Modelling and simulation of the humanoid robot HOAP-3 in the OpenHRP3 platform

C.A. Monje, P. Pierro, T. Ramos, M. González-Fierro, C. Balaguer.

***Abstract*— The aim of this work is to model and simulate the humanoid robot HOAP-3 in the OpenHRP3 platform. Our purpose is to create a virtual model of the robot so that different motions and tasks can be tested in different environments. This will be the first step before testing the motion patterns in the real HOAP-3. We use the OpenHRP3 platform for the creation and validation of the robot model and tasks. The procedure followed to reach this goal is detailed in this paper. In order to validate our experience, different walking motions are tested and the simulation results are compared with the experimental ones.**

I. INTRODUCTION

The existence of computer simulation platforms is fundamental in robotics research, especially if it is with humanoid robots, since it allows us to develop the controllers and the necessary programming without compromising the complex and expensive mechanical system. In general, the objectives that the simulators allow us to approach are:

- To visualize three-dimensional work environment and the model of the robot in motion.
- To provide a test center for the development and evaluation of controls and software of the robot.
- To serve as a graphical user interface, which can even be interactive in real time with the robot.

A necessary requirement for really effective simulations is that the mechanical behavior of the virtual robot answers as closely as possible to the real robot, so the preparation work for a good virtual reality simulation platform turns out to be crucial. Thereby, the programming developed over the simulator will be able to be inherited by real applications.

One of the first principles and overviews of dynamic simulators was given in Baraff and Witkin [1]. The more advanced guidelines and the important problems that should be considered in humanoid dynamic simulations are mentioned in the General Human/Humanoid-Dynamics Simulator proposed in Vukobratovic et al [2]. Although one can discuss the proposed implementations, e.g., the contact model, the paper provides some general guidance for the simulator design and the effects that should be taken into account in the dynamics simulations. Notably, they are a flexibility at the joints when the transmission between the motor and the corresponding joint is not completely rigid but features some elasticity (one additional DOF) and a flexible (deformable) contact between the robot foot and the ground, i.e., elastodynamic contact [3][4]. One of the most widely used simulators, particularly for mobile robots, is the Player/Stage Open Source framework [5]. This consists of a Player robot device server and a 2D Stage multiple robot simulator. The main objective of the framework is research into multi-robot systems, with experiments and control of large population of robots without having to buy real hardware counterparts. The Gazebo [6] platform, an add-on to the Player/Stage framework, is based on OpenGL graphics, specifically on the GLUT toolkit, and Open Dynamics Engine. However, it should generally work in conjunction with Player software running on the robot, and it is mostly applied to mobile robot control. Some simulators, e.g., in Ponticelli and Armada [7], are developed purely in Matlab using Simulink toolboxes, such as SimMechanics, and visualizing systems, such as VRML Viewer toolbox. This approach enables rapid controller design and testing, but it lacks some important features such as surface modeling, and consequently has no collision detection feature.

Existing robotics simulators include, among others, Honda and Sony simulators (proprietary for ASIMO and the QRIO), the Fujitsu HOAP simulator (Fujitsu sells HOAP with a basic simulation software), RoboWorks (a commercial software developed by Newtonium), SD/FAST (by Symbolic Dynamics, which provides nonlinear equations of motion from a description of an articulated system of rigid bodies), and Webots (a commercial software by Cyberbotics). Even Microsoft has developed a product named Microsoft Robotics Studio, which is primarily used for mobile robots.

It is important to mention the OpenHRP platform [8][9] Open Architecture Humanoid Robotics Platform) as a simulator and motion control library for humanoid robots developed at NIAIST (National Institute of Advanced Industrial Science and Technology (Japan)). This is a distributed framework based on CORBA (Common Object Request Broker Architecture), created with the idea of sharing a code between real and virtual robots, and ultimately of developing identical controllers for real and virtual robots. This is a free solution that will be used in this work to model the humanoid robot HOAP-3 and simulate walking patterns before the final test in real time with the real robot.

The work presented here is divided into the following sections. First, Section 2 describes the simulation platform OpenHRP3, which is the version 3 of OpenHRP. Section 3 presents the humanoid robot HOAP-3 model in OpenHRP3. Section 4 details the generation of stable motion patters for HOAP-3 and gives the references for each joint movement. In Section 5, the motion of HOAP-3 in OpenHRP3 is tested for the joint references given previously. After that, the motion of the real robot is measured and compared with that from simulation. Finally, Section 6 presents the main conclusions of this work and future lines to be followed.

