2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Visually-Guided Grasping while Walking on a Humanoid Robot

Nicolas Mansard, Olivier Stasse, François Chaumette, Kazuhito Yokoi

Abstract— **In this paper, we apply a general framework for building complex whole-body control for highly redundant robot, and we propose to implement it for visually-guided grasping while walking on a humanoid robot. The key idea is to divide the control into several sensor-based control tasks that are simultaneously executed by a general structure called *stack of tasks*. This structure enables a very simple access for task sequencing, and can be used for task-level control. This framework was applied for a visual servoing task. The robot walks along a planed path, keeping the specified object in the middle of its field of view and finally, when it is close enough, the robot grasps the object while walking .**

I. Introduction

In this paper, we describe a complete implementation of a complex whole-body motion to realize a visually-guided grasping while walking on a humanoid robot. It is based on a general framework for building whole-body control for highly redundant robot, that is easily modulable and can be adapted for a vast class of robotic problems.

One of the first approach to generate full-body motion considering a human-size humanoid robot is motion planning. Proposed by Kuffner et al. [11] it relies on a discretization of the possible foot steps for walking, and a general path planning algorithm for manipulation. By choosing a reasonable number of foot placement it is possible to plan footstep in a dynamic environment based on vision feedback [16]. In [17], a whole body motion based on a combination of several postures is planned to reach a distant point with the arm end-effector. Those remarkable results however did not address the problem of manipulation while walking, and the problem of motion generation based on sensor feedback.

In answer to this problem, we mainly focus on implementing a complex sensor-based reactive full-body control on a humanoid robot. Sensor-feedback control loop techniques, such as visual servoing [6], [4] provide very efficient solutions to control robot motions. It supplies high positioning accuracy, good robustness to sensor noise and calibration uncertainties, and reactivity to environment changes.

Very few work can be found for sensor-based control of a whole-body humanoid robot. In [26] visual servoing is used to position the leg of a HOAP-1 Fujitsu humanoid robot. Several works have been proposed to solve a vision-based manipulation task on humanoid torso or on non-walking robot [3], [24], [13]. All these works demonstrate the efficiency of sensor-based reactive control for developing robust and accurate task for humanoid robots. However, none of them was extended for full-body motion generation.

Such framework has been proposed by Sentis et al. in [19]. It integrates task-oriented dynamic control and control prioritization. Impressive results have been demonstrated on simulation. Another framework has been proposed by Sian et al. [20] which relies on the use of Kajita's preview control for the walking and on Resolved Momentum Control (RMC) to make sure that the tasks demanded at the upper body level keep the robot balance. As the final goal of this work is to teleoperate a HRP-2 humanoid robot, real-time whole-body motion generation with short cycle is necessary. This framework has been proved very reliable during several days at the AICHI 2005 universal exposition.

In this work we propose to implement visual servoing for full-body motion generation on a humanoid robot, using a similar framework that was already proposed and validated for arm manipulator robot [15]. This framework is called task sequencing. Like the frameworks presented above, it enables simple definitions of a complex task. Using the easy access on the low-level controller, a task-level controller has also been designed to deal with obstacles. The general idea is to sequence a set of tasks to extend the local convergence domain of the reactive visual-based control schemes, through the use of a general structure called stack of task, that ensures the task prioritization along with the motor input continuity at task change. This work is based on earlier works such as switching control laws [5], [2] and task sequencing [22], [18] that use a task-level reasoning controller to modify the reactive-level control loop, in order to extend the convergence domain and avoid obstacles. We will prove in the following that the task sequencing framework is very suitable for whole-body motion generation of a humanoid robot, and that it allows to implement complex tasks such as grasping while walking, by a simple and very efficient way.

The next section will recall the task sequencing framework, as a generic solution for humanoid control. Section III presents the application of this framework to the HRP-2 robot. As an application, we propose to implement a visually-guided grasping while walking, taking into account the joint limits of the robot. The experiments on the real robot are finally presented in Section IV.

N. Mansard and F. Chaumette are with IRISA / INRIA Rennes, France {Nicolas.Mansard,Francois.Chaumette}@irisa.fr
O. Stasse and K. Yokoi are with JRL ISRI/AIST-CNRS Tsukuba, Japan {Olivier.Stasse,Kazuhito.Yokoi}@aist.go.jp

II. General control method

The general control method used to realize a full-body-motion sensor-based task is first presented. It is easily modulable, and could be adapted very efficiently for various type of task. The control law is computed using a general structure called *stack of tasks* [14] that is able to apply simultaneously several tasks while ordering them to avoid conflicts. This structure enables a very easy high-level control access, by providing a simple way to activate or inactivate any task during the execution to modify the robot behavior.

In the following, the stack of tasks structure is recalled, in a generic way. We will insist on the use of the stack of task for whole-body motion generation, to automatically compensate the motions due to the walk that could perturb the manipulation task.

A. Stack of tasks

The stack of tasks is a structure that orders the tasks currently active. Only the tasks in the stack are taken into account in the control law. The task at the bottom level has priority over all the others, and the priority decreases as the stack level increases. The control law is computed from the tasks in the stack, in accordance with three rules:

- any new task added in the stack does not disturb the tasks already in the stack.
- the control law is continuous, even when a task is added or removed from the stack.
- if possible, the additional constraints should be added to the control law, but without disturbing the tasks in the stack.

The control law is computed from the stack, using the redundancy formalism introduced in [21]. The additional constraints are added at the very top of the stack, which means that they are taken into account only if some degrees of freedom (DOF) remain free after applying the active tasks. This priority order may seem illogical, considering that the constraints are obstacles that the robot should avoid. However, the positioning task has priority since it is the task we want to see completed, despite the obstacles. The high-level controller is then used to ensure that the constraints are respected when it is obvious that the robot will violate them.

