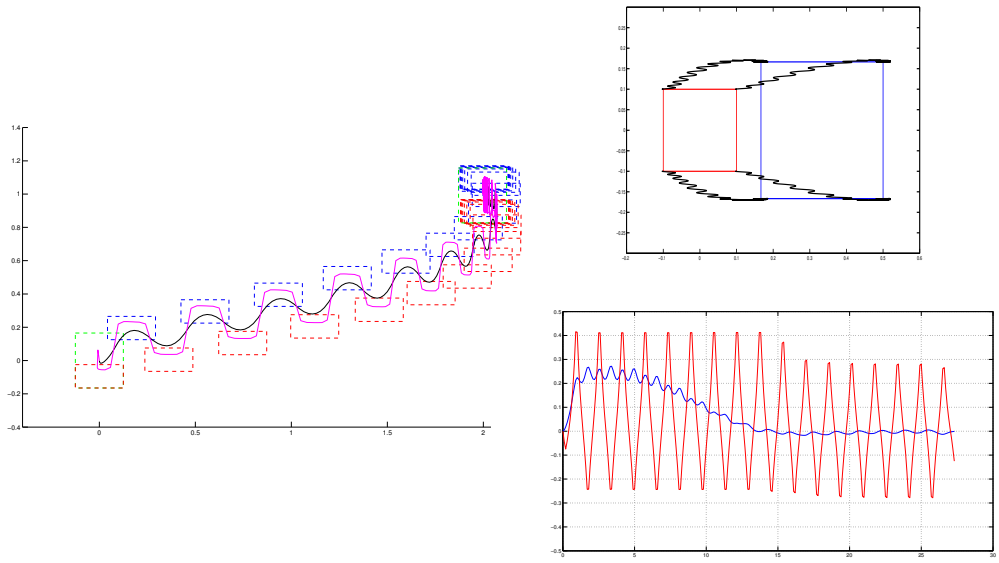


Figure 4.3: The behavior of our system in the second simulation (β is 5 times higher), with the same graphical conventions as in Fig. 4.2. The robot is following a quite distinct trajectory, with a backwards part close to the end. As the priority is to minimize features errors, disregarding the jerks, high velocities are taken.

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. 4.6, 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 (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 MPC is that it naturally filters out these oscillations because we minimize the errors in a full cycle. It is remarkable that,

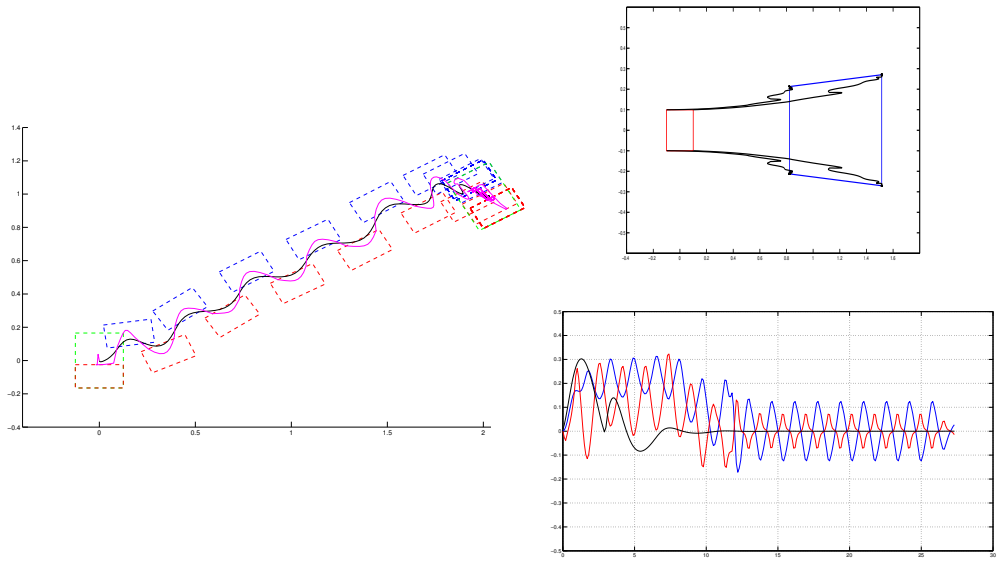


Figure 4.4: The behavior of our system in the third simulation, involving rotation, with the same graphical conventions as in Fig. 4.2. In the velocities graph (right, bottom) we added the rotation velocity in black. The robot is walking in the sagittal direction most of the time, after an initial rotation. The amplitude of the oscillations 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.

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. 4.7). In Fig. 4.7, 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.

4.5.2 Comparisons: coupled vs. decoupled approaches

We conducted a simple experiment to illustrate better the differences between the two approaches of Section 4.2 and Section 4.3, and to have quantitative comparisons between them. In this experiment, the robot has simply to go

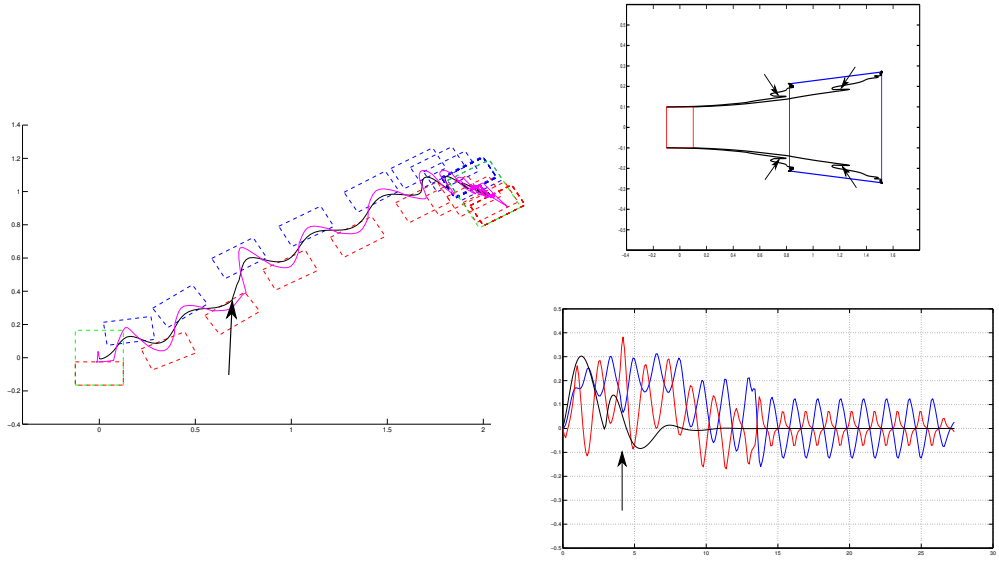


Figure 4.5: The behavior of our system in the fourth simulation, with the same graphical conventions as in Fig. 4.4. The robot is following a trajectory similar to the one of Fig. 4.4. After a few footsteps, a strong perturbation is applied to the CoM, inducing a peak in the velocities graph (right, bottom). This perturbation is small in distance metric, so it is not visible in the features trajectories. The perturbation is quasi-instantly recovered.

three meters forward. In Fig. 4.8, we depict the robot trajectories and footsteps (left column) and the features trajectories (right column) for each approach (the coupled one in the upper line, the decoupled one 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 oscillations (as a result of the immediate stepping) are visible.

In Fig. 4.9 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

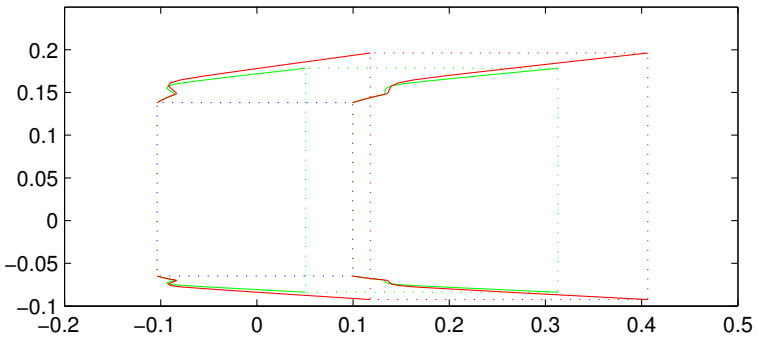


Figure 4.6: In this figure, we show the trajectory of the visual features in one iteration of the QP, and compare the evolution of the features obtained by the linearization model (green lines) and features obtained by the exact non-linear model (red lines).

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 walking pattern generator with visual predictive control, this is not true anymore, see Fig. 4.9 right, in particular, because we take into account future information, so the coupled system converges faster. In the errors evolution, we use the same symmetry argument as in Fig. 4.7. We should note that the u components in the image plane are theoretically the oscillatory ones (from the stepping motion). In Fig. 4.9, 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.

4.6 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 chapter, we have proposed a novel approach to close the control loop by introducing visual information into the walking pattern generation. Our online pattern generator integrates the regulation of the relative pose of image features while simultaneous ensuring safety and stability. In order to keep the optimization formulation as a QP, the perspective projection equations have been linearized

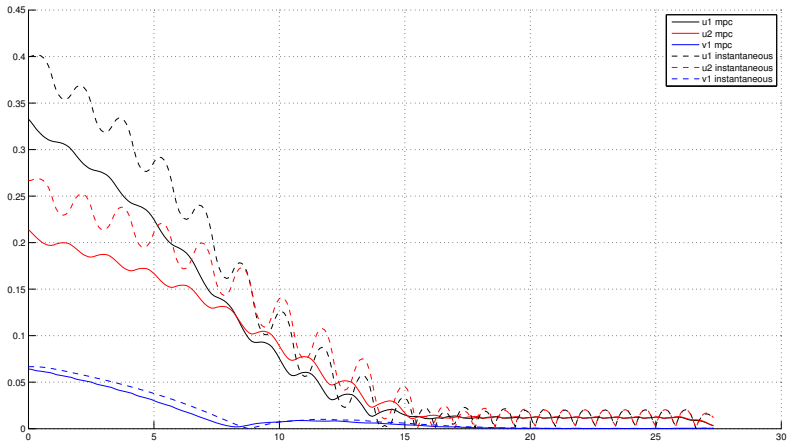


Figure 4.7: Evolution of the errors for the u, v components of each feature, in the second simulation. In dashed line, we depict the instantaneous errors along time, and in solid line, we depict the errors estimated in the horizon (i.e. individual terms of Eq. 4.10, normalized by the size of the horizon). 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.

around the features positions at the beginning of each cycle. However, our current approach uses 3D information that strongly depends on the localization. As an ongoing work, we wish to drop the need of 3D information by predicting the image positions of the landmarks in terms on the velocity of the robot in the horizon (IBVS); we also plan to perform real experiments on the HRP-2 humanoid platform.