II. OpenHRP3 Platform

OpenHRP3 [10] (Open Architecture Humanoid Robotics Platform, version 3) is a simulation platform for humanoid robots and software development. It allows the users to inspect the original model of the robot and the control program across a dynamic simulation. In addition, OpenHRP3 provides several calculation software components and libraries that can be used to develop software related to robotics.

To use OpenHRP3 the following programs, libraries, and programming languages are needed [11]: Microsoft Visual Studio, BOOST, CLAPACK, and TVMET libraries, graphical environment OpenRT, Adaptive Communication Environment (ACE), ORB, Python, Java, and Jython. OpenHRP3 is currently supported on Ubuntu Linux platforms 7 or later and Windows XP and Vista (32bit/64bit). In our case we use Windows XP 64bit.

This virtual humanoid robot platform consists of a simulator of humanoid robots and motion control library for them which can also be applied to a compatible humanoid robot as it is. OpenHRP also has a view simulator of humanoid robots on which humanoid robot vision can be studied. The consistency between the simulator and the robot is enhanced by introducing a new algorithm to simulate repulsive force and torque between contacting objects. OpenHRP is expected to initiate the exploration of humanoid robotics on an open architecture software and hardware, thanks to the unification of the controllers and the examined consistency between the simulator and a real humanoid robot.

The configuration of OpenHRP is shown in Fig. 1. OpenHRP can simulate the dynamics of structure-varying kinematic chains, both open chains and closed ones such as humanoid robots [12]. It can detect the collision between robots and their working environment (including other robots) very fast and precisely, computing the forward dynamics of the objects. It can also simulate the fields of vision of the robots, force/torque sensors, and gradient sensors according to the simulated motions. The sensor simulations are essential to develop the controllers of the robots. OpenHRP is implemented as a distributed object system on CORBA [13]. A user can implement a controller using an arbitrary language on an arbitrary operating system if it has a CORBA binding.

The dynamics simulator of OpenHRP consists of five kinds of CORBA servers (see Fig. 1) and these servers can be distributed on the Internet and executed in parallel. Each server can be replaced with another implementation if it has the same interface defined by IDL (Interface Definition Language). Using the language independence feature of CORBA, ModelParser and OnlineViewer are implemented using Java and Java3D, other servers are implemented using C++. The functions of each server are as follows.

- ModelParser. This server loads a VRML file describing the geometrical models and dynamics parameters of robots and their working environment, and provides these data to other servers.
- CollisionChecker. The interference between two sets of triangles is inspected, and the position, normal vector, and the depth of each intersecting point are found. RAPID (Robotics Application Programming Interactive Dialogue) is enhanced to this end.
- Dynamics. The forward dynamics of the robots are computed.
- Controller. This server is the controller of a robot, which is usually developed by the users of OpenHRP.
- OnlineViewer. The simulation results are visualized by 3D graphics and recorded.

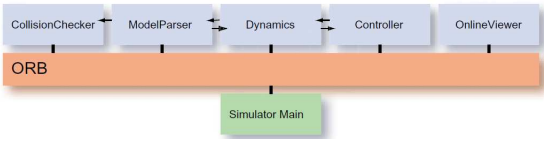

Fig. 1. OpenHRP3 functions.

Using the servers, the forward dynamics of the robots are computed in the following procedure. The total control flow is shown in Fig. 1.

- Setting up of the simulation environment. ModelParser reads a VRML file via HTTP protocol. The kinematics and dynamics parameters are sent to Dynamics and the geometric model is to CollisionChecker.
- Execution of the dynamics simulation. Controller reads the outputs of the simulated sensors while communicating with Dynamics. Controller and Dynamics execute the computations. Note that these computations can be run in parallel. The outputs of Controller are the torques of the actuators, and those of Dynamics are the updated states of the robot. While the forward dynamics is computed, CollisionChecker is called to find the position, normal vector, and the depth of each intersecting point. After these computations, Controller sends the control outputs to Dynamics.
- Visualization and recording. The current states of the world are sent from Dynamics to OnlineViewer, which visualizes the simulated world and records it.

A. *OpenHRP3 simulation interface*

The simulation interface of the OpenHRP3 platform is shown in Fig. 2.

The interface of this simulator is basically divided into three parts. The first part (Fig. 2, Part 1, left) shows the tree structure of the different modules loaded for the simulation. These modules can be the model of the robot, the model of the environment, the collision (Collision Pair) and control modules, and the graphical display package, among others. A command window is available at the right side of Part 1 to introduce command lines directly.

The second part is directed to display the robot model during the simulation (Fig. 2, Part 2). In this area there are two toolbars, the vertical and the horizontal ones, which allow us to modify aspects of the display of the robot environment, take pictures, and record videos of tasks.