1) Ensuring the priority: Let $(\mathbf{e_1}, \mathbf{J_1})$ ... $(\mathbf{e_n}, \mathbf{J_n})$ be n tasks. The control law computed from these n tasks should ensure the priority, that is the task $\mathbf{e_i}$ should not disturb the task $\mathbf{e_j}$ if $i > j$. A recursive computation of the articular velocity is proposed in [21]:

$$\begin{cases} \mathbf{\dot{q}_0} = 0 \\ \mathbf{\dot{q}_i} = \mathbf{\dot{q}_{i-1}} + (\mathbf{J_i P^A_{i-1}})^+(\mathbf{\dot{e}_i} - \mathbf{J_i \dot{q}_{i-1}}), \quad i = 1..n \end{cases} \tag{1}$$

where $\mathbf{P^A_i}$ is the projector onto the null-space of the augmented Jacobian $\mathbf{J^A_i} = (\mathbf{J_1}, \dots \mathbf{J_i})$ and $\widetilde{\mathbf{J}}_\mathbf{i} = \mathbf{J_i P^A_{i-1}}$ is the limited Jacobian of the task i . The robot articular velocity realizing all the tasks in the stack is $\mathbf{\dot{q}} = \mathbf{\dot{q}_n}$.

2) Ensuring the continuity: ¿From (1), the control law is obtained by imposing a reference velocity $\mathbf{\dot{e}_i}$ for each task in the stack. Generally, an exponential decrease is required by imposing the first order differential equation $\mathbf{\dot{e}_i} = -\lambda_i \mathbf{e_i}$. However, this equation does not ensure the continuity of the robot velocity when the stack is changed. In [14], we proposed a solution to properly smooth the robot velocity at the transition, by imposing a specific second order equation:

$$\mathbf{\ddot{e}_i} + (\lambda_i + \mu)\ \mathbf{\dot{e}_i} + (\lambda_i \mu)\ \mathbf{e_i} = 0 \tag{2}$$

where λ_i is the gain that tunes the convergence speed of task $\mathbf{e_i}$, and μ sets the transition smoothness of the global control law. The control law is obtained by introducing (2) in (1):

$$\begin{cases} \mathbf{\dot{q}_i} = \mathbf{\dot{q}_{i-1}} + (\mathbf{J_i P^A_{i-1}})^+(-\lambda_i \mathbf{e_i} - \mathbf{J_i \dot{q}_{i-1}}) \\ \mathbf{\dot{q}} = \mathbf{\dot{q}_n} + e^{-\mu(t-\tau)}\ \big(\mathbf{\dot{e}}(\tau) + \Lambda \mathbf{e}(\tau)\big) \end{cases} \tag{3}$$

where τ is the time of the last modification of the stack.

3) Adding the secondary constraints: The constraints are added using the Gradient Projection Method [12], [10]. The constraints are described by a cost function V. The gradient $\mathbf{g}(\mathbf{q})$ of this cost function can be considered as an artificial force, pushing the robot away from the undesirable configurations. It is introduced as the last task of the stack. It has thus to be projected onto the null space of each task into the stack. Using (3), the complete control law is finally

$$\mathbf{\dot{q}} = \mathbf{\dot{q}_n} + e^{-\mu(t-\tau)}\ \big(\mathbf{\dot{e}}(\tau) + \Lambda \mathbf{e}(\tau)\big) - \kappa \mathbf{P^A_n g} \tag{4}$$

The reader is invited to refer to [14], [15] for more details.

B. Pattern generator

We re-implemented a pattern generator similar to the one commercially available on the HRP-2 robot, based on [7]. From the footsteps given as an input, the center of mass (CoM) trajectory is generated using a 3D linear inverted pendulum model of the robot whose CoM moves on a plane. The key is to solve an inverse problem from the ZMP reference position deduced from the footsteps. This inverse problem is solved using a preview controller and thus implies to know the future in the corresponding window (here 1.6 seconds). In order to take into account the real model of the robot, Kajita proposed to use a second stage of preview control to compensate the difference between the ZMP of the multibody model and the ZMP of the inverted pendulum. This second stage of preview control is extremely efficient but adds 1.6 seconds of the future to be known, and thus in total 3.2 seconds of the future are needed. Therefore the main problem regarding a reactive control loop is to integrate the immediate command generated by the task using the upper body and the pattern generator to maintain stability.

Some solutions to this problem are the RMC [8] and the CoM Jacobian [23]. In both cases, the main idea is to cancel disturbances of the CoM reference trajectory by using the remaining DOF. Full body motion generation based on RMC has been already realized in [20] but did not integrate a task prioritization as in this work. In previous works, we have tested experimentally the capabilities of the pattern generator together with the stabilizer to cope with violations of the

CoM's planar motion and small disturbances of the ZMP. It allowed us to make the first humanoid dynamically stepping over obstacles [25]. For sake of simplicity and as a first step we implemented a simple heuristic where the left arm is used to compensate partially for the perturbation induced by the stack of tasks implemented. For this reason we did not integrate the chest as a free joint to stay in the area of stability. Our future work will integrate COG Jacobian to use fully the capabilities of the system.

C. Using the stack of task to compensate the inner motions due to the walk

At each iteration, the pattern generator produces the next reference position that should be reached by the robot. The walking behavior can thus be written as a task function:

$$\mathbf{e_{walk}} = \mathbf{q_{leg}} - \mathbf{q_{leg}}^* \tag{5}$$

where $\mathbf{q_{leg}}$ is the current articular position of the two legs and $\mathbf{q_{leg}}^*$ is the reference position produced by the pattern generator. The jacobian $\mathbf{J_{walk}}$ is very simply:

$$\mathbf{J_{walk}} = \begin{bmatrix} \mathbf{I_{nleg}} & \mathbf{0_{nleg}} & \mathbf{0_{n-2nleg}} \\ \mathbf{0_{nleg}} & \mathbf{I_{nleg}} & \mathbf{0_{n-2nleg}} \end{bmatrix} \tag{6}$$