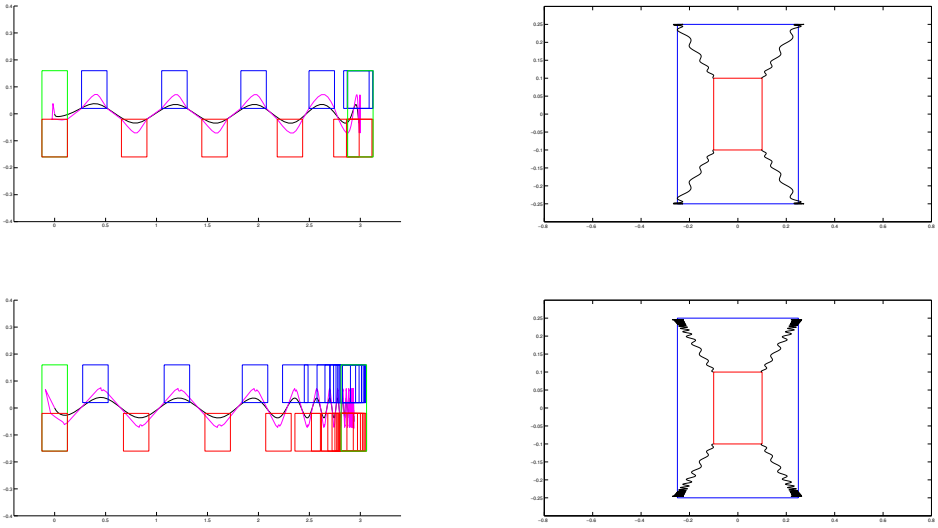


Figure 4.8: Comparison between the coupled (up) and decoupled approaches (bottom). On the left side, we depict the robot trajectory with the footsteps (red and blue), and the CoM (black) and CoP (pink) trajectories. On the right side, we depict the evolution of the instantaneous features positions during the experiment (from the red rectangle, at the beginning of the simulation, to the blue one, at the end).

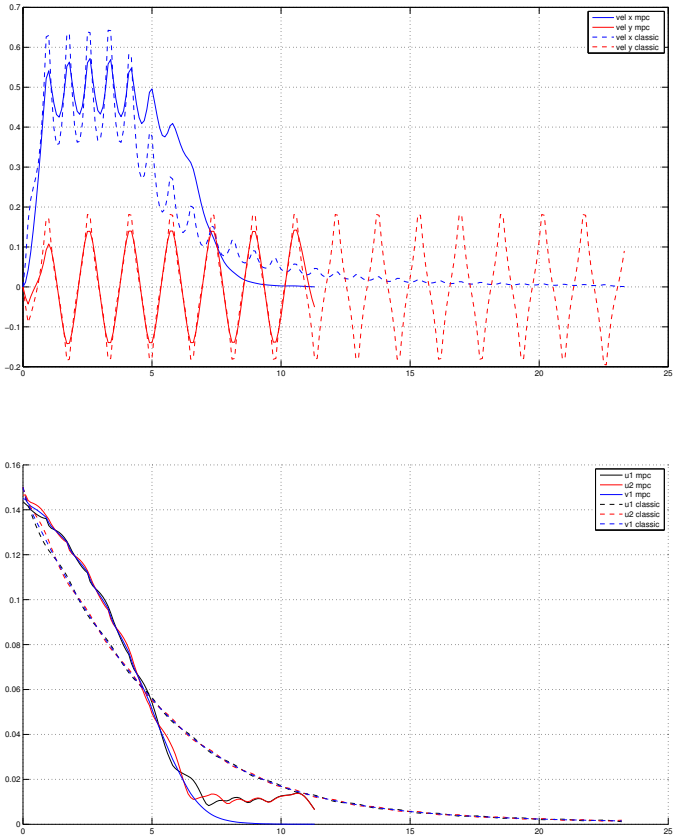


Figure 4.9: Left, comparison of velocity profiles in the x (blue) and y (red) axis, for the decoupled (dashed line) and coupled (solid line) approaches. Right, comparison of the features errors evolution.

5

Planning Visual Tasks for Reactive Humanoid Walking

In the previous chapter, we presented a scheme to control a humanoid robot directly by using visual servoing techniques. However, the trajectories realized by the robot in that case are generated to minimize the distance in the image features space and might create unnecessary motion in the space of the footprints. This is because visual servoing is a local control technique. If we want to take into account the shape of the trajectory or obstacles avoidance, a planning method would be necessary.

Motion planning aims to take the robot from an initial to a final position while ensuring feasibility and obstacle avoidance. The notion of optimality arises in motion planning. From the work of [Salaris et al., 2010; Hayet et al., 2012], it exists a set of optimal trajectories to take a nonholonomic robot, evolving in free space, from an initial to a final position while keeping in sight a landmark. Also, it has been shown that the human walking behaves nonholonomic for open spaces. In this chapter, we will work upon those works and we will inject the

motion primitives from the planner into the pattern generator level of the robot.

5.1 Global planning with visibility constraints

For visual servoing, localization, surveillance, or any robotic task based on the observation of visual cues, it is important to ensure that the robot under control does not lose sight of the landmark(s) used as a visual reference. Hence, several works have been proposed to provide motion planners that guarantee landmark visibility, mainly for wheeled robots.

With this concern in mind, we borrow one such global planner, described in [Hayet et al., 2012], as a base tool for generating paths that guarantee to enforce the required visual constraints while giving locally optimal trajectories in distance. This base planner for humanoid robots uses as an underlying model of motion, a disk-shaped non-holonomic Differential Drive Robot (DDR). In [Hayet et al., 2012], if a solution path is found for the DDR, it is converted into a solution path for the humanoid robot by generating a footprints sequence coherent with the humanoid dimensions. A nice property inherited from using the DDR model of motion is that the global planner is complete, i.e. if a solution exists, it will give one, otherwise, it will say so.

However, even if, locally, the used motion primitives are optimal in distance, this approach does not necessarily give globally optimal trajectories. We will not detail the complete methodology here, but we recall the basic results hereafter.

The aforementioned algorithm takes as an input a 2D map of the environment, with its obstacles and visual landmarks, an initial configuration of the robot in the plane, (x_A, y_A, θ_A), and a final (goal) configuration in the plane, (x_B, y_B, θ_B). Its output is a set of footprints in the plane to be followed by the robot. The building of this path follows a classical recursive strategy: A roadmap is built over the free space, which includes both, points free of collision with the obstacles in the environment and points that are not within the shadows cast by the landmarks. Then, the initial and final configurations are tested, to check whether the optimal primitives of [Salaris et al., 2010] can connect them without colliding with forbidden regions. If the computed path is without collision, then the algorithm ends, otherwise the holonomic shortest path on the roadmap from the initial configuration to the final (if it exists) is divided in two parts by its middle

point, and the whole process is repeated on the two sub-parts.

The resulting path is made of several parts, each one corresponding to a locally optimal path given from [Salaris et al., 2010] (made of a combination of straight lines, logarithmic spirals and in-site rotations). This allows to define a set of S sub-segments to be performed by the robot along locally optimal paths, which will be referred to as (p^s, q^s), for $s = 1 \dots S$:

$$p^s = (x_p^s, y_p^s, \theta_p^s)^T \; ; \; q^s = (x_q^s, y_q^s, \theta_q^s)^T,$$

with p^s the initial configuration and q^s the final one. The complete computed path is executed by reaching the successive sub-goals q^s.

5.2 Defining a local reference trajectory

Let us focus on the execution of each sub-path s. Suppose that we start with the robot at some configuration close to p^s (see Fig. 5.1), because localization may not be perfect, and suppose that we want to reach the final configuration or sub-goal q^s, as determined in [Hayet et al., 2012].

5.2.1 Adapting the reference trajectory for the MPC

Without loss of generality, we suppose that the object of interest to be kept at sight by the robot is located at $L^s = (0, 0)$. In all the following, x, y, θ will refer to coordinates w.r.t a global frame centered at this origin.

It is noteworthy that, given the sub-goal $q^s = (x_q^s, y_q^s, \theta_q^s)^T$ to reach, and the landmark it is associated to, the work of [Salaris et al., 2010] gives us a full synthesis of optimal paths in free space. This synthesis can be represented as a mapping σ:

$$\begin{array}{rccc} \sigma & : & \mathbb{R}^2 & \mapsto & \mathbb{S}^1 \times \mathbb{R}^2 \\ & & (x, y) & \to & (\theta^*(x, y), v^*(x, y), \omega^*(x, y)) \end{array}$$

where $\theta^*(x, y)$ is the orientation the robot (viewed as a nonholonomic cart) should have in order to start walking along the shortest path to q^s, from its current position (x, y). The pair $(v^*(x, y), \omega^*(x, y))$ are the instantaneous (reference) velocities needed to be applied to follow the shortest paths. Without loss of

generality, we can suppose that $v^*(x,y) = 0$ (for in-site rotations) or $v^*(x,y) = \pm 1$ (elsewhere). Given that the linear velocity $v^*(x,y)$, $\omega^*(x,y)$ is defined in function of the nature of the path segment,

$$\begin{cases} \omega^*(x,y) & = & 0 & \text{(straight line)} \\ & = & \pm\frac{\sin(\phi_{max})}{\sqrt{x^2+y^2}} & \text{(spiral)}, \end{cases}$$

where ϕ_{max} is the maximal bearing angle possible for the landmark, given the sensor and robot characteristics.

Our claim is that the knowledge of optimal policies at each point provides flexibility when generating a walking pattern. This is achieved by optimizing the footprint positions and the CoM trajectory "around" the nominal path output from the planner. Which is done by using these policies within the pattern generation.

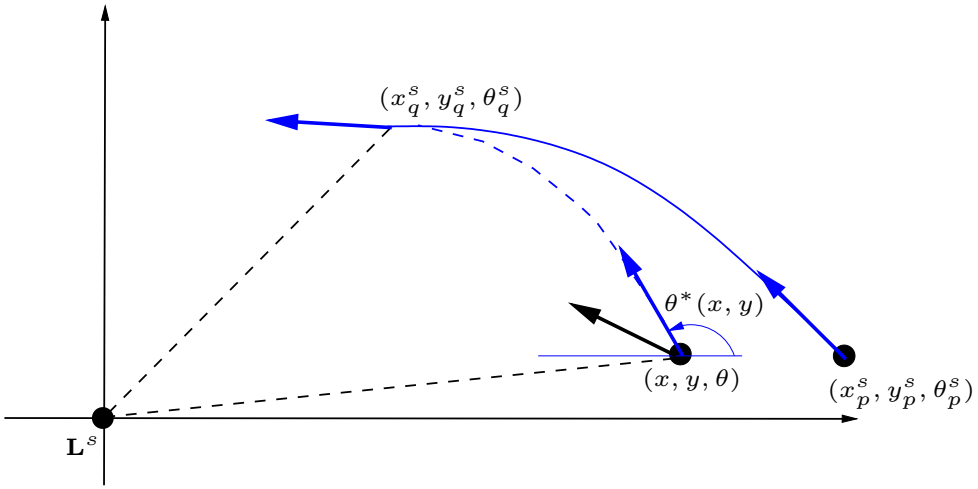


Figure 5.1: From a configuration (x,y,θ) and its corresponding position (x,y), and a sub-goal $q^s = (x_q^s, y_q^s, \theta_q^s)$ to reach, the optimal path (dashed line) is the one we want the humanoid robot to follow. For that, we use the tangent orientation $\theta^*(x,y)$ to this path. Because of the errors in control or localization, this optimal path may be different from the shortest path (solid line) computed from the first configuration $p^s = (x_p^s, y_p^s, \theta_p^s)$.