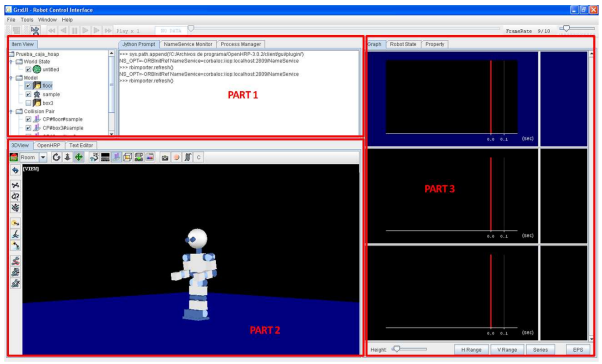

Fig. 2. OpenHRP3 interface.

Fig. 3. Humanoid robot HOAP-3.

The third part is placed at the right of the interface (Fig. 2, Part 3). It displays the signals that represent the evolution of the joint variables during the simulation (position, velocity, acceleration) as well as the information of the different sensors included along the mechanical structure of the robot (force/torque sensors, etc).

III. HOAP-3 Model in OpenHRP3

The miniature humanoid robot HOAP-3 [14] (Humanoid for Open Architecture Platform) has been developed by Fujitsu in collaboration with Fujitsu Automation Lab (Fig. 3).

The HOAP-3 robot has a height of 60 cm and an approximate weight of 8 Kg. It has 28 degrees of freedom (DOF) distributed as shown in Fig. 4.

For modeling in OpenHRP the VRML [15] file describing the geometrical model and dynamics parameters of the robot must be created. The VRML structure that relates the different DOF of the robot is shown in Fig. 5. The main joint of the model is the WAIST, and the upper (arms) and lower (legs) bodies are defined from it, in this order.

The Humanoid node is the root node of the model. Along with it, Joint and Segment nodes (joint and link, respectively) are basic elements of VRML and define the type of each joint (Joint) and its corresponding shape (Segment).

Once the VRML model of the robot is created according to the geometrical and dynamic parameters provided by Fujitsu, the file can be loaded as a module in the tree structure of the simulator interface, as defined previously.

In order to reduce the computation cost and speed up the test procedure, we have modeled a more basic shape of the robot instead of modeling its real appearance, which would imply a more complex VRML file to be computed.

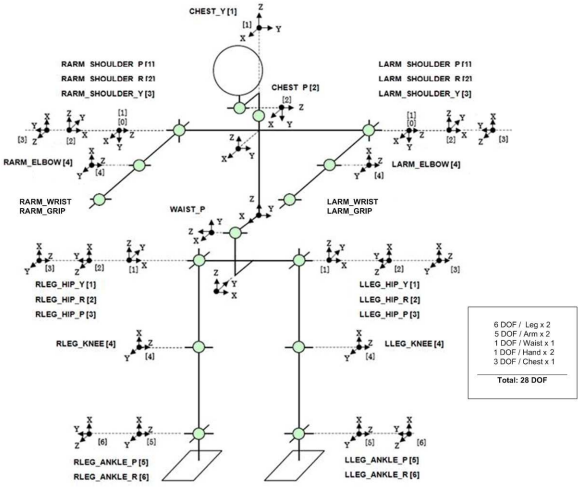

Fig. 4. Distribution of DOF in HOAP-3.

IV. Generation of Stable Motion Patterns for HOAP-3

In order to generate stable walking patterns of the humanoid, we have used the cart-table model [16]. This model (Fig. 6) is based on the ZMP (Zero Moment Point) preview control scheme that obtain the COG (Center of Gravity) trajectory from a defined ZMP trajectory. The relationship between the ZMP trajectory and the COG trajectory is defined by the following equations:

$$p_x = x - \frac{\ddot{x}}{g} z_c, \tag{1}$$

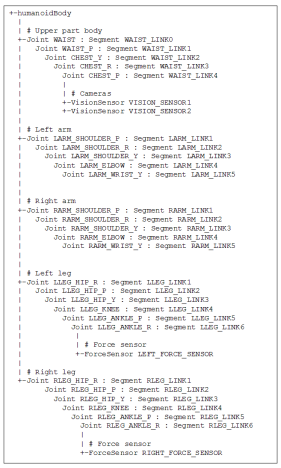

Fig. 5. Joint structure in VRML model.

$$p_y = y - \frac{\ddot{y}}{g} z_c, \tag{2}$$

where, in the sagittal plane, p_x is the ZMP reference, x is the COG trajectory, $\ddot{x}$ is the COG acceleration, z_c is the COG height, and g is the gravity. We are going to focus on the calculations in the sagittal plane. For the frontal plane the procedure is the same but using the y component of these terms.