where n is the total number robot joints, and $n_{leg} = 6$ is the number of joints of each leg. As shown by (5), the walking task uses the 12-DOF of the legs and no redundancy is available for any other task. The other tasks can be realized using the upper-body joints. Let us first consider a controller whose only entries are the upper-body articular velocity $\dot{\mathbf{q}}_{\mathbf{up}}$. Let $\mathbf{e_{up}}$ be a task function whose jacobian $\mathbf{J_{up}^{up}} = \frac{\partial \mathbf{e_{up}}}{\partial \mathbf{q_{up}}}$ is full rank. If the support leg moves (for example while walking), then the task $\mathbf{e_{up}}$ is perturbed. The time derivative of the task error can thus be written:

$$\dot{\mathbf{e}}_{\mathbf{up}} = \mathbf{J_{up}^{up}} \dot{\mathbf{q}}_{\mathbf{up}} + \frac{\partial \mathbf{e_{up}}}{\partial t} \tag{7}$$

where $\frac{\partial \mathbf{e_{up}}}{\partial t}$ are all the motions that are not due to the upper body. In the present case, this motions are equals to the perturbation due to the support-leg motions. They can be written $\frac{\partial \mathbf{e_{up}}}{\partial t} = \frac{\partial \mathbf{e_{up}}}{\partial \mathbf{q_{leg}}} \dot{\mathbf{q}}_{\mathbf{leg}}$. The control law that executes the reference task motion $\dot{\mathbf{e}}_{\mathbf{up}}^*$ while compensating the leg motions can finally be written:

$$\dot{\mathbf{q}}_{\mathbf{up}} = \mathbf{J_{up}^{up}}^+ \big(\dot{\mathbf{e}}_{\mathbf{up}}^* - \frac{\partial \mathbf{e_{up}}}{\partial \mathbf{q_{leg}}} \dot{\mathbf{q}}_{\mathbf{leg}} \big) \tag{8}$$

We suppose now that the stack state is $\big(\mathbf{e_{walk}} \ \mathbf{e_{up}} \big)$. Let us prove that the stack of task is able to generate exactly the same compensation of the upper-body motion due to the walk. The full-body jacobian $\mathbf{J_{up}}$ can be decomposed in two parts by separating the legs from the rest of the body:

$$\mathbf{J_{up}} = \begin{bmatrix} \mathbf{J_{up}^{leg}} \ \mathbf{J_{up}^{up}} \end{bmatrix} \tag{9}$$

where $\mathbf{J_{up}^{leg}} = \frac{\partial \mathbf{e_{up}}}{\partial \mathbf{q_{leg}}}$. For the two tasks in the stack, the control law (1) can be simply written:

$$\dot{\mathbf{q}} = \mathbf{J_{walk}^+} \dot{\mathbf{e}}_{\mathbf{walk}} + \big(\mathbf{J_{up}} \mathbf{P_{walk}} \big)^+ \Big(\dot{\mathbf{e}}_{\mathbf{up}} - \mathbf{J_{up}} \mathbf{J_{walk}^+} \dot{\mathbf{e}}_{\mathbf{walk}} \Big) \tag{10}$$

Since $\mathbf{J_{walk}} = \big[\mathbf{I_{2nleg}} \ \mathbf{0} \big]$, the projector is:

$$\mathbf{P_{walk}} = \begin{bmatrix} \mathbf{0_{2nleg}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{11}$$

Eq. (10) can finally be written:

$$\begin{aligned} \dot{\mathbf{q}} &= \begin{bmatrix} \dot{\mathbf{e}}_{\mathbf{walk}} \\ \mathbf{0} \end{bmatrix} + \mathbf{J_{up}^{up}}^+ \big(\dot{\mathbf{e}}_{\mathbf{up}} - \mathbf{J_{up}} \mathbf{J_{walk}^+} \dot{\mathbf{e}}_{\mathbf{walk}} \big) \\ &= \begin{bmatrix} \dot{\mathbf{e}}_{\mathbf{walk}} \\ \mathbf{J_{up}^{up}}^+ \big(\dot{\mathbf{e}}_{\mathbf{up}} - \mathbf{J_{up}} \dot{\mathbf{q}}_{\mathbf{leg}} \big) \end{bmatrix} \end{aligned} \tag{12}$$

where $\dot{\mathbf{q}}_{\mathbf{leg}} = \dot{\mathbf{e}}_{\mathbf{walk}}$. The second part of control vector (which corresponds to the upper-body motion) is equal to (8). This last result proves that the stack of task is appropriate to generate full-body motion by automatically compensating any motions due to the walk.

D. Execution controller

The stack of tasks provides a very simple solution to allow control at the task level. In [15], we have proposed a complete solution that ensures the respect of several constraints during the execution of a non-redundant task on a manipulator robot. Here this solution is applied for ensuring the joint limits avoidance while walking and grasping.

As explained in the previous sections, the constraints are applied at the top of the stack. They could thus be respected only locally, and nothing ensures that some tasks of the stack will not violate them. To ensure that the constraints are never violated, a task-level controller has been designed. The controller detects the collision by a linear prediction. It then chooses the best task to be removed according to an optimal criteria proposed in [15]. The optimal DOF is thus freed up, and can be used to avoid the collision.

A second controller was added to ensure the realization of the global task when far enough from the constraints. The second controller detects that the collision has been avoided by a second prediction phase, and pushes back the removed tasks in the stack as soon as possible. The reader is invited to refer to [15] for more details.

III. Application to grasping

We now present how the task sequencing framework can be used for a specific task. We have implemented a grasping based on visual servoing. Thanks to the stack structure that intrinsically compensates the motion due to the walk, the robot is able to grasp an object while walking. Thanks to the robustness of visual servoing, it was even possible to grasp a slowly-moving object.

A good estimate of the object position is obtained using the two stereo cameras mounted on the robot head. The first task is to keep the object centered in the image by visual servoing during all the humanoid motion, to ensure its visibility. The second task brings the robot gripper at the object position so that it can grasp it. Finally, the joint limits constraint is taken into account through the stack of tasks. In the following, we will present this three tasks, along with the high-level rules that have been used to realize the grasping and to ensure the joint-limit avoidance.