In pattern generation algorithms such as [Herdt et al., 2010a], the cost function within the Model Predictive Control (MPC) window uses the configurations of the CoM (see Chap. 3). Here, focusing more specifically on the (x,y,θ) CoM coordinates, we use the reference trajectory to be followed as an input of the

algorithm. And a synthesis of shortest paths leading to a given objective, as a direct output from [Salaris et al., 2010]. As depicted in Fig. 5.1, at each configuration evaluated within the MPC, two forces should apply through the optimization scheme: one driving the robot to the "correct" orientation $\theta^*(x, y)$, and one making the velocities follow the shortest path, $(v(x, y), \omega(x, y))$, so we can follow the trajectories according to the optimal synthesis. Also, a strong visibility constraint should apply, to ensure the object of interest to stay visible. In Fig. 5.2(a), we give an illustration of the shortest paths synthesis from [Salaris et al., 2010], displayed through the local orientations of optimal paths at each point of the plane, given the objective to reach ("end point") and the landmark to keep in sight.

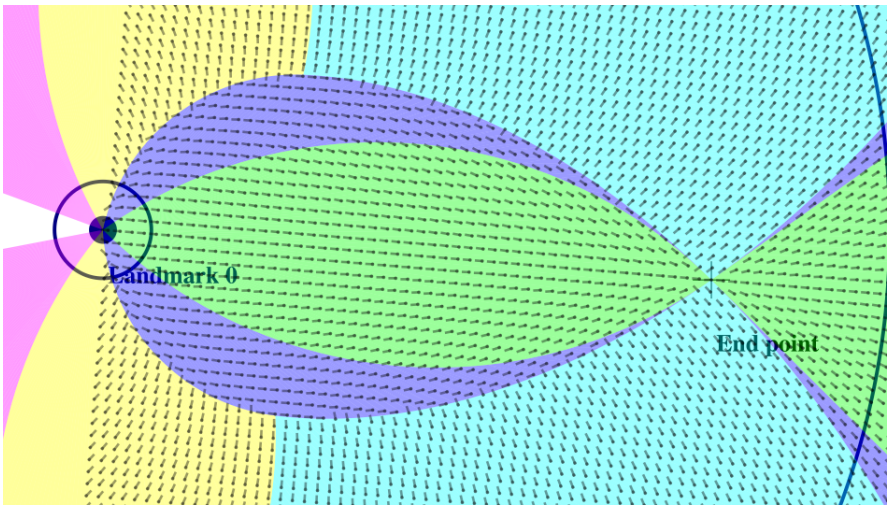
5.2.2 Using the optimal path synthesis within the MPC

Instead of utilizing in the Equation 3.14, a single, constant reference trajectory, defined by the plan computed from p^s to q^s, and from which $(\dot{X}^{ref}_{k+1}, \dot{Y}^{ref}_{k+1})$ would be evaluated, our idea is to use explicitly the mapping σ described above. This way, we can adapt the path execution and find the shortest path to the next subgoal from any point, not just from p^s. Even more, we can adapt to any change in the execution on the fly. As explained above, one can associate to any x, y the tangent $\theta^*(x, y)$ to the shortest path. One way to enforce the tracking of these shortest paths is to set as a reference velocity (supposing we are following a piece of curve with $v = 1$) defined in the time horizon ($l > k$, where k is the current time index):

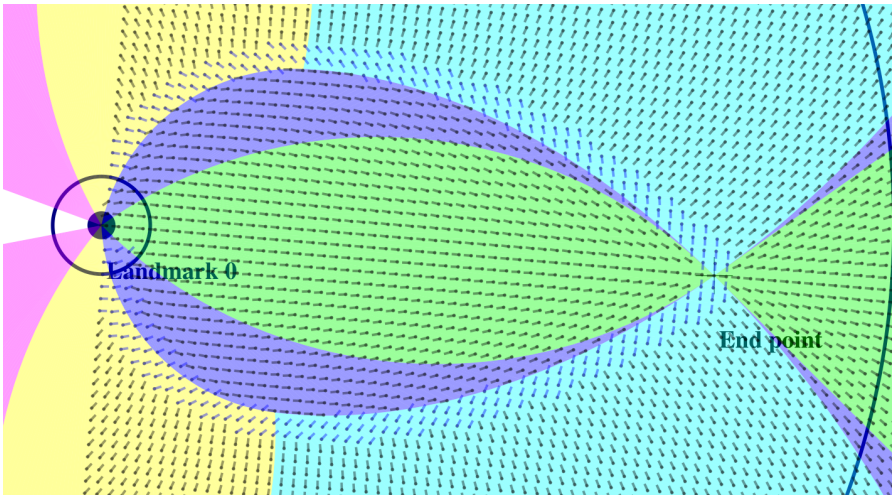
$$\dot{x}^{ref}_l = \cos(\theta^*(x_l, y_l)) \qquad \dot{y}^{ref}_l = \sin(\theta^*(x_l, y_l))$$

that depends non-linearly on x_l, y_l. The idea is to use these variable reference velocities inside the terms of the pattern generator QP. However, as this mapping is non-linear, a direct use would make us lose the QP form, that allows an efficient resolution of the problem. The mapping $\theta^*(x, y)$ has not always an analytic form, so we evaluate it numerically on a fine scale grid inside the area of operation of the robot, as pre-computed values. We also estimate numerically the partial derivatives $\frac{\partial\theta^*(x,y)}{\partial x}, \frac{\partial\theta^*(x,y)}{\partial y}$.

This way, we can approximate each of these reference velocities, at time



(a) Motion direction field resulting from the synthesis of [Salaris et al., 2010].



(b) Motion direction field with lateral motions.

Figure 5.2: Visualization of optimal motion directions $\theta^*(x, y)$. Both figures indicate the nature of optimal paths with the filling colors, and the tangent direction to the optimal curve all over a grid defined around the final point. In Fig. (a), the field is computed directly from the primitives of [Salaris et al., 2010]. The locus of in-site rotations is identifiable at the dark blue zone border. In Fig. (b), the non-holonomy behavior close to this locus is replaced by lateral motions.

position l within the optimization time window, by performing a linearization

of θ^* around a reference position (x^0, y^0), i.e.,

$$\begin{aligned}\theta^*(x_l, y_l) \;\approx\; & \theta^*(x_l^0, y_l^0) \\ & + \tfrac{\partial\theta^*(x^0,y^0)}{\partial x}(x_l - x^0) + \tfrac{\partial\theta^*(x^0,y^0)}{\partial y}(y_l - y^0).\end{aligned}$$

Note that the linearization point (x^0, y^0) is chosen in the aforementioned grid of pre-computed values $(\theta^*(x,y), \frac{\partial\theta^*(x,y)}{\partial x}, \frac{\partial\theta^*(x,y)}{\partial y})$, at the closest point to the first (current) CoM position. This grid is depicted in the background of Figs. 5.2, 5.3, 5.4 among others. To simplify the notation, let us write

$$\theta^0 \stackrel{\text{def}}{=} \theta^*(x^0, y^0),\; \frac{\partial\theta^0}{\partial x} \stackrel{\text{def}}{=} \frac{\partial\theta^*(x^0, y^0)}{\partial x},\; \frac{\partial\theta^0}{\partial y} \stackrel{\text{def}}{=} \frac{\partial\theta^*(x^0, y^0)}{\partial y}.$$

Then, we re-write the errors to the reference velocities at each time step $l > k$, as a linear function of $\dot{x}_l, \dot{y}_l, x_l, y_l$

$$\begin{cases} \dot{x}_l - \dot{x}_l^{ref} &= \dot{x}_l - v\cos(\theta^0) \\ & \quad + v\sin(\theta^0)\frac{\partial\theta^0}{\partial x}(x_l - x^0) \\ & \quad + v\sin(\theta^0)\frac{\partial\theta^0}{\partial y}(y_l - y^0), \\ \dot{y}_l - \dot{y}_l^{ref} &= \dot{y}_l - v\sin(\theta_l^0) \\ & \quad - v\cos(\theta^0)\frac{\partial\theta^0}{\partial x}(x_l - x^0) \\ & \quad - v\cos(\theta^0)\frac{\partial\theta^0}{\partial y}(y_l - y^0), \end{cases}$$

and by stacking the errors within the horizon window as in Eq. 3.10, we get the following linear relations

$$\begin{aligned} \dot{C}_x(k+1) - \dot{C}_x^{ref}(k+1) &= \dot{C}_x(k+1) - \dot{C}_x^0(k+1) + A_x^0 C_x(k+1) + B_x^0 C_y(k+1), \\ \dot{C}_y(k+1) - \dot{C}_y^{ref}(k+1) &= \dot{C}_y(k+1) - \dot{C}_x^0(k+1) + A_y^0 C_x(k+1) + B_y^0 C_y(k+1), \end{aligned} \tag{5.1}$$

where $A_x^0, B_x^0, A_y^0, B_y^0$ are diagonal matrices collecting the terms $v\sin(\theta_l^0)\frac{\partial\theta_l^0}{\partial x}$ and alike. Then, the walking pattern generation is formulated exactly as in Eq. 3.14, with the reference velocities given by Eq. 5.1, and with the optimization variable being $U(k)$, leading to a canonical Quadratic Program (QP) similar to Eq. 3.16.

Non-linear constraints arise from the CoP position to be included in the support polygon. In this case, we set the robot and feet orientation as specified in next section and include the computed values into the QP.

5.2.3 Control of the rotation angle

To get the robot oriented with the tangent to the optimal path, θ^0, we use a decoupled approach in a very similar way as what we already described in Section 3 for the control of the rotation angles of the trunk and the feet [Herdt et al., 2010b]. With this formulation, we also mantain the QP form. Hence, in a first stage, we optimize the orientations in the MPC time window by

$$\begin{aligned}\min_{\dddot{C}_\theta(k),\dddot{F}_\theta(k)} \quad & \frac{\beta}{2}\left\|C_\theta(k+1)-\theta^0\right\|^2+\frac{\gamma}{2}\left\|F_\theta(k+1)-\theta^0\right\|^2 \\ & +\frac{\alpha}{2}\left\|\dddot{C}_\theta(k+1)\right\|^2+\frac{\alpha}{2}\left\|\dddot{F}_\theta(k+1)\right\|^2,\end{aligned}$$

and then in a second stage, we introduce these angles as constant in the main QP (Eq. 3.14). This approach gives us the advantage of introducing constraints like maximum rotation between both feet, between feet and trunk and also a rotation limit to keep the visibility of the landmarks.

5.3 Including holonomic behavior

Handling holonomic and non-holonomic behaviors together during locomotion has been previously been discussed in [Mombaur et al., 2008] in a context of motion planning.

One of the most visible disadvantages of the optimal non-holonomic paths given from [Hayet et al., 2012] is the presence of in-site rotations, that are not efficient in terms of footsteps number. One solution is to use a different formulation of the cost function using weights according to the robot direction and control the head. But including this vision based control in Eq. 3.14 is incompatible with a QP formulation. A strategy we propose here is to take advantage of the fact that the set of points where in-site rotations occur is very well defined geometrically in the plane, as a direct consequence of the synthesis

from [Salaris et al., 2010]. As illustrated in Fig. 5.2(a), it is the outer boundary of the partition zone where the shortest paths have to be done as line segments followed by spirals, i.e. the dark blue region of the figure. This curve is made of two parts [Salaris et al., 2010], one arc of circle and one piece of logarithmic spiral. Hence, we propose to perform the following: for all points inside the partition regions in contact with this locus, we evaluate its distance in terms of the primitive to be done to reach this locus and modify the reference velocities as follows.