In the cart table model, the cart mass corresponds to the center of mass of the robot. If the cart accelerates at a proper rate, the table can be upright for a while. At this moment, the moment around p_x is equal to zero, so the ZMP exists:

$$\tau_{ZMP} = mg(x - p_x) - m\ddot{x} z_c. \tag{3}$$

In order to obtain the COG trajectory, we can define the ZMP control as a servo problem. Using the optimal preview servo controller technique [17], the COG trajectory can be obtained from a ZMP reference. We define the time derivative of acceleration of the COG as

$$u_x = \frac{d}{dt}\dddot{x}. \tag{4}$$

Using u_x as the input of (1), we can put the ZMP equations in the form of a variable state problem:

$$\frac{d}{dt}\begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} u_x, \tag{5}$$

$$p_x = \begin{pmatrix} 1 & 0 & z_c/g \end{pmatrix}\begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix}. \tag{6}$$

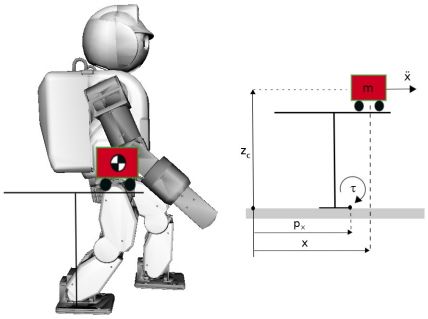

Fig. 6. Cart table model in sagittal plane.

The trajectories of the COG are discretized as piecewise cubic polynomials at intervals of constant time T, using the notation:

$$\hat{x}_k = \begin{pmatrix} x(kT) \\ \dot{x}(kT) \\ \ddot{x}(kT) \end{pmatrix}, u_k = u_x(kT), p_k = p_x(kT). \tag{7}$$

Equation (5) and (6) can be transformed into

$$\hat{x}_{k+1} = \begin{pmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{pmatrix}\hat{x}_k + \begin{pmatrix} T^3/6 \\ T^2/2 \\ \ddot{x} \end{pmatrix} u_k, \tag{8}$$

$$p_k = \begin{pmatrix} 1 & 0 & z_c/g \end{pmatrix}\hat{x}_k. \tag{9}$$

The constraints of the COG are defined by:

$$p_k^{min} \leq p_k \leq p_k^{max}, \tag{10}$$

where the maximal and minimal value are defined by the edge of the feet.

To design the optimal servo controller, the performance index can be expressed as

$$J = \sum_{i=k}^{\infty} \left\{ Q_e e(i)^2 + \Delta x^T(i) Q_x \Delta x(i) + R\Delta u^2 \right\}, \tag{11}$$

where $e(i) = p(i) - p^{ref}(i)$ is the servo error, Q_e, $R > 0$, Q_x are symmetric non-negative definite matrices, $\Delta x = x(k) - x(k-1)$ is the incremental state vector, and $\Delta u = u(k) - u(k-1)$ is the incremental input.

The optimal controller that minimizes the index in (11) is given by

$$u(k) = -G_i \sum_{i=0}^{k} e(k) - G_x \hat{x}(k) - \sum_{j=1}^{N_L} G_p(j) p^{ref}(k+j), \tag{12}$$

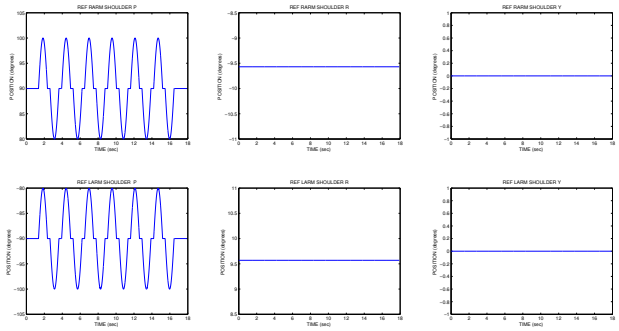

Fig. 7. Trajectories for the 3 DOF of the right and left shoulders, respectively.

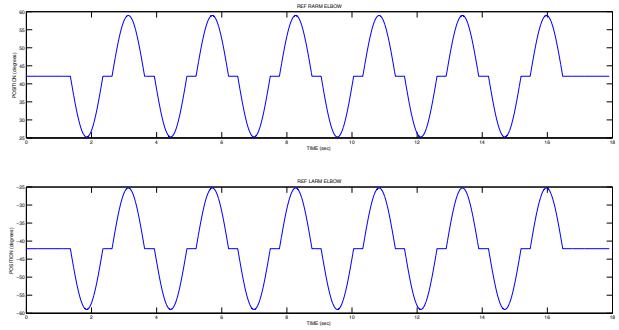

Fig. 8. Trajectories for the 1 DOF of the right and left elbows, respectively.