The robot input controller is the full-body joint velocity:

$$\dot{\mathbf{q}} = \left(\dot{\mathbf{q}}_{\mathbf{ll}}, \dot{\mathbf{q}}_{\mathbf{rl}}, \dot{\mathbf{q}}_{\mathbf{chest}}, \dot{\mathbf{q}}_{\mathbf{neck}}, \dot{\mathbf{q}}_{\mathbf{rarm}}, \dot{\mathbf{q}}_{\mathbf{larm}}\right) \qquad (13)$$

where $\dot{\mathbf{q}}_{\mathbf{rl}}$, $\dot{\mathbf{q}}_{\mathbf{ll}}$, $\dot{\mathbf{q}}_{\mathbf{chest}}$, $\dot{\mathbf{q}}_{\mathbf{neck}}$, $\dot{\mathbf{q}}_{\mathbf{rarm}}$ and $\dot{\mathbf{q}}_{\mathbf{larm}}$ are the joint velocities of the left leg, right leg, chest, neck, right and left arm respectively.

A. Visual servoing

A visual servoing task is based on an error $\mathbf{e_i}$ defined as the difference between the current value of a visual feature $\mathbf{s_i}$ observed in the image, and its desired value $\mathbf{s_i^*}$ [4]:

$$\mathbf{e_i} = \mathbf{s_i} - \mathbf{s_i^*} \qquad (14)$$

where $\mathbf{s_i}$ is the current value of the visual features for subtask $\mathbf{e_i}$ and $\mathbf{s_i^*}$ their desired value. The interaction matrix $\mathbf{L_{s_i}}$ related to $\mathbf{s_i}$ is defined so that $\dot{\mathbf{s_i}} = \mathbf{L_{s_i}}\mathbf{v}$, where $\mathbf{v}$ is the instantaneous camera velocity. From (14), it is clear that the interaction matrix $\mathbf{L_{s_i}}$ and the task Jacobian $\mathbf{J_i}$ are linked by the relation:

$$\mathbf{J_i} = \mathbf{L_{s_i}}\mathbf{M}\mathbf{J_q} \qquad (15)$$

where the matrix $\mathbf{J_q}$ denotes the robot Jacobian ($\dot{\mathbf{r}} = \mathbf{J_q}\dot{\mathbf{q}}$) and $\mathbf{M}$ is the matrix that relates the variation of the camera velocity $\mathbf{v}$ to the variation of the chosen camera pose parametrization ($\mathbf{v} = \mathbf{M}\dot{\mathbf{r}}$).

B. Centering task

In order to ensure the object visibility during the servo, and to stabilize the image motion to improve the image processing, the image of the object is centered in one of the camera view. The centering task is thus simply:

$$\mathbf{e_G} = \mathbf{p_{left}} \qquad (16)$$

where $\mathbf{p_{left}} = \left(x_G,\ y_G\right)$ is the current position of the object centroid in the left camera image. The interaction matrix of $\mathbf{e_G}$ is the well known interaction matrix of the point [4]. Finally, the full-body jacobian of the centering task is:

$$\mathbf{J_{qG}} = \left[\ {}^{\mathbf{cam}}\mathbf{J_{sl}} \quad {}^{\mathbf{cam}}\mathbf{J_{chest}} \quad {}^{\mathbf{cam}}\mathbf{J_{neck}} \quad \mathbf{0} \quad \mathbf{0}\ \right] \qquad (17)$$

where ${}^{\mathbf{cam}}\mathbf{J_{sl}}$, ${}^{\mathbf{cam}}\mathbf{J_{chest}}$ and ${}^{\mathbf{cam}}\mathbf{J_{neck}}$ are the jacobians of the support leg, the chest and the neck respectively, computed in the left camera frame. If the right leg is on the ground, the jacobian of the support leg is:

$${}^{\mathbf{cam}}\mathbf{J_{sl}} = \left({}^{\mathbf{cam}}\mathbf{T_{rfoot}}\,{}^{\mathbf{rfoot}}\mathbf{T_{waist}}\mathbf{J_{rleg}} \quad \mathbf{0}\right) \qquad (18)$$

where ${}^{\mathbf{cam}}\mathbf{T_{rfoot}}$ and ${}^{\mathbf{rfoot}}\mathbf{T_{waist}}$ are the twist matrices from the right foot frame (right-leg end effector) to the camera frame and from the waist frame to the camera frame respectively. The matrix $\mathbf{J_{rleg}}$ is the jacobian of the right leg computed in the waist frame. The opposite construction of ${}^{\mathbf{cam}}\mathbf{J_{sl}}$ is done if the right foot is in flight.

C. Grasping task

The grasping task is mainly a 3D-positioning of the right-hand gripper at the object position. However, to ensure that the gripper will be properly oriented when grasping, we have chosen to dissociate the positioning task in two parts. The first part controls the orientation of the gripper, the second part controls the distance to the object.

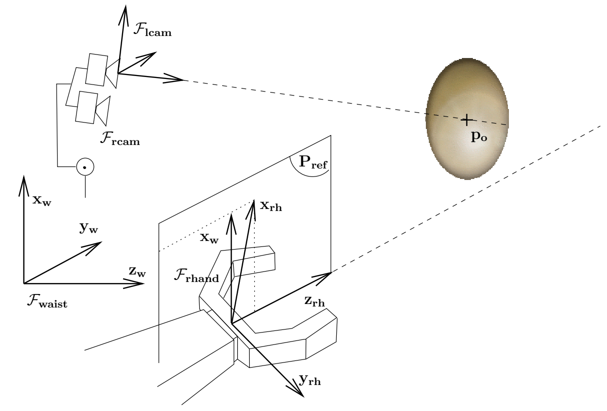


Fig. 1. Definition of frames $\mathcal{F}_{\mathbf{rhand}}$ and $\mathcal{F}_{\mathbf{lcam}}$.