If the robot is far from the locus of in-site rotations, then the path to follow is continuously derivable and goes either forwards or backwards; in that case, we use the "non-holonomic behavior" as described in Section 5.2, with the robot orientation controlled to stay close to $\theta^*(x_l, y_l)$,

$$\dot{x}_l^{ref} = \cos(\theta^*(x_l, y_l)) \ , \ \dot{y}_l^{ref} = \sin(\theta^*(x_l, y_l)),$$

and if the robot configuration is close to this locus, then we use instead lateral motions, with orientation $\phi(x_l, y_l) + \pi$, where $\phi(x_l, y_l) = \arctan \frac{y_l}{x_l}$ is the polar angle of (x_l, y_l),

$$\dot{x}_l^{ref} = -\varepsilon \sin(\phi(x_l, y_l)) \ , \ \dot{y}_l^{ref} = \varepsilon \cos(\phi(x_l, y_l)),$$

where $\varepsilon = \pm 1$ in function of the relative position of the goal to reach with respect to the evaluated point.

In Fig. 5.2(b), we illustrate this modification by drawing the lateral motion field with blue arrows, together with the "non holonomic" field "far" from the locus of in-site rotations.

5.4 Results

In this section, we present three experiments. In the first one, we test the performance of our approach in the absence of localization uncertainty, and with only non-holonomic motion. In the second one, we introduce localization uncertainty. In the last one, we present an improvement to the border behavior with the possibility of holonomic motion. The three experiments have the same set-up, with initial position in $(0.5, 2.5)$, landmark position in $(0, 0)$ and final

position in $(-3, 0)$. We chose this trajectory because it makes the robot pass through several regions of the partition and illustrates the border behavior which is one of the problems we found.

In Fig. 5.3, we depict the performance in perfect and noiseless conditions. We see that using the planner vector field, for driving the robot to the goal produces smooth and stable trajectories of the CoM. However, in real conditions, the robot will not perform the control exactly e.g., because of sliding with the floor. Moreover the robot needs a localization system which will be inherently noisy. To model this situation, we perturbed the current position of the CoM with white gaussian noise $\sim \mathcal{N}(0, \sigma^2)$ in each coordinate with $\sigma = 0.2$m for translation and with $\sigma = 10$ degrees for the orientation.

In Fig. 5.4, we show the behavior in that situation. We can still appreciate a smooth trajectory in the regions far from the border of the dark blue region of the partition, where the vector field is smooth. We also notice the good behavior of the orientation angle mainly because the QP formulation controls it. The main problem of this approach also appears: The discontinuities in the partition regions in which an in-site rotation occurs. Hence, the robot may perform unnecessary maneuvers, involving in-site rotations, which is far from optimal in the case of human walking (Fig. 5.5). Moreover, in-site rotations cause high rotation speeds of the feet (Fig. 5.6), which is not desirable from a balance point of view.

To address the aforementioned problem, we present the results of the improvement introduced in Section 5.3. In Fig. 5.7, we perform the same trajectory but by introducing the possibility of holonomic motion in the region where in-site rotations would be necessary. The robot is not performing in-site rotations anymore, but a mixture of holonomic and non-holonomic motions, depending on its position with respect to the border. We can appreciate better the transition between non-holonomic and holonomic motions in Fig. 5.8.

5.5 Conclusion

In most of the current literature, the link between planning and locomotion control has been given by footsteps. In this paper, we propose a novel approach that uses directly the optimal motion synthesis derived from the planner to the control system without going through footsteps. Instead we compute them within the

Walking Pattern Generator. With this approach, we can drive the robot to a desired goal by using an external planner, not necessarily the one used in this paper. We tested our approach simulating real situations such as situations with localization noise and sliding of the feet. We have also tried to make the walking pattern more efficient by introducing the possibility of using holonomic motion. Our next objective is to test the approach introduced in this paper on the HRP-2 platform.

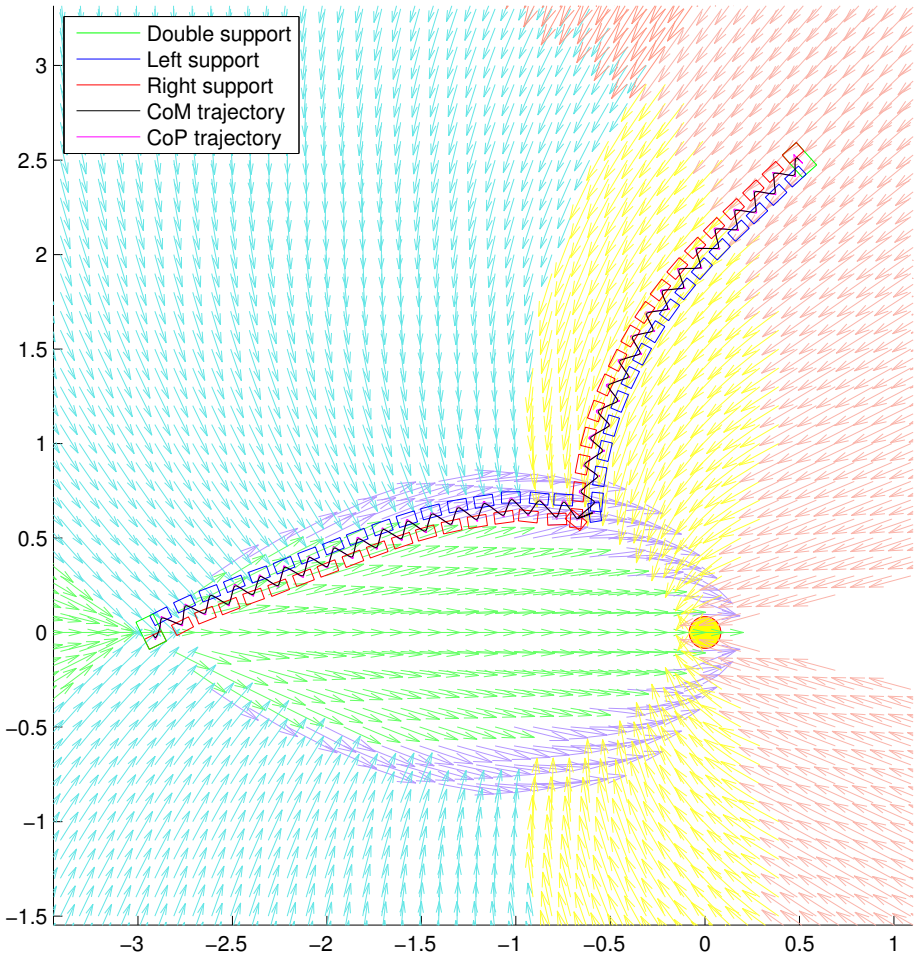


Figure 5.3: Vector fields generated by optimal trajectories planned with visibility constraints for a non-holonomic robot [Salaris et al., 2010]. To each color is associated a family of global paths that reach the goal (at $(-3, 0)$) according to the current configuration of the robot while keeping the landmark (yellow disk) visible at all times. The footsteps, CoM and CoP trajectories for the humanoid robot are all depicted as indicated in the upper box.

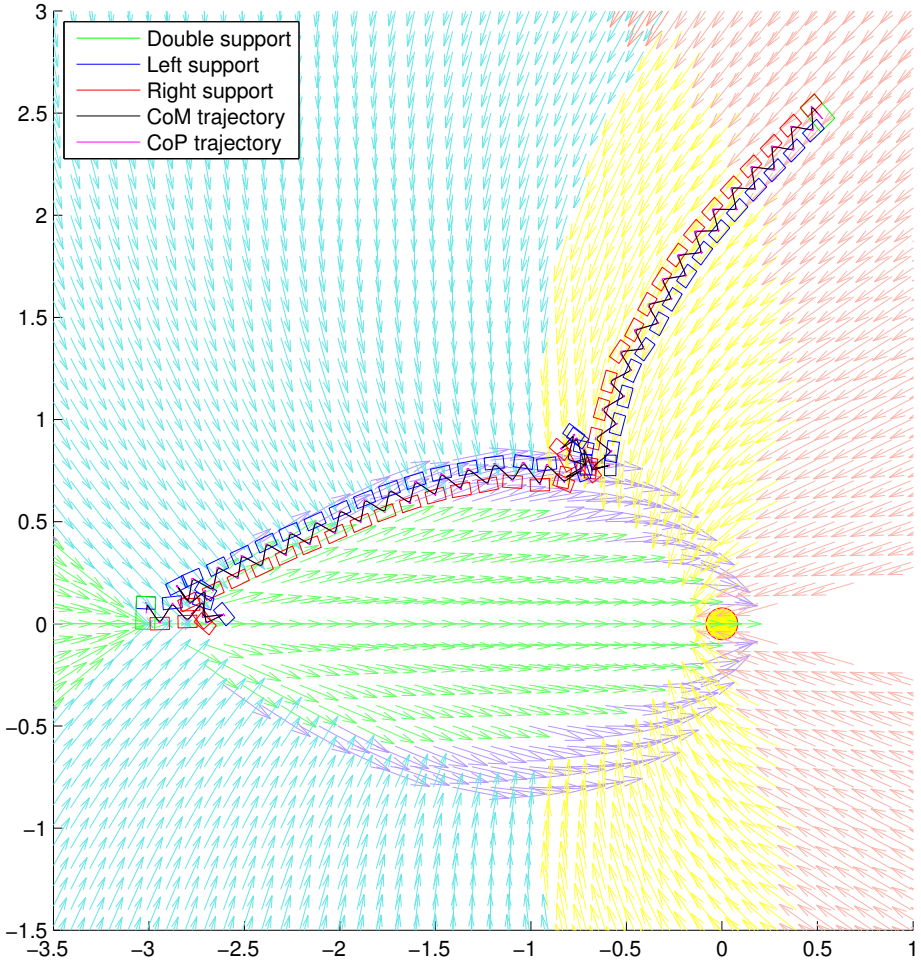


Figure 5.4: If the control is not followed well and if we use an imperfect localization system, problems may arise in the borders of the partition, in case of relying on purely non-holonomic behaviors.

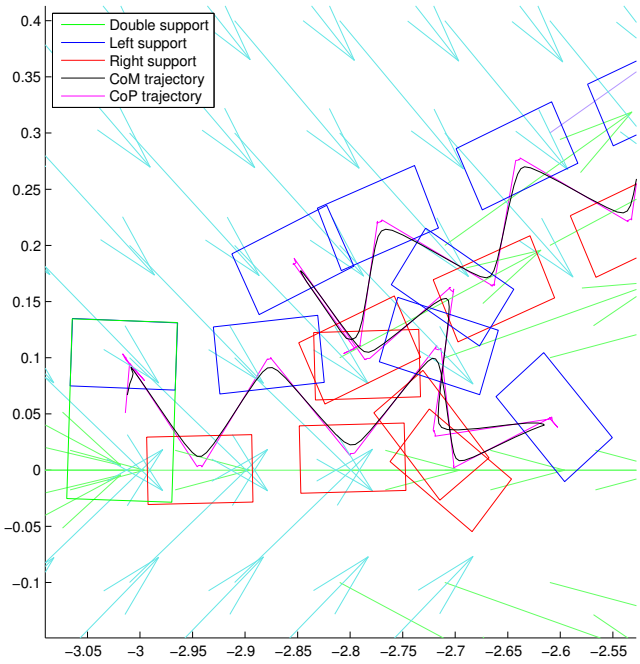


Figure 5.5: Zoom on Fig. 5.4, close to $(-3, 0)$. As the robot compensates sliding and localization errors non-holonomically, more in-site rotations occur.

