HUMANOID ROBOT SIMULATOR: A REALISTIC DYNAMICS APPROACH

José L. Lima, José C. Gonçalves, Paulo G. Costa, A. Paulo Moreira

Department of Electrical Engineering
Faculty of Engineering of University of Porto
jllima@ipb.pt, goncalves@ipb.pt, paco@fe.up.pt, amoreira@fe.up.pt

Abstract: This paper describes a humanoid robot simulator with realistic dynamics. As simulation is a powerful tool for speeding up the control software development, the suggested accurate