1) Gripper orientation control: The end effector of the right hand is noted $\mathcal{F}_{\mathbf{rhand}}$ (see Fig. 1). The origin of this frame is set at the center of the grip, and the Z-axis is set to correspond to the opening of the gripper. To properly grasp the object, it has to be be kept in front of the gripper opening, that is to say on the Z-axis. We note $\mathbf{p_o} = \left(\ X_o, Y_o, Z_o\right)$ the center of the object expressed in $\mathcal{F}_{\mathbf{rhand}}$. The orientation task is thus:

$$\mathbf{e}_\theta = \left[\begin{array}{c} X_o \\ Y_o \end{array}\right] \qquad (19)$$

The interaction matrix of this task can be obtained by derivation of $\mathbf{e}_\theta$:

$$\dot{\mathbf{e}_\theta} = \left[\begin{array}{c} \dot{X_o} \\ \dot{Y_o} \end{array}\right] = \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \end{array}\right]\dot{\mathbf{p_o}} \qquad (20)$$

We note $\mathbf{v_{rhand}} = \left(\boldsymbol{v}, \boldsymbol{\omega}\right)$ the cartesian velocity of frame $\mathcal{F}_{\mathbf{rhand}}$ ($\mathbf{v_{rhand}} = {}^{\mathbf{rh}}\mathbf{J_q}\dot{\mathbf{q}}$). The velocity of $\mathbf{p_o}$ with respect to the arm velocity is given by:

$$\dot{\mathbf{p_o}} = -\boldsymbol{v} + \boldsymbol{\omega} \times \mathbf{p_o} = -\boldsymbol{v} - \widetilde{\mathbf{p_o}}\boldsymbol{\omega} = \left[\ \mathbf{I_3} \quad \widetilde{\mathbf{p_o}}\ \right]\mathbf{v} \qquad (21)$$

where $\widetilde{\mathbf{p_o}}$ is the cross-product matrix of $\mathbf{p_o}$. By introducing (21) into (20), the interaction matrix $\mathbf{L}_\theta$ of $\mathbf{e}_\theta$ is finally obtained:

$$\mathbf{L}_\theta = \left[\begin{array}{cccccc} -1 & 0 & 0 & 0 & -Z_o & Y_o \\ 0 & -1 & 0 & Z_o & 0 & -X_o \end{array}\right] \qquad (22)$$

The articular jacobian of $\mathbf{e}_\theta$ is the right-arm end-effector jacobian with respect to the joints of the support leg, the chest and the right arm. Its computation is similar to (17) and is thus left to the reader.

2) Gripper position control: When the task $\mathbf{e}_\theta$ is active and realized, the remaining DOF to grasp the object is controlled by the distance between the gripper and the object. We have chosen to use the task defined by:

$$\mathbf{e_{3D}} = \mathbf{p_{rhand}} - \mathbf{p_o} \qquad (23)$$

where $\mathbf{p_{rhand}}$ and $\mathbf{p_o}$ are computed in the same frame. The frame $\mathcal{F}_{\mathbf{rhand}}$ has been chosen as a common frame (the error is thus $\mathbf{e_{3D}} = -{}^{\mathbf{rhand}}\mathbf{p_o}$). The interaction matrix is thus simply the identity matrix $\mathbf{L_{3D}} = \left[\mathbf{I_3}\ \mathbf{0_3}\right]$, and the articular jacobian is the same than the articular jacobian of task $\mathbf{e}_\theta$.

The task $\mathbf{e_{3D}}$ is thus strongly coupled to the task $\mathbf{e}_\theta$. However, thanks to the priority order provided by the stack of tasks, an artificial decoupling is imposed that ensures that the two tasks will not conflict during the servo. For that, we simply impose that task $\mathbf{e_{3D}}$ has a lower priority than task $\mathbf{e}_\theta$.

3) Vertical orientation of the gripper: The tasks presented above only constrain three of the six DOF of the gripper. To improve the quality of the grasping, another task is introduced to control the vertical orientation of the gripper during the grasping. This is a first step toward grasping complex object.

The arm end-effector frame is noted $\mathcal{F}_{rhand} = (\mathbf{x_{rh}}, \mathbf{y_{rh}}, \mathbf{z_{rh}})$ (see Fig. 1). The task regulates the position of the plane $\mathbf{P} = (\mathbf{x_{rh}}, \mathbf{z_{rh}})$ to the vertical, that is to say to the reference plane $\mathbf{P_{ref}} = (\mathbf{x_w}, \mathbf{z_{rh}})$. The regulation of the plane is equivalent to the regulation of its normal. The normal of $\mathbf{P}$ is $\mathbf{y_{rh}}$. The error could thus be written:

$$\mathbf{e_{vert}} = \mathbf{y_{rh}} - \mathbf{y_{ref}} \tag{24}$$

where $\mathbf{y_{ref}}$ is the normal to $\mathbf{P_{ref}}$: $\mathbf{y_{ref}} = \mathbf{z_{rh}} \times \mathbf{x_w}$. The interaction matrix is given by [9]:

$$\mathbf{L_{vert}} = \left[\begin{array}{cc} \mathbf{0_3} & \widetilde{\mathbf{y_{rh}}} \end{array} \right] \tag{25}$$

It has three lines, but is always of rank 2. The articular jacobian of this task is the same than the articular jacobian of the task $\mathbf{e}_\theta$.

D. Joint limit avoidance

The joint-limit constraint is added at the top of the stack using (3). The cost function for joint-limit avoidance is defined directly in the articular space. It reaches its maximal value near the joint limits, and it is nearly constant (so that the gradient is nearly zero) far from the limits.