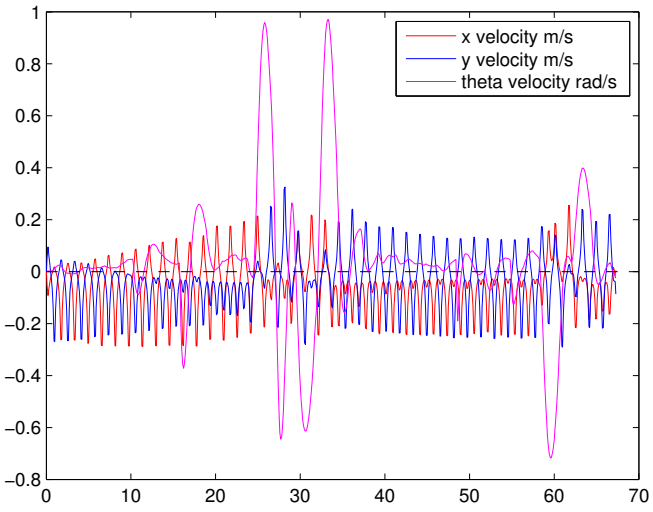


Figure 5.6: Velocities profile for the trajectory depicted in Fig. 5.4. The overshoots in the angular velocity are caused by the in-site rotations at the border of the dark blue region of the partition.

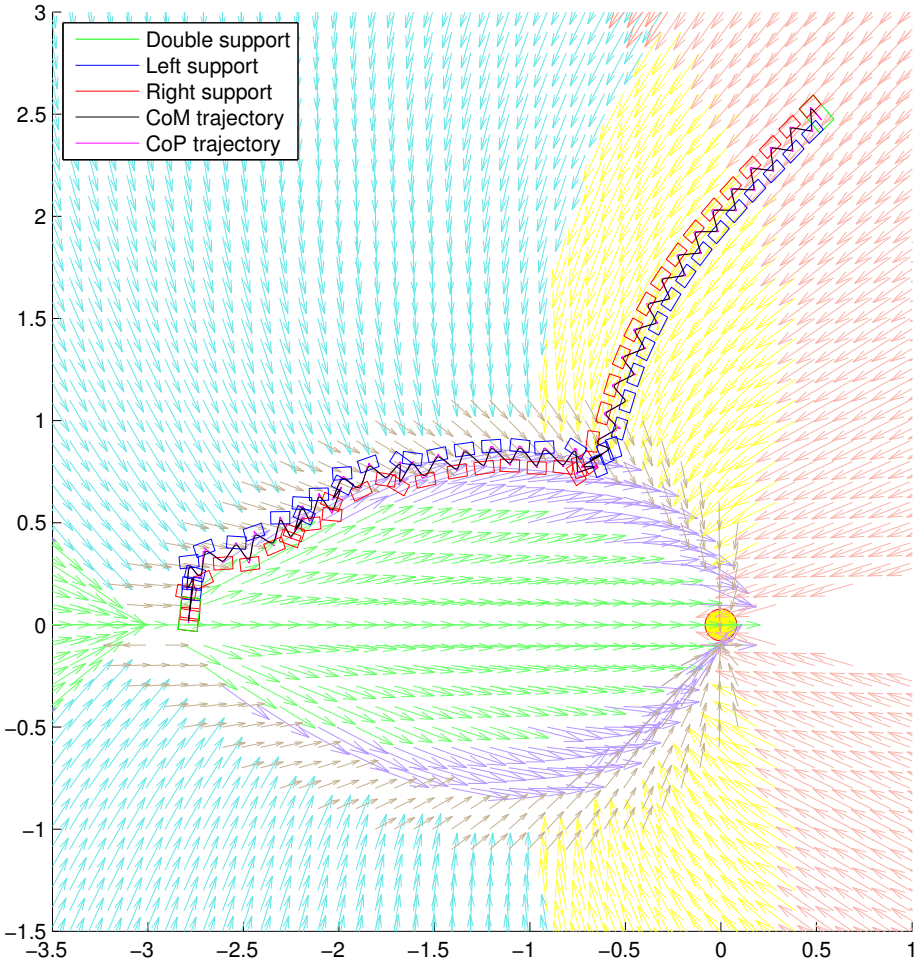


Figure 5.7: Trajectory with holonomic motion made possible. Lateral motion is allowed in the khaki vector field in the orientation discontinuity region.

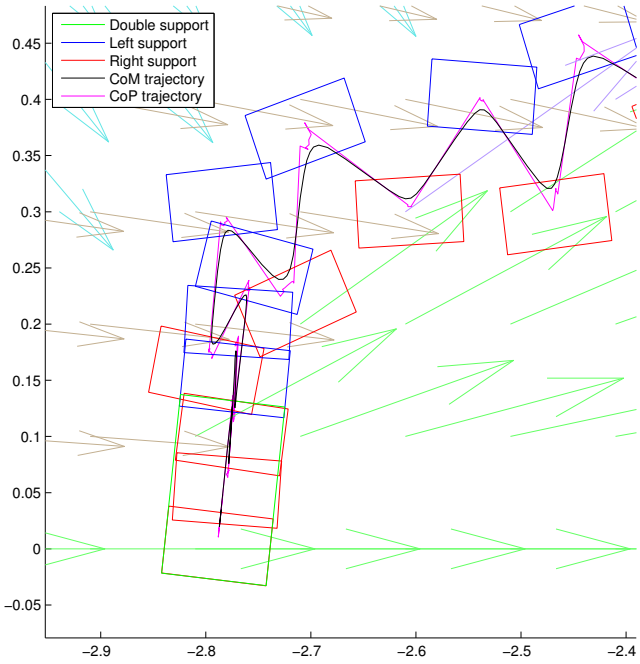


Figure 5.8: Transition between non-holonomic and holonomic controls. Moving sideways is much more efficient for such a trajectory to reach the goal.

6

Stereo Reconstruction of Dense Surfaces to Walk on Rough Terrain

As we stated in figure 1.1 computer vision can be introduced (from high to low level) in planning level, pattern generator level or whole-body motion level. In this chapter we will address the introduction of computer vision in the lowest level of robot motion generation: The Generalized Inverse Kinematics (GIK). This chapter differs from the previous ones in the sense that here we do not work at the pattern generator level. Consequently it is not directly related with the previous chapters of this thesis.

6.1 Walking on Rough terrain

In [Ramos et al., 2013] a proposal to walk on rough terrain was presented. It uses an inverse dynamics approach and a 3D reconstruction system to implement a compliant walking scheme. The inverse dynamics control scheme relies on a quadratic programming optimization solver and is used to let each foot go from

its initial to its final position, controlling also the center of mass and the waist. A 3D model reconstruction of the ground is obtained through the robot cameras located on its head as a stereo vision pair. The model allows the system to know the ground structure where the swinging foot is going to step on. This is needed, e.g. for all the algorithms we mentioned in Chapters 3-5. Thus, contact points can be handled to adapt the foot position to the ground conditions.

The inverse dynamics scheme is based on three tasks: (i) a task for tracking the position of the CoM given by the Pattern Generator, (ii) a task to partially track the waist trajectory, this task controls the height of the waist as well as its orientation and, (iii) an interpolation task, this task takes the swinging foot from its initial position to its desired final position, the foot reacts in a compliant way if a contact is detected before arriving to the final position on the ground.

The contact points detection is done using a 3D model of the floor in front of the robot. This 3D model comes from a stereo reconstruction system which we will explain in more details in next section.

6.2 Stereo reconstruction

To perform the dense reconstruction of the floor surface in front of the robot, we rely on a real-time approach similar to KinectFusion algorithm proposed in [Newcombe et al., 2011]. This approach, originally developed for a RGB-D sensor, models 3-D surfaces as zero-valued level sets of functions defined over the workspace volume. These functions are referred to as truncated signed distance function (TSDF) and they are built incrementally, by integrating the depth measurements the sensor provides, frame after frame. TSDF are defined in the 3D space, and their value is the signed distance to the closest obstacle, as depicted in Fig. 6.1. Here, we extend this approach, initially proposed for RGB-D depth data to disparity data generated from a stereo head. Although the stereo data is noisier than the one from RGB-D sensors, it is a passive sensor and can be used outdoors in sunlight conditions.

Consider, as in the previous sections, that k is a discretized time index. The idea is to update a mathematical representation of the surface through a volumetric TSDF model (defined over a 3D grid), referred to as F_k. The basic steps for integrating one new set of disparity measurements at time k, to update

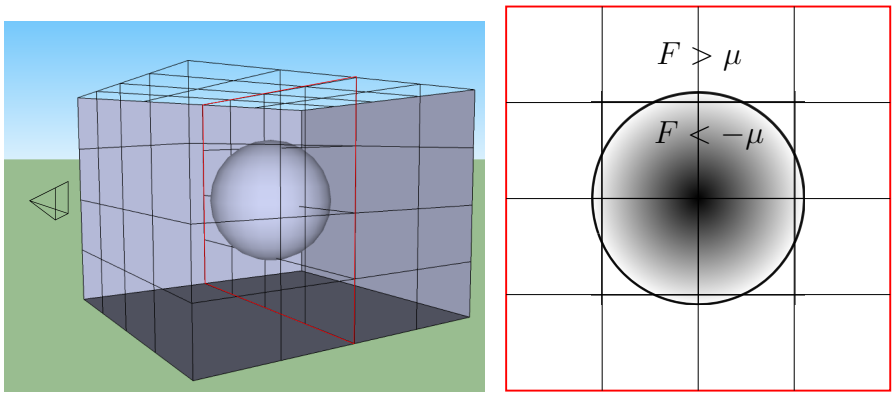


Figure 6.1: Consider the workspace of the left image. We discretize this space in a volumetric grid. And we assign to each voxel its truncated signed distance to the closest obstacle in the ray direction from the sensor, i.e. the surface, right image.

F_k and the corresponding surface, are the following ones: (i) Filter the raw depth measurements generated from the stereo head, D_k; here we used bilateral filtering for that purpose; (ii) From these filtered measurements and the prediction of the estimated surface at the previous step, estimate the transformation between the measured surface and the predicted one using the Iterative Closest Point (ICP) algorithm and update the camera pose; (iii) Compute a volumetric grid formed from "local" TSDF values F_{D_k}, to which confidence weights W_{D_k} are associated, and integrate them into the global volumetric grid $\{F_k, W_k\}$; (iv) Predict a new surface for the next iteration by using ray-casting over the zero-crossings of the fused global volumetric grid $\{F_k, W_k\}$.