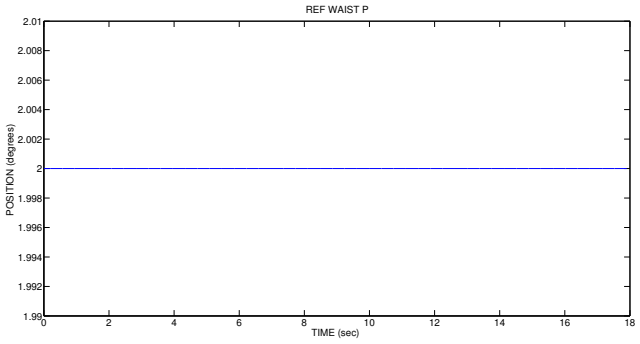

Fig. 9. Trajectories for the 1 DOF of the waist.

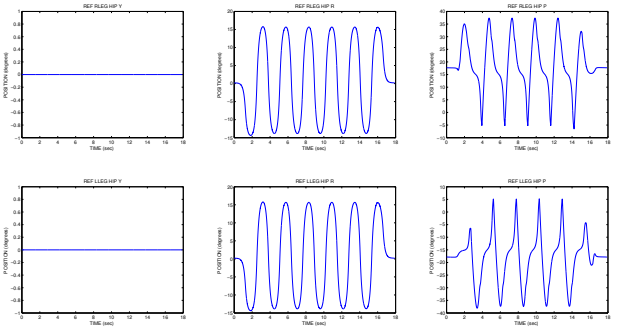

Fig. 10. Trajectories for the 3 DOF of the right and left hips, respectively.

where G_i, G_x and G_p are the controller gains and N_L is the preview step.

Once the COG trajectory is obtained, the trajectories of the lower body joints are calculated applying inverse kinematics. In our case, we have select a step distance of 8 cm, a COG height of 32 cm, and a preview time of 0.75 sec. The robot walks 12 stpdf forward. The resulting joints trajectories are shown from Fig. 7 to Fig. 12, including joints references for the shoulders and elbows of both arms to perform a more natural walking.

V. Simulation and Experimental Results

Our purpose is to test the walking patterns obtained in Section 4 in the real HOAP-3. Prior to this experimental test, the OpenHRP3 simulation platform is used to check the stability of the robot during this walking action in simulation. The specifications of the computer used include CPU:Intel Core Duo 2.4GHz, RAM Memory: 2GB, and OS:Windows XP 64 bits. The simulation time in this case is 18 seconds.

The VRML file of HOAP-3 is loaded in the modules tree of the simulator together with the CollisionChecker and Controller modules defined by OpenHRP3. After the simulation test, the stable sequence of motions shown in Fig. 13 is obtained.

Once the stability of the robot is guaranteed in simulation, the joints trajectories are loaded in the real HOAP-3 platform and the walking test in performed experimentally. Very good results are obtained, as can bee seen from Fig. 14 to Fig. 19 comparing the experimental and simulation joint angles measured by the robot encoders and simulator angular position sensors, respectively.

According to the good results obtained, a more complex and complete sequence of motions has been tested in HOAP-3 following the same procedure explained here, that is: first, we generate the joints references from the application of the cart table concept to ensure the stability of the patterns; then, we test the sequence in the OpenHRP3 simulation platform; finally, after the validation of the stability in simulation, the sequence of motions is tested in the real HOAP-3 platform. Following

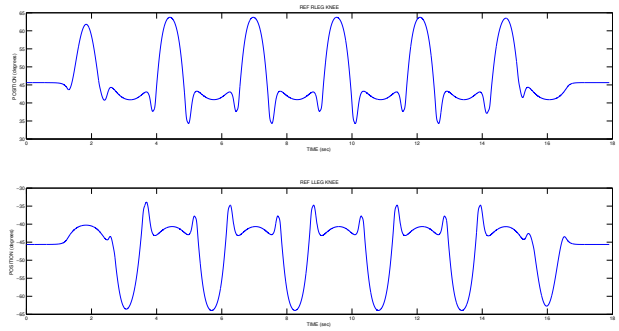

Fig. 11. Trajectories for the 1 DOF of the right and left knees, respectively.