The robot lower and upper joint limits for each axis i are denoted $\bar{q}_i^{\min}$ and $\bar{q}_i^{\max}$. The robot configuration $\mathbf{q}$ is said acceptable if, for all i, $\mathbf{q_i} \in [\bar{q}_{\ell i}^{\min}, \bar{q}_{\ell i}^{\max}]$, where $\bar{q}_{\ell i}^{\min} = \bar{q}_i^{\min} + \rho \bar{q}_i$, $\bar{q}_{\ell i}^{\max} = \bar{q}_i^{\max} - \rho \bar{q}_i$, $\bar{q}_i = \bar{q}_i^{\max} - \bar{q}_i^{\min}$ is the length of the domain of the articulation i, and ρ is a tuning parameter, in $[0, 1/2]$ (typically, $\rho = 0.1$). $\bar{q}_{\ell i}^{\min}$ and $\bar{q}_{\ell i}^{\max}$ are activation thresholds. In the acceptable interval, the avoidance force should be zero. The cost function V^{jl} is thus given by [1]:

$$V^{jl}(\mathbf{q}) = \frac{1}{2} \sum_{i=1}^{n} \frac{{\delta_i}^2}{\Delta \bar{q}_i} \tag{26}$$

where

$$\delta_i = \begin{cases} \mathbf{q_i} - \bar{q}_{\ell i}^{\min}, & \text{if } \mathbf{q_i} < \bar{q}_{\ell i}^{\min} \\ \mathbf{q_i} - \bar{q}_{\ell i}^{\max}, & \text{if } \mathbf{q_i} > \bar{q}_{\ell i}^{\max} \\ 0, & \text{else} \end{cases}$$

E. Two high-level rules

Finally, the high-level controller is added to ensure a good execution of the complex required behavior. Two rules are applied to drive the high-level controller. The first one ensures that the joint-limit constraint is preserved during the execution. The second one ensures that the robot really tries to grasp the object only when it is close enough.

1) Balance versus joint limits avoidance: As explained upper, we have chosen not to control explicitly the balance of the robot due to the upper body motions. Indeed, the lower-body controller has been experimentally proved to be robust enough to ensure the robot balance under the constraint that the chest joints are used as little as possible. However, these joints are necessary to bring some redundancy to the upper-body tasks. In particular, the chest joints are necessary to enlarge the neck joint domain, that is very short by itself. We thus define a last task $\mathbf{e_{chest}}$ that constrains the chest joints to stay at its zero position. By introducing this task in the lower part of the stack, we ensure that none of the upper-body tasks will use the chest joints in the general case.

The high-level controller recalled in II-D is then used to ensure the neck-joint-limit avoidance, and the visibility task execution. When the neck joints are going to collide their limits, the task $\mathbf{e_{chest}}$ is automatically removed from the stack. This gives the necessary redundancy for joint-limit avoidance and task execution. When no collision is detected, the task $\mathbf{e_{chest}}$ is pushed back, which ensures that the chest joints are used as little as possible.

This task $\mathbf{e_{chest}}$ is a very *ad hoc* (but very efficient) way to ensure the robot balance. In the near future, we plan to generalize this solution by implementing a real CoM control, as done for example in [20].

2) Grasping only when close enough: This second rule is added to limit the time where the arm is fully extended. Indeed, this position is better to be avoided, because of the singular arm configuration, and also because of the disturbance caused to the robot balance. Let d_o be the distance from the shoulder to the object ($d_o = ||{}^{\mathbf{rs}}\mathbf{M_{cam}}\mathbf{p_o}||$, where ${}^{\mathbf{rs}}\mathbf{M_{cam}}$ is the homogeneous matrix passing from the camera frame to the right shoulder frame. When the object is too far ($d_o > d_{max}$, where d_{max} is the length of the arm), then the task $\mathbf{e_{3D}}$ is removed from the stack. To avoid any oscillation when d_o is close to d_{max}, the task $\mathbf{e_{3D}}$ is pushed back when d_o is below 80% d_{max}. This simple rule enables the robot to prepare the grasping when it is far from the object, by regulating the task $\mathbf{e}_\theta$ to 0, without penalizing the arm manipulability or the robot balance.

IV. Experiments and results

The robot used for the experiments is a HRP-2 humanoid robot. The control loop runs at $200Hz$. The camera feedback runs at $30Hz$. A Kalman filter is used to synchronize the two process loops. The presented experiment is a typical execution of the complete application. An object is placed in the workspace, and is moved randomly. The robot is walking along a planned trajectory that passes close to the object. While walking, the robot has to grasp the object.

The experiment is summed up in Fig. 2 to 7. The robot starts walking at iteration 2300. It arrives close enough from the object at Event (1) (see Fig. 2). The task $\mathbf{e_{3D}}$ is then added in the stack, and is quickly completed. At iteration 3800, the object is in the hand, and the robot closes its griper. All the tasks are finally removed from the stack to finish the execution, as soon as the torque sensors detects that the gripper is closed on the object. The camera sensors feedbacks are given in Fig. 3 and 4. The object moves during the robot displacement, but thanks to the closed-loop control, the error $\mathbf{e_G}$ is properly regulated and the object is kept close to the image center (see Fig. 3). Similarly, the task $\mathbf{e}_\theta$ is properly regulated to 0. The last component of $\mathbf{e_{3D}}$ has a very high value at the beginning of the servo and decreases while the robot moves forward. From Event (2), the task $\mathbf{e_{3D}}$ is added in the stack, and then quickly decreases to zero, while the hand moves to grasp the object. The task $\mathbf{e_{3D}}$ is kept then to zero while the gripper grasps the object.

The joint velocities are given in Fig. 5. During the robot approach, until Event (1), the joint velocities are small and very oscillatory, since it is only a compensation of the walking motion and of the ball motion. At Event (1), the robot accelerates to realize very quickly the task $\mathbf{e_{3D}}$ and to catch the ball. The velocities then decrease since they correspond once more to a compensation motion. The values are higher than at the beginning of the walk, since the object is much closer, and the walk perturbation have thus much more effect on the object motion in the image.