To use this algorithm with stereo data and generate local data D_k, a disparity map from a pair of rectified images is estimated, from which the depth map D_k is derived, supposing the stereo rig is completely calibrated. The literature of algorithms that estimate disparity maps is huge, but since a real time one is needed for this application, the one proposed in [Geiger et al., 2010] has been used. This algorithm estimates a piece-wise disparity map using an initial sparse disparity map of high textured points as vertices that define a triangulation of the image. Then, the dense disparity map of each sub-region is estimated by using the initial sparse disparity map as a prior in a probabilistic scheme. The steps of the reconstruction process are illustrated in Fig. 6.2

6.2.1 Pre-processing

As mentioned above, it is necesary to apply a bilateral filter to the raw depth map to smooth it while preserving the borders. Then we project the pixel $\mathbf{u}$ to obtain a 3D point $\mathbf{p}$. With the projection of all the pixels, we generate a vertex map,

$$\mathbf{V}_k(\mathbf{u}) = D_k(\mathbf{u})\mathbf{K}^{-1}(\mathbf{u}), \tag{6.1}$$

where $\mathbf{K}^{-1}(\mathbf{u})$ is the ray corresponding to the pixel $\mathbf{u}$. Since the depth measurements is a regular map, we can compute the normal vectors $\mathbf{N}_k(\mathbf{u})$ using the neighbours.

6.2.2 Reconstruction

The core of this algorithm is the computation and fusion of volumetric grids (i.e., the third step mentioned above). For a 3D point $\mathbf{p}$ expressed in the global frame g, its value in the current local volumetric grid $\{F_{D_k}, W_{D_k}\}$ is computed as

$$\begin{cases} F_{D_k}(\mathbf{p}) & = & \Psi(\lambda^{-1} \left\|\mathbf{t}_{g,k} - \mathbf{p}\right\| - D_k(\mathbf{x})), \\ W_{D_k}(\mathbf{p}) & \propto & \cos(\theta)/D_k(\mathbf{x}), \end{cases}$$

with

$$\Psi(\eta) = \begin{cases} \min(1, \frac{\eta}{\mu})\ \mathrm{sgn}(\eta) & \text{iff } \eta \geq -\mu \\ \text{null} & \text{otherwise} \end{cases}, \ \lambda = \left\|\mathbf{K}^{-1}[\mathbf{x}^\top 1]^\top\right\|,$$

μ being a truncation distance (parameter of the algorithm), $\mathbf{x} = \pi([\mathbf{K}, \mathbf{1}]T_{g,k}^{-1}\mathbf{p}) \in \mathbb{R}^2$ being the image projection of $\mathbf{p}$. $\mathbf{K}$ is the 3×3 matrix of intrinsic parameters of the camera, π is the projection operator, $T_{g,k} = \begin{bmatrix} \mathbf{R}_{g,k} & \mathbf{t}_{g,k} \\ 0 & 1 \end{bmatrix}$ the pose of the camera, at time k, in the global frame g, and θ the angle between the associated pixel ray direction and the surface normal.

The global volumetric grid at time k is formed by the weighted average of all individual volumetric grids up to $k-1$. It can be shown that the optimal grid can be obtained incrementally using a simple point-wise on-line weighted average,

$$\begin{aligned} F_k(\mathbf{p}) &= \frac{W_{k-1}(\mathbf{p})F_{k-1}(\mathbf{p}) + W_{D_k}(\mathbf{p})F_{D_k}(\mathbf{p})}{W_{k-1}(\mathbf{p}) + W_{D_k}(\mathbf{p})}, \\ W_k(\mathbf{p}) &= W_{k-1}(\mathbf{p}) + W_{D_k}(\mathbf{p}). \end{aligned}$$

With the previous equation, we integrate incrementally the current measurement $\{W_{D_k}(\mathbf{p}), F_{D_k}(\mathbf{p})\}$ into the previous estimation of the grid $\{W_{k-1}(\mathbf{p}), F_{k-1}(\mathbf{p})\}$ to get current estimation of the grid $\{W_k(\mathbf{p}), F_k(\mathbf{p})\}$

6.2.3 Surface prediction

With the latest reconstruction we can compute a dense surface prediction by rendering the surface at the zero level of the TSDF on a virtual camera with the current estimation of $T_{g,k}$. This process will give us an estimated vertex and normal maps $\hat{\mathbf{V}}_k$ and $\hat{\mathbf{N}}_k$ that we will use in the camera pose estimation. The vertex map is predicted by marching each ray $T_{g,k}K^{-1}\mathbf{u}$ and stopping when a zero crossing ($+\nu e$ to $-\nu e$ for the visible side) is found from visible side to non-visible side indicating surface interface. It may find a back face or exit of the working volume so there is no surface measurement in that cases. The normal map is computed as the gradient of the surface interface at $\mathbf{p}$ using numerical derivative:

$$R_{g,k}\hat{\mathbf{N}}_g^k(\mathbf{u}) = \nu\nabla F(\mathbf{p}), \nabla F(\mathbf{p}) = \left[\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}\right]^T \tag{6.2}$$

further this derivative is scaled.

For the marching ray, skipping, which provides useful acceleration is used. This is done by marching along the ray in steps of size $< \mu$ while values of $F(\mathbf{p})$ have $+\nu e$ truncated values, so a step μ must pass at least one non-truncated value before stepping over the zero crossing. Then to refine the intersection we take F_t^+ and $F_{t+\Delta t}^+$ which are the trilinearly interpolated values either side of the zero crossing along the ray t and $t + \Delta t$ from its starting point. Then the parameter at which the intersection occurs more precisely is computed,

$$t^* = t - \frac{\Delta t F_t^+}{F_{t+\Delta t}^+ - F_t^+} \tag{6.3}$$

so we have a vextex and normal maps in the interpolated point in the global frame.

6.2.4 Sensor pose estimation

The tracking is done by aligning the current surface measurement ($\mathbf{V}_k$,$\mathbf{N}_k$) against the model predicted from the previous frame ($\hat{\mathbf{V}}_{k-1}$,$\hat{\mathbf{N}}_{k-1}$). In first place, a projective data association algorithm is used to get a set of vertex correspondences $\{\mathbf{V}_k(\mathbf{u}), \hat{\mathbf{B}}_{k-1}(\hat{\mathbf{u}}) \mid \Omega(\mathbf{u}) \neq null\}$ by computing the perspectively projected point $\hat{\mathbf{u}} = \pi(K\tilde{T}_{k-1.k}\dot{\mathbf{V}}_k(\mathbf{u}))$ using an estimate of the frame transform and testing the predicted and measured vertex and normal for compatibility.

An iterative solution $\tilde{T}^z_{g,k}$ is obtained by minimizing a linearized version of the global point-plane energy around the previous estimate $\tilde{T}^{z-1}_{g,k}$. Using the small angle assumption for an incremental transform $\tilde{T}^z_{inc}$, the update is $\tilde{T}^z_{g,k} = \tilde{T}^z_{inc}\tilde{T}^{z-1}_{g,k}$. The minimization is done using the incremental point transfer $\tilde{\mathbf{V}}^g_k(\mathbf{u}) = \tilde{T}^{z-1}_{g,k}\dot{\mathbf{V}}_k(\mathbf{u})$. Arranging the parameters of the transformation as $\mathbf{x} = (\beta, \gamma, \alpha, t_x, t_y, t_z)^T$ we can write

$$\tilde{T}^z_{g,k}\dot{\mathbf{V}}_k(\mathbf{u}) = \tilde{R}^z\tilde{\mathbf{V}}^g_k(\mathbf{u}) + \tilde{\mathbf{t}}^z = \mathbf{G}(\mathbf{u})\mathbf{x} + \tilde{\mathbf{V}}^g_k(\mathbf{u}). \tag{6.4}$$

An iteration is obtained by solving

$$\min_{\mathbf{x}\in\mathbb{R}^6} \sum_{\Omega_k(\mathbf{u})\neq null} \|E\|_2^2 \tag{6.5}$$

$$E = \tilde{\mathbf{N}}^g_{k-1}(\mathbf{u})^T \left(\mathbf{G}(\mathbf{u})\mathbf{x} + \tilde{\mathbf{V}}^g_k(\mathbf{u}) - \hat{\mathbf{V}}^g_{k-1}(\hat{\mathbf{u}})\right) \tag{6.6}$$

The minimum can be found analytically by derivating the objective function and setting the derivative to zero. A 6×6 symmetric linear system is generated for each correspondence. Solving for $\mathbf{x}$, we obtain $\tilde{T}^z_{g,k}$. The data association and pose estimation is embedded into a coarse to fine framework using the bottom 3 levels of the vertex and normal maps pyramid, the camera pose results of the last iteration.

6.3 Results

For the implementation, we adapted the Kinfu algorithm which is provided with the Point Cloud Library (PCL). We also used the stereo matching algorithm which is provided in the authors web page [1]. We made the experiments with the stereo rig which is in the head of the HRP-2 humanoid robot. In order to do that, we used the Robot Operating System (ROS) framework as a middleware to link the different parts.

In Fig. 6.2 we depict the input and output of the stages of the algorithm. In Figs. 6.2, 6.3 we show the results with the scene of flat floor and some objects on it, like books and small boxes. In Fig. 6.4 we tested the algorithm in a stairs scene with the small objects as well. Finally, in Fig. 6.5 we depict a fully dynamic simulation of the HRP-2 robot while walking on rough terrain using the reconstruction system presented in this chapter.

6.4 Conclusion

In this chapter we presented a combination of existing algorithms for efficient stereo matching and 3D reconstruction to be used in a more complex system to make the robot walk on rough terrain. This connection was done over ROS and tested successfully with scenes of flat floor and stairs with small objects on them. The output of this algorithm provides a model of the world to a control system which implements a compliant walking scheme.