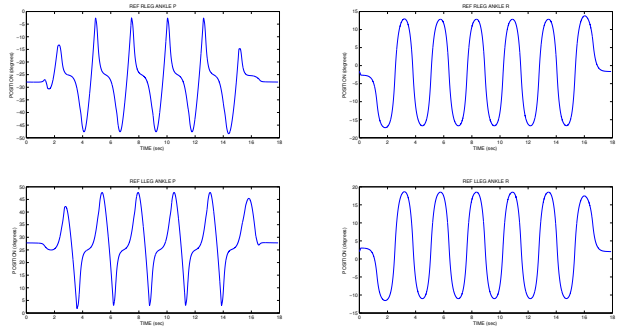

Fig. 12. Trajectories for the 2 DOF of the right and left ankles, respectively

this procedure, several dance steps have been performed by HOAP-3, as can be seen in the following video showing the robot dancing in real time in an exhibition during 3 minutes: http://www.youtube.com/watch?v=mu5psxG7bwA Some pictures taken from this performance are shown in Fig. 20. As can be seen, the simulation in OpenHRP3 is run and shown in a top window during the performance. The simulation also includes the same environment as the real one (forest area).

Another important approach we are currently working on is the manipulation of objects using HOAP-3. For that purpose, it is necessary to measure the torque applied to the robot wrists. The HOAP-3 platform has two force sensors on board, one in each wrist. Our objective now is to introduce these torque measurements in simulation and create an environment where the robot can operate freely, interacting with objects in a stable way. We have already taken some steps towards this goal, as can be seen in the sequence shown in Fig. 21. In this case, the robot raises his arms until the hands touch the wall. The torques in each wrist are measured

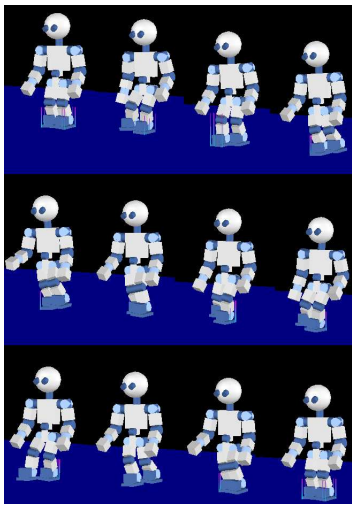

Fig. 13. Sequence of motions for HOAP-3 during a step.

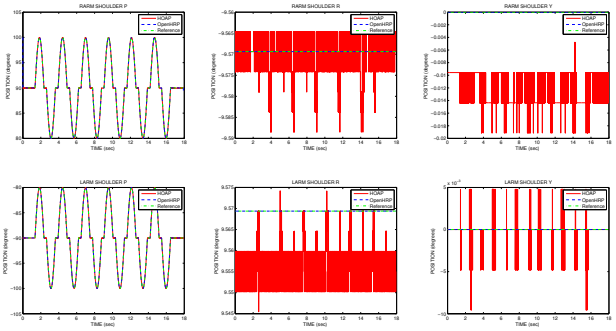

Fig. 14. Trajectories for the 3 DOF of the right and left shoulders, both in OpenHRP3 and HOAP-3.

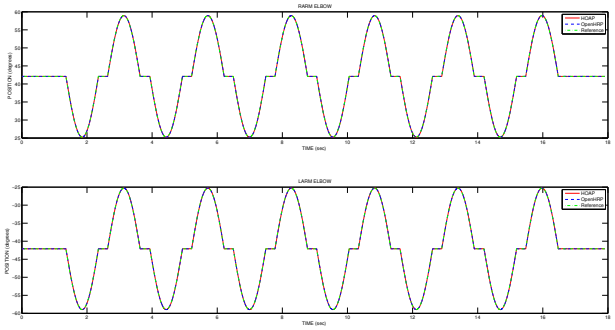

Fig. 15. Trajectories for the 1 DOF of the right and left elbows, both in OpenHRP3 and HOAP-3.

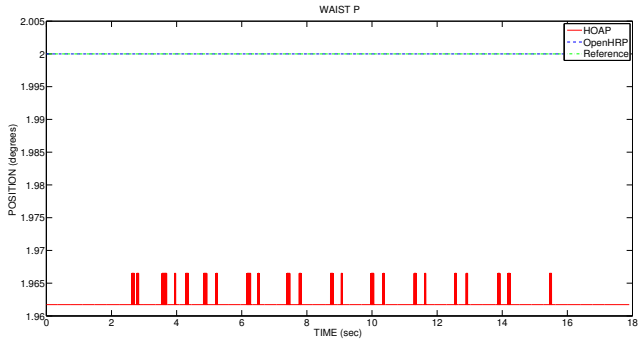

Fig. 16. Trajectories for the 1 DOF of the waist, both in OpenHRP3 and HOAP-3.