Finally, a brief overview of the experiment is given in the two last figures. Fig. 6 gives an overview of the robot motion taken from an external camera. Fig. 7 gives the corresponding snapshots taken from one of the embedded stereo camera.

V. Conclusion

In this paper, we have presented a general method to obtain complex full-body motion on humanoid robots. We propose to use a generic structure called stack of tasks, already validated for robot arm, in order to generate a full-body motion computed from several sensor-based subtasks. Using this structure, it is then very easy to sequence the subtasks to control the execution and obtain a complex behavior. We applied this solution for a grasping task while walking. The complete application was very easy to write, thanks to the use of the stack of task framework. It was then experimentally validated on the HRP-2 robot. The robot was able to grasp an object close to its walking trajectory without stopping.

As said in the text, our future work is mainly to integrate the classical solutions to ensure the upper-body robot balance. This is necessary since the robot balance is currently ensured only practically by the stabilizer that is robust enough to damp the upper-body motions. We are also focusing on introducing in the stack all necessary constraints to realize a manipulation task in realistic environment, such as obstacle or self-collision avoidance.

Acknowledgment

The authors would like to thank the JSPS Summer Program for partial funding of this work.

References

[1] F. Chaumette and E. Marchand. A redundancy-based iterative scheme for avoiding joint limits: Application to visual servoing. *IEEE Trans. on Robotics and Automation*, 17(5):719–730, October 2001.

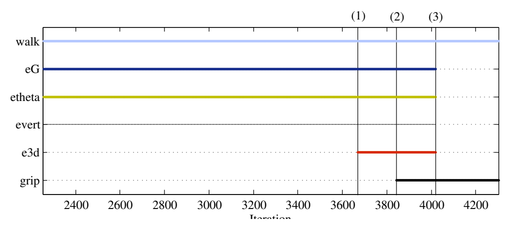


Fig. 2. Experiment B: Activation of the task in the stack during servo. At the beginning of the servo, the tasks $\mathbf{e_{walk}}$, $\mathbf{e_G}$, $\mathbf{e_{vert}}$ and $\mathbf{e}_\theta$ are in the stack. At Event (1) (Iteration 3650), the task $\mathbf{e_{3d}}$ is put in the stack to grasp the object. The object is in the gripper from Event (2) (Iteration 3800). The gripper closure starts. All the tasks are removed from the stack as soon as the gripper is close on the object, at Event (3) (Iteration 4000).

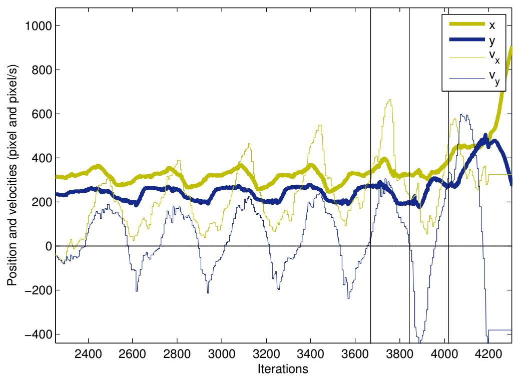


Fig. 3. Experiment B: Position and velocity of the object in the image, obtained after Kalman filtering. The filter is able to properly filter the object inner motion by integrating the robot joint velocities. The ball is kept at the center of the field of view $(320, 240)$.

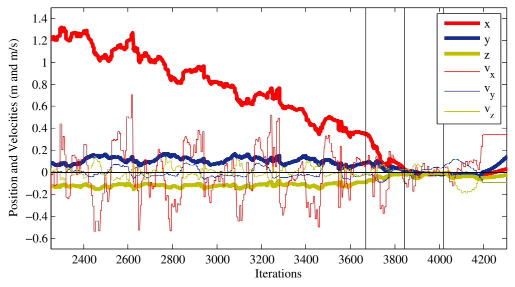


Fig. 4. Experiment B: Position and velocity of the object in the gripper frame, obtained after Kalman filtering. The two components x and y are regulated to 0 thanks to the task $\mathbf{e}_\theta$. The last component z slowly decreases to 0 while walking, and is finally regulated to 0 from iteration 3700 when the task $\mathbf{e_{3D}}$ is introduced in the stack.

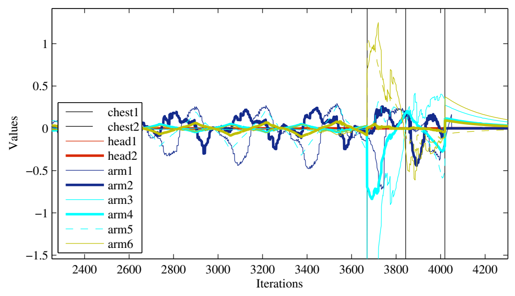
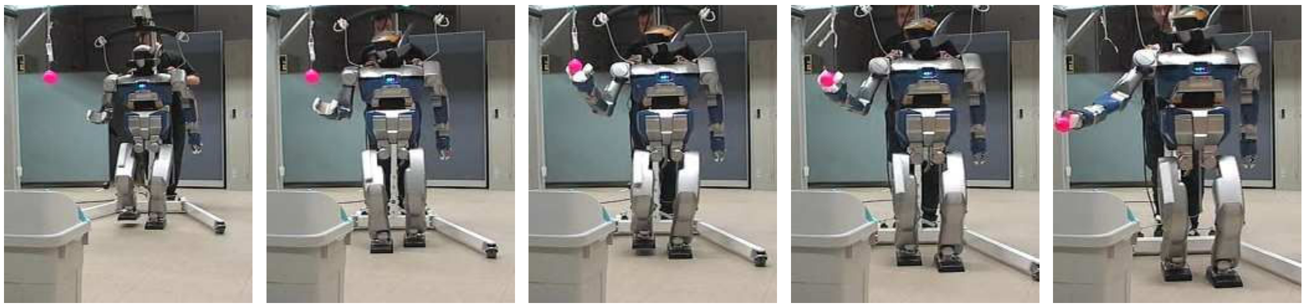
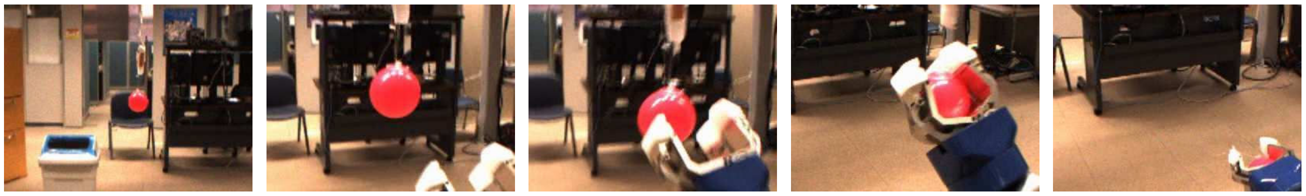