[1] http://www.cvlibs.net/software/libelas/

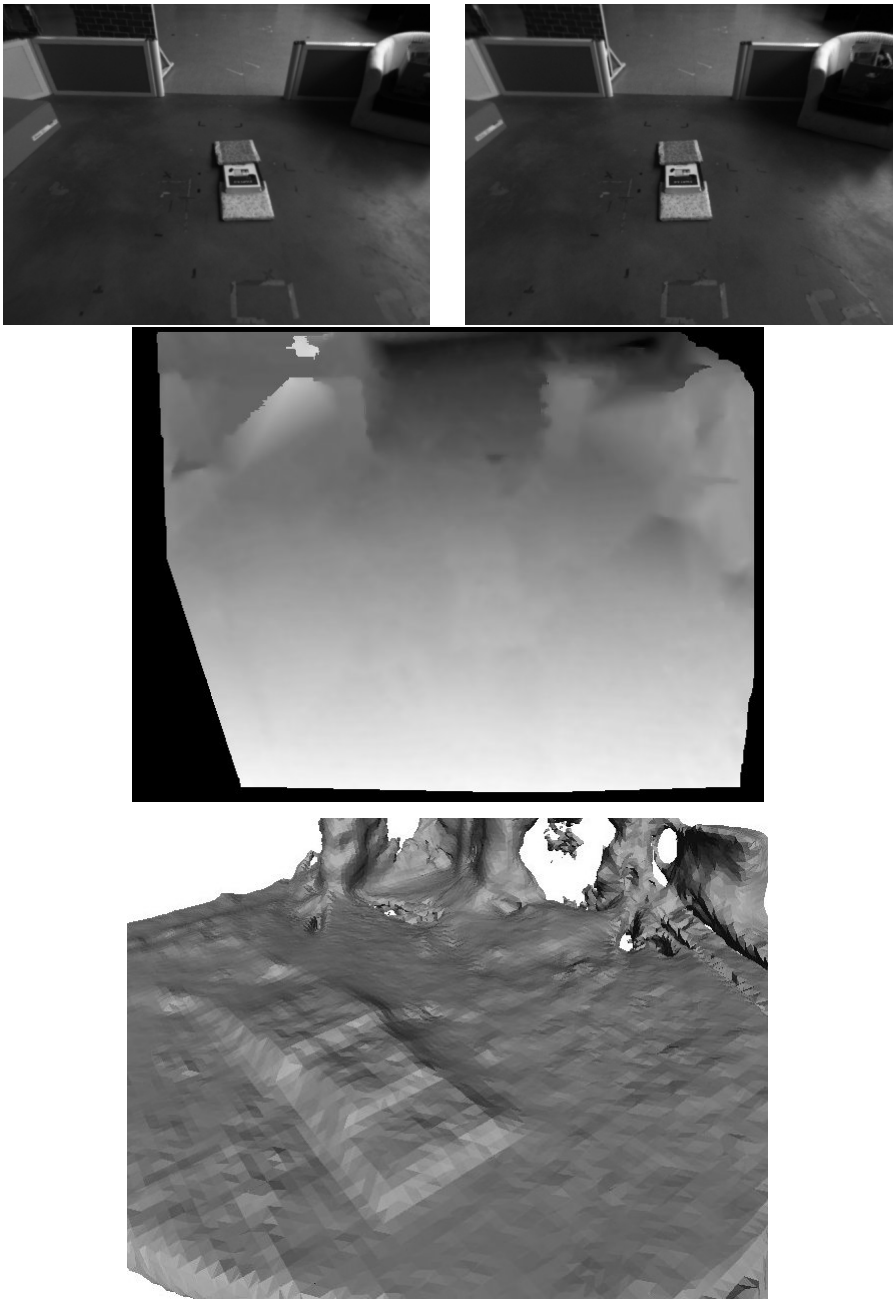


Figure 6.2: From a pair of images of the scene in front of the robot (up) we estimate a dense disparity map (middle) and from this disparity map we estimate a dense surface (bottom) integrating the previous frames into the volumetric grid.

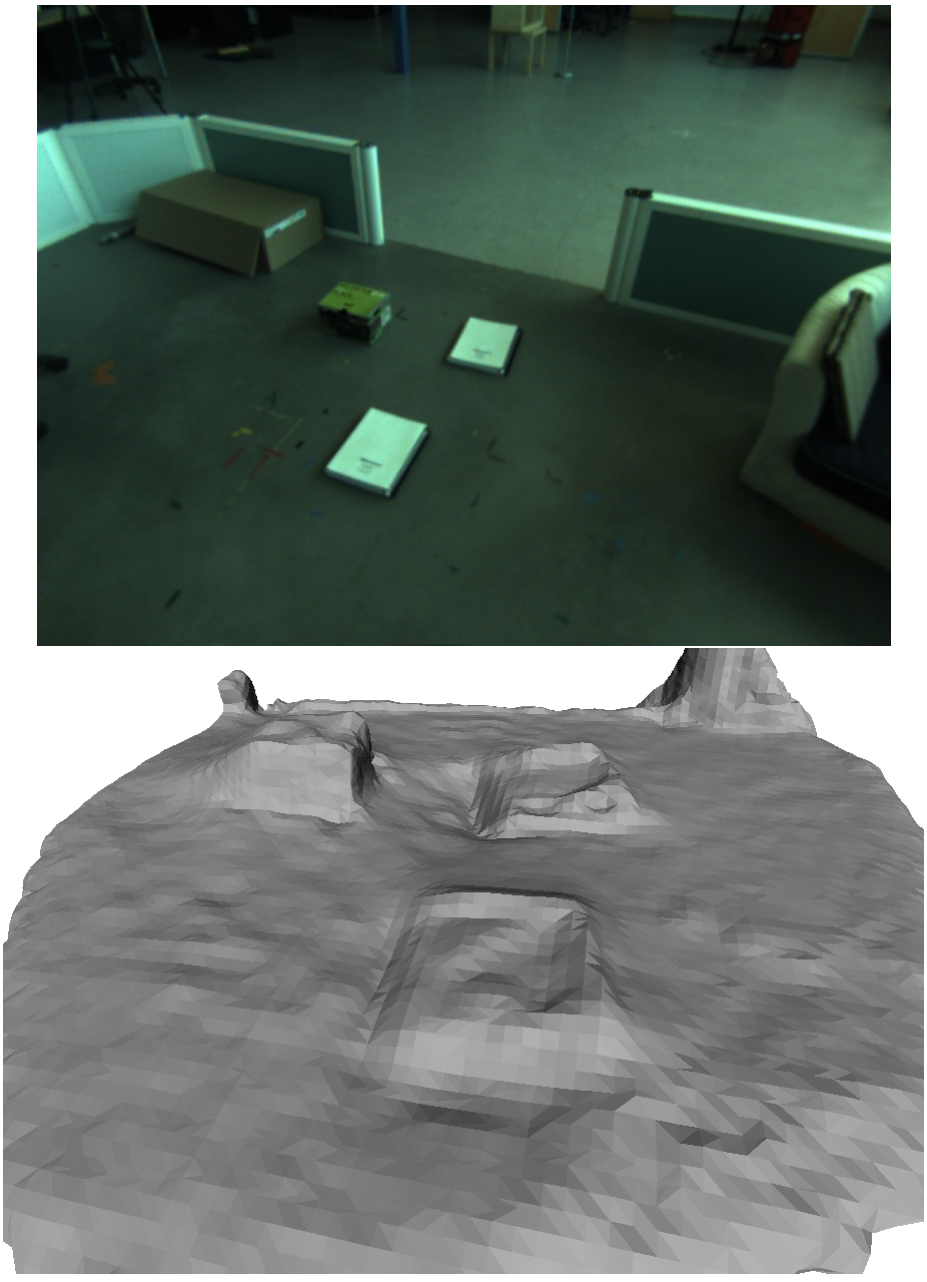


Figure 6.3: Reconstruction of a scene of flat floor with small objects on it like books and boxes.

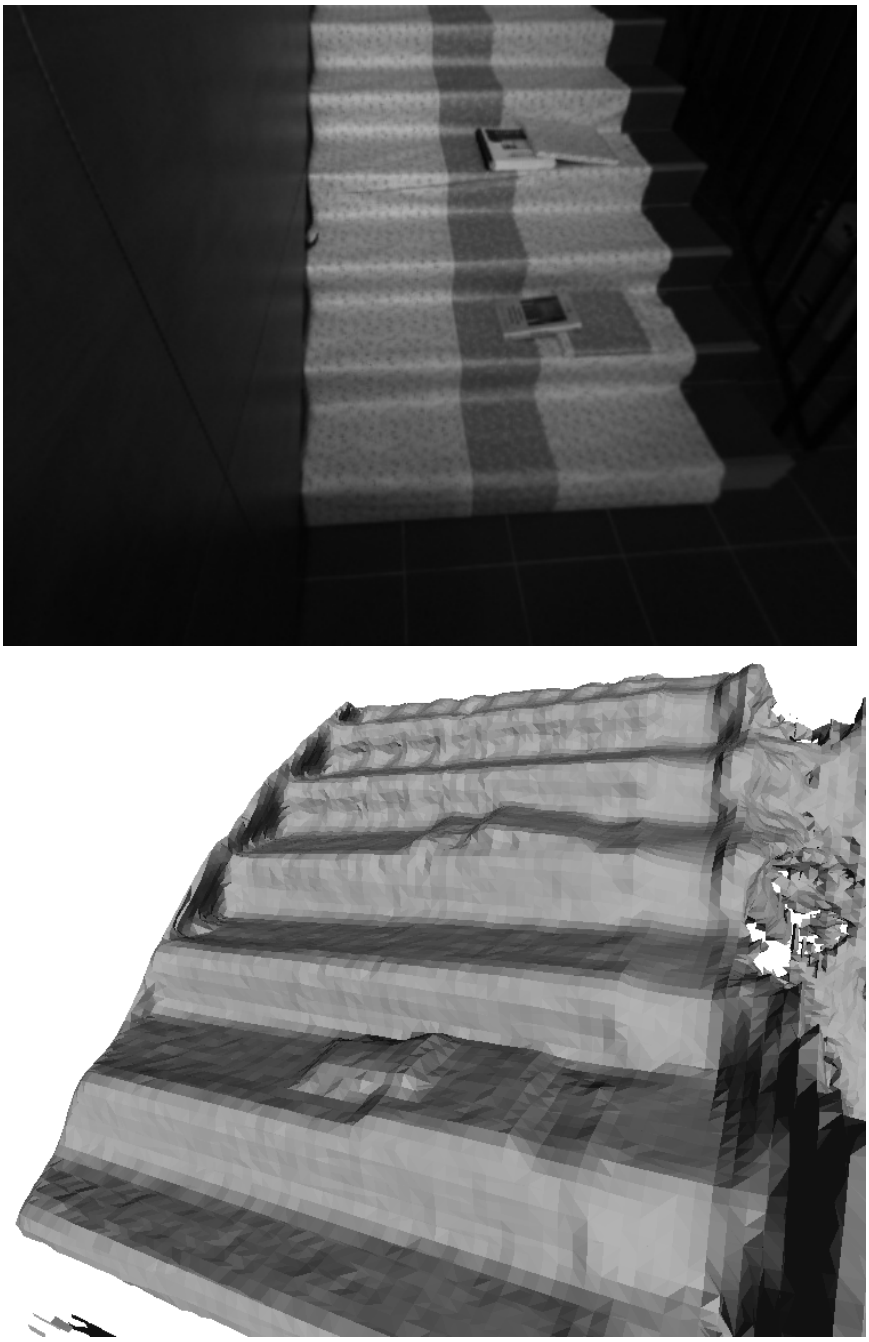
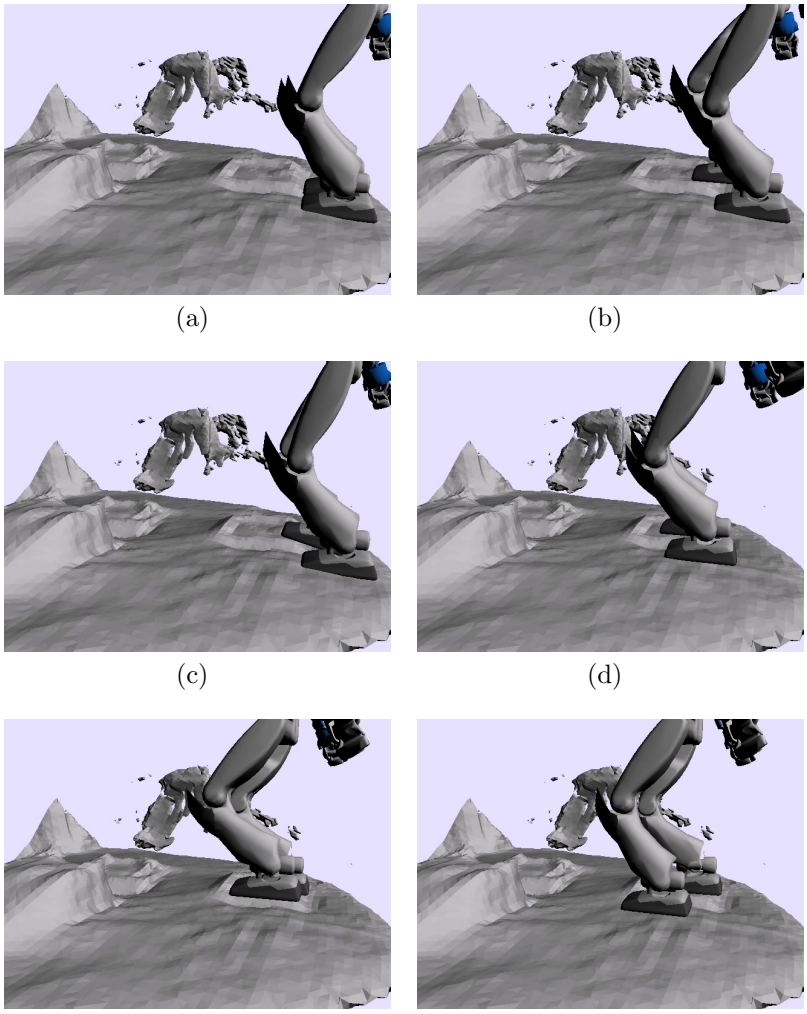


Figure 6.4: Reconstruction of a stairs scene with some objects on it.