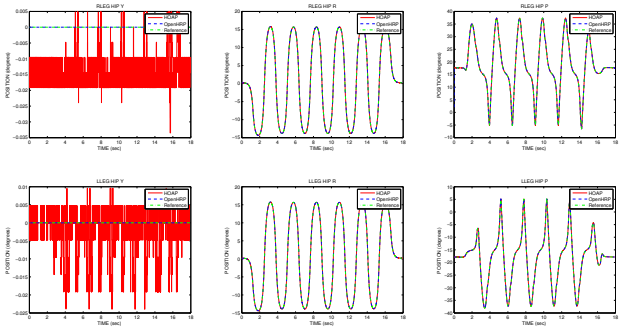

Fig. 17. Trajectories for the 3 DOF of the right and left hips, both in OpenHRP3 and HOAP-3.

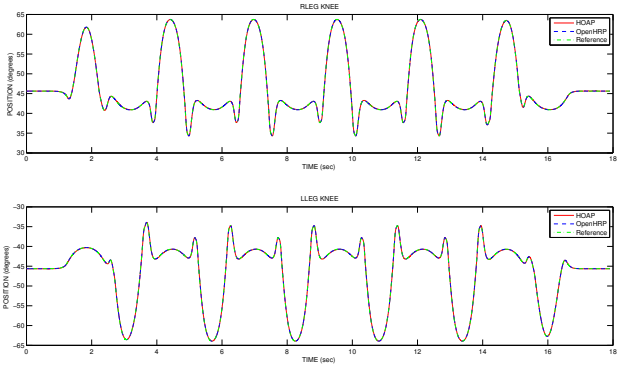

Fig. 18. Trajectories for the 1 DOF of the right and left knees, both in OpenHRP3 and HOAP-3.

by two torque sensors included in the model of HOAP-3 in OpenHRP3, one in each wrist. The torque measurements for the right and left wrist are the ones show in Fig. 22.

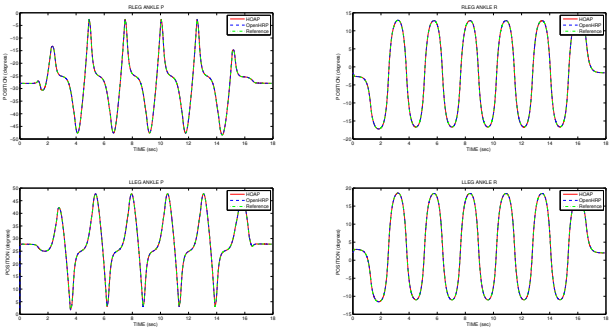

Fig. 19. Trajectories for the 2 DOF of the right and left ankles, both in OpenHRP3 and HOAP-3.

Fig. 20. HOAP-3 dancing in an exhibition during 3 minutes. The simulation is run and shown in a top window during the performance.

For real tests, we have built a moon environment for HOAP-3 to interact with objects, as can be seen in Fig. 23. We are now working on the creation of this environment in OpenHRP3 and using the sensorial integration of this platform to program manipulation activities.

VI. Conclusions and Future Works

In this paper, we have used the potential of the OpenHRP3 platform to create a virtual model of the humanoid robot HOAP-3 and simulate different motions. The cart table model has been used to obtain stable motion patterns for the robot. These patterns have been tested in simulation and the stability of HOAP-3 during the walking action has been validated. After this validation, experimental tests have been performed with the real robot with very good results, comparing simulation and experimental data and showing the efficient performance of the robot model in OpenHRP3.

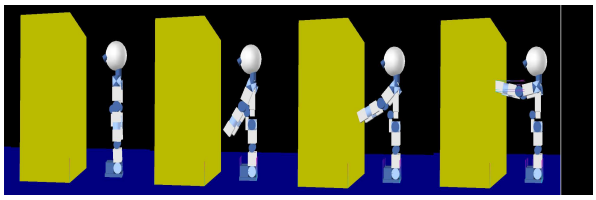

Fig. 21. Sequence of motions for HOAP-3 touching a wall.

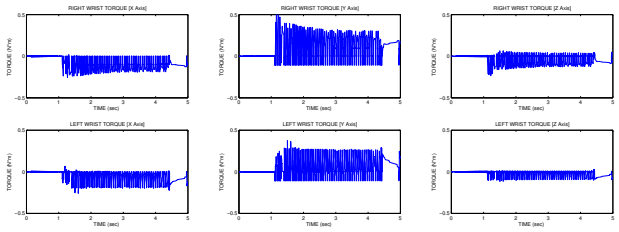

Fig. 22. Torques of right (up) and left (down) wrists during the touching motion.