Fig. 5. Experiment B: Joint velocities. To improve the readability, only the joints of the chest, the head and the arm are given.



(iter 2800) (iter 3600) (iter 3700) (iter 3800) (iter 4200)

Fig. 6. Experiment B: Key images of the grasping sequence.



(iter 2800) (iter 3600) (iter 3700) (iter 3800) (iter 4200)

Fig. 7. Experiment B: Key images of the grasping sequence, taken from the embedded (left) camera, and are used during the servo to track the object.

[2] G. Chesi, K. Hashimoto, D. Prattichizzo, and A. Vicino. A switching control law for keeping features in the field of view in eye-in-hand visual servoing. In *IEEE Int. Conf. on Robotics and Automation (ICRA'03)*, pages 3929–3934, Taipei, Taiwan, September 2003.

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

[4] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation*, 8(3):313–326, June 1992.

[5] N. R. Gans and S. A. Hutchinson. An experimental study of hybrid switched approaches to visual servoing. In *IEEE Int. Conf. on Robotics and Automation (ICRA'03)*, pages 3061–3068, Taipei, Taiwan, September 2003.

[6] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Trans. on Robotics and Automation*, 12(5):651–670, October 1996.

[7] 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 *International Conference on Robotics And Automation, Taipei Taiwan*, pages 1620–1626, September 2003.

[8] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Resolved momentum control: Humanoid motion planning based on the linear and angular momentum. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, Las Vegas, Nevada*, pages 1644–1650. IEEE, 2003.

[9] D. Khadraoui, G. Motyl, P. Martinet, J. Gallice, and F. Chaumette. Visual servoing in robotics scheme using a camera/laser-stripesensor. *IEEE Trans. on Robotics and Automation*, 12(5):743–750, October 1996.

[10] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. Journal of Robotics Research*, 5(1):90–98, Spring 1986.

[11] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Motion planning for humanoids robots. In *Proceeding of the 11th International Symposium of Robotics Research, ISRR*, 2003.

[12] A. Liegeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans. on Systems, Man and Cybernetics*, 7(12):868–871, December 1977.

[13] M. Lopes and J. Santos-Victor. Visual learning by imitation with motor representations. *IEEE Transactions on Systems, Man and Cybernetics*, 35(3):438–449, June 2005.

[14] N. Mansard and F. Chaumette. Tasks sequencing for visual servoing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'04)*, pages 992–997, Sendai, Japan, November 2004.

[15] N. Mansard and F. Chaumette. Task sequencing for sensor-based control. *IEEE Trans. on Robotics and Automation*, 2006. To appear.

[16] P. Michel, J. Chestnutt, J. Kuffner, and T. Kanade. Vision-guided humanoid footstep planning for dynamic environments. In *Proc. IEEE/RAS Int. Conf. on Humanoid Robotics (Humanoids'05)*, pages pages 13–18, 2005.

[17] K. Nishiwaki, M. Kuga, S. Kagami, M. Inaba, and H. Inoue. Whole-body cooperative balanced motion generation for reaching. In *2004 4th IEEE/RAS International Conference on Humanoid Robots*, pages 672–689, Los Angeles, USA, November 2004.

[18] L. Peterson, D. Austin, and D. Kragic. High-level control of a mobile manipulator for door opening. In *IEEE Int. Conf. on Robotics and Automation (ICRA'03)*, pages 2333–2338, Taipei, Taiwan, September 2003.

[19] L. Sentis and O. Khatib. A whole-body control framework for humanoids operating in human environments. In *IEEE Int. Conf. on Robotics and Automation (ICRA'06)*, pages 2641–2648, Orlando, USA, May 2006.

[20] N. Sian, K. Yokoi, S. Kajita, F. Kanehiro, and K Tanie. A switching command-based whole-body operation method for humanoid robots. *IEEE/ASME Transactions on Mechatronics*, 10(5):546–559, 2005.

[21] B. Siciliano and J-J. Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *IEEE Int. Conf. on Advanced Robotics (ICAR'91)*, pages 1211–1216, Pisa, Italy, June 1991.

[22] P. Soueres, V. Cadenat, and M. Djeddou. Dynamical sequence of multi-sensor based tasks for mobile robots navigation. In *7th Symp. on Robot Control (SYROCO'03)*, pages 423–428, Wroclaw, Poland, September 2003.

[23] T. Sugihara and Y. Nakamura. Whole-body cooperative balancing of humanoid robot using cog jacobian. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems, EPFL, Lausanne, Switzerland*, pages 2575–2580, October 2002.

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

[25] B. Verrelst, K. Yokoi, O. Stasse, H. Arisumi, and B. Vanderborght. Mobility of humanoids robots: Stepping over large obstacles dynamically. In *International Conference on Mechatronics and Automation, ICMA, LuoYang - Henan, China*, pages 1072–1079, June 2006. Best Conference Paper Award.

[26] K. Yamamura and N. Maru. Positionning control of the leg of the humanoid robot by linear visual servoing. In *IEEE International Conference on Humanoids Robot*, 2004.