(a) (b) (c) (d) (e) (f)

Figure 6.5: HRP-2 walking on an obstacle.

7 Conclusions

In this thesis, we tackled the problem of humanoid walking using visual information. Following the perception-decision-action paradigm, in this thesis we partially addressed all of them. A chain of improvements has been done since the ZMP Preview Control scheme was first introduced in [Kajita et al., 2003]. We have built on the work of prominent previous works and proposed new approaches to improve the humanoid walking.

7.1 Contributions

We presented a visual servoing scheme at the pattern generator level using Model Predictive Control. In the case of humanoid robots, visual servoing is a very useful approach for controlling precisely the robot position in contexts such as interactions with humans (to set the robot in front of a person), or to pre-position the robot before starting some manipulation tasks. The advantage of visual servoing is that the positioning task is defined relatively to a specific target. In any of the aforementioned applications, the higher-level navigation

can be advantageously left to a planner, that would determine the sequence of landmarks, or human interactions, to reach consecutively.

Since Visual Servoing is a local controller, we extended the scope of the pattern generation by directly introducing planning. Traditionally, planning was introduced at the pattern generator level using footprints. In our approach, we directly injected the motion primitives to the patterns generator using a velocity reference guided by a vector field.

In fact, these two paradigms are more complimentary than opposite: whereas planning may require high computational costs, but solve more complex problems of path finding, visual servoing is fast, by essence, but in general confined to local tasks that do not require high level reasoning, e.g. positioning the robot with respect to a given object.

Both contributions are based on Linear Model Predictive Control. However, due to the new optimization techniques and new computation performance, Nonlinear Model Predictive Control should be explored. This way we could use the whole-body motion instead of inverted pendulum simplification and moreover, deal with more accurate models.

Paralelly to the work of the pattern generation, we also presented an implementation of the combination of stereo matching and 3D reconstruction algorithms to be used by a walking control scheme. This reconstruction system was successfully tested on the HRP-2 robot and is used into a dynamic simulation to walk on rough terrain.

7.2 Perspectives

Our final goal is that the humanoid robots can achieve highly reactive motions in uncontrolled environments. Even if there has been impressive demonstrations of humanoid robots performing reactive motions based on perception, they are done in controlled environments. In current robotics, perception is one of the most important challenges.

In the control side, there already exists very suitable models of the humanoid locomotion and control. Those models are mainly stated as optimization problems. However, today's computers and algorithms can not solve in real time those complex models. Consequently, we rely in simplified but useful models to

make the robots walk. Moreover, the introduction of torque controlled actuators can considerably improve the performance of humanoid motion.

Considering the current humanoid robotics development and challenges, we would like to continue contributing in humanoid locomotion based on vision. Always taking into account state of the art perception algorithms, optimization solvers and mathematical models of humanoid walking.

Bibliography

[Allibert et al., 2010] G. Allibert, E. Courtial, and F. Chaumette. Visual servoing via nonlinear predictive control. In G. Chesi and K. Hashimoto, editors, *Visual Servoing via Advanced Numerical Methods*, volume 401 of *Lecture Notes in Control and Information Sciences*, pages 375–393. Springer London, 2010. ISBN 978-1-84996-088-5.

[Bourgeot et al., 2002] J.-M. Bourgeot, N. Cislo, and B. Espiau. Path-planning and tracking in a 3d complex environment for an anthropomorphic biped robot. In *IROS*, volume 3, pages 2509–2514, 2002.

[Chaumette and Hutchinson, 2006] F. Chaumette and S. Hutchinson. Visual servo control (i). Basic approaches. *IEEE Robotics Automation Magazine*, 13(4): 82–90, 2006. ISSN 1070-9932. doi: 10.1109/MRA.2006.250573.

[Chaumette and Hutchinson, 2007] F. Chaumette and S. Hutchinson. Visual servo control (ii). Advanced approaches. *IEEE Robotics Automation Magazine*, 14 (1):109–118, 2007. ISSN 1070-9932. doi: 10.1109/MRA.2007.339609.

[Chestnutt et al., 2005] J. Chestnutt, M. Lau, G. Cheung, J.J Kuffner, J. Hodgins, and T. Kanade. Footstep planning for the honda asimo humanoid. In *ICRA*, pages 629–634, 2005.

[Chestnutt et al., 2007a] J. Chestnutt, K. Nishiwaki, J. Kuffner, and S. Kagami. An adaptive action model for legged navigation planning. In *ICRA*, pages 196–202, 2007a.

[Chestnutt et al., 2007b] J.E Chestnutt, P. Michel, J.J. Kuffner, and T. Kanade. Locomotion among dynamic obstacles for the honda asimo. In *IROS*, pages 2572–2573, 2007b.

[Dang et al., 2011] D. Dang, F. Lamiraux, and J-P. Laumond. A framework for

manipulation and locomotion with realtime footstep replanning. In *ICHR*, pages 676–681, 2011.

[Dune et al., 2010] C. Dune, A. Herdt, O. Stasse, P. B Wieber, K. Yokoi, and E. Yoshida. Cancelling the sway motion of dynamic walking in visual servoing. In *IROS*, pages 3175–3180, 2010. doi: 10.1109/IROS.2010.5649126.

[Geiger et al., 2010] A. Geiger, M. Roser, and R. Urtasun. Efficient large-scale stereo matching. In *Asian Conf. on Computer Vision*, Queenstown, New Zealand, November 2010.

[Gutmann et al., 2008] J.-S. Gutmann, M. Fukuchi, and M. Fujita. 3d perception and environment map generation for humanoid robot navigation. *International Journal of Robotics Research*, 27(10):1117–1134, 2008.

[Hayet et al., 2012] J.-B. Hayet, C. Esteves, G. Arechavaleta, O. Stasse, and E. Yoshida. Humanoid locomotion planning for visually-guided tasks. *Int. Journal of Humanoid Robotics*, 09:1–26, 2012.

[Herdt et al., 2010a] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl. Online walking motion generation with automatic footstep placement. *Advanced Robotics*, 24(5-6):719–737, 2010a.

[Herdt et al., 2010b] A. Herdt, N. Perrin, and P.-B Wieber. Walking without thinking about it. In *IROS*, pages 190–195, 2010b. doi: 10.1109/IROS.2010.5654429.

[Hornung and Bennewitz, 2012] A. Hornung and M. Bennewitz. Adaptive level-of-detail planning for efficient humanoid navigation. In *ICRA*, pages 997–1002, 2012.

[Kajita et al., 1992] S. Kajita, Tomio Yamaura, and A. Kobayashi. Dynamic walking control of a biped robot along a potential energy conserving orbit. *Robotics and Automation, IEEE Transactions on*, 8(4):431–438, 1992. ISSN 1042-296X. doi: 10.1109/70.149940.

[Kajita et al., 2003] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *ICRA*, pages 1620–1626, 2003.

[Kanoun et al., 2011] O. Kanoun, J.-P. Laumond, and E. Yoshida:. Planning foot placements for a humanoid robot: A problem of inverse kinematics. *IJRR*, 30(4):476–485, 2011.

[Lorch et al., 2002] O. Lorch, A. Albert, J. Denk, M. Gerecke, R. Cupec, J. Seara, W. Gerth, and G. Schmidt. Experiments in vision-guided biped walking. In *IROS*, pages 2484–2490, 2002.

[Michel et al., 2007] P. Michel, J. Chestnutt, S. Kagami, K. Nishiwaki, J. Kuffner, and T. Kanade. Gpu-accelerated real-time 3d tracking for humanoid locomotion and stair climbing. In *IROS*, pages 463–469, October 2007.

[Mombaur et al., 2008] K. Mombaur, J.-P. Laumond, and E. Yoshida. An optimal control model unifying holonomic and nonholonomic walking. In *ICHR*, pages 648–653, 2008.

[Newcombe et al., 2011] R.A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A.J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *IEEE Int. Symp. on Mixed and Augmented Reality (ISMAR)*, pages 127–136, Washington, DC, USA, 2011.

[Nishiwaki and Kagami, 2009] K. Nishiwaki and S. Kagami. Online walking control system for humanoids with short cycle pattern generation. *IJRR*, 28(6):729–742, 2009.

[Ramos et al., 2013] O. E. Ramos, M. Garcia, N. Mansard, O. Stasse, J.-B. Hayet, and P. Soueres. Towards reactive vision-guided walking on rough terrain: An inverse-dynamics based approach. *IJHR Submitted*, 2013.

[Salaris et al., 2010] P. Salaris, D. Fontanelli, L. Pallottino, and A. Bicchi. Shortest paths for a robot with nonholonomic and field-of-view constraints. *IEEE Trans. on Robotics*, 26:269–281, 2010.

[Vahrenkamp et al., 2009] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner, and R. Dillmann. Humanoid motion planning for dual-arm manipulation and re-grasping tasks. In *IROS*, pages 2464–2470, 2009.

[Vukobratovic and Borovac, 2004] M. Vukobratovic and B. Borovac. Zero-moment point thirty five years of its life. *International Journal of Humanoid Robotics*, 01(01):157–173, 2004. doi: 10.1142/S0219843604000083. URL `http://www.worldscientific.com/doi/abs/10.1142/S0219843604000083`.

[Vukobratovic and Stepanenko, 1972] M. Vukobratovic and J. Stepanenko. On the stability of anthropomorphic systems. *Mathematical Biosciences*, 15:1–37, 1972.

[Wieber, 2006] P. B Wieber. Trajectory free linear model predictive control for stable walking in the presence of strong perturbations. In *Humanoid Robots,*

2006 6th IEEE-RAS International Conference on, pages 137–142, 2006. doi: 10.1109/ICHR.2006.321375.

[Yi et al., 2011] Seung-Joon Yi, Byoung-Tak Zhang, Dennis Hong, and D.D. Lee. Online learning of a full body push recovery controller for omnidirectional walking. In *ICHR*, pages 1–6, 2011.