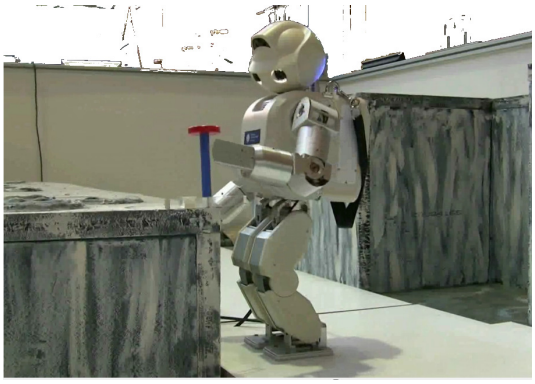

Fig. 23. Real moon environment for HOAP-3 for interaction and manipulation of objects.

Finally, a more complex and complete sequence of motions has been successfully tested again in simulation and experimentally, in which the robot dance during 3 minutes ensuring the stability during the whole task.

New steps are also being taken towards the simulation of manipulation tasks with HOAP-3. For this purpose, a wall has been integrated in the simulation environment and the torques in the robot wrists when they get in contact with the wall are measured by torque sensors included in the model of the robot in OpenHRP3.

Future works will be in the line of creating a virtual environment where the robot can interact with other robots or objects thanks to the sensorial integration available in OpenHRP3.

VII. ACKNOWLEDGEMENTS

This work has been supported by the CYCIT Project PI2004-00325 and the European Project Robot@CWE FP6-2005-IST-5, both developed by the research team RoboticsLab at the University Carlos III of Madrid.

REFERENCES

[1] A. Witkin, "Physically based modeling: Principles and practice constrained dynamics," in *COMPUTER GRAPHICS*. Citeseer, 1997.

[2] M. Vukobratovic, V. Potkonjak, and S. Tzafestas, "Human and humanoid dynamics," *Journal of Intelligent and Robotic Systems*, vol. 41, no. 1, pp. 65–84, 2004.

[3] Y. Fujimoto, S. Obata, and A. Kawamura, "Robust biped walking with active interaction control between foot and ground," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 3. IEEE, 1998, pp. 2030–2035.

[4] T. Sugihara and Y. Nakamura, "Contact phase invariant control for humanoid robot based on variable impedant inverted pendulum model," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 1. IEEE, 2003, pp. 51–56.

[5] B. Gerkey, R. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proceedings of the 11th international conference on advanced robotics*. Citeseer, 2003, pp. 317–323.

[6] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, pp. 2149–2154, September 2004.

[7] R. Ponticelli and M. Armada, "Vrsilo2: dynamic simulation system for the biped robot silo2," *9th International Conference on Climbing and Walking Robots and the Supporting Technologies for Mobile Machines (CLAWAR)*, 2006.

[8] F. Kanehiro, N. Miyata, and S. Kajita, "Virtual humanoid robot platform to develop con- trollers of real humanoid robots without porting," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1093–1099, 2001.

[9] F. Kanehiro, H. Hirukawa, and S. Kajita, "Openhrp: Open architecture humanoid robotics platform." *I. J. Robotic Res.*, vol. 23, no. 2, pp. 155–165, 2004. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijrr/ijrr23.html

[10] "Openhrp3 official site." [Online]. Available: http://www.openrtp.jp/openhrp3/en/index.html

[11] K. Yamane and Y. Nakamura, "Dynamics computation of structure-varying kinematic chains for motion synthesis of humanoid." in *ICRA*, 1999, pp. 714–721. [Online]. Available: http://dblp.uni-trier.de/db/conf/icra/icra1999-1.html

[12] "Object management group." [Online]. Available: http://www.omg.org/

[13] S. Gottschalk, M. Lin, and D. Manocha, "Obb-tree: A hierarchical structure for rapid interference detection," *23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 171–180, 1996.

[14] FUJITSU, *HOAP-3 Instruction Manual.*

[15] "The virtual reality model language." [Online]. Available: http://www.web3d.org/x3d/specifications/vrml/VRML1.0/index.html

[16] S. Kajita, F. Kanehiro, K. Keneko, K. Fujiwara, K. Harada, and K. Yokoi, "Biped walking pattern generation by using preview control of zero-moment point," *IEEE International Conference on Robotics & Automation (ICRA)*, pp. 1620–1626, 2003.

[17] T. Katayama, T. Ohki, and T. Kato, "Design of an optimal controller for a discrete time system subject to previewable demand," *International Journal of Control*, vol. 41, no. 3, pp. 677–699, 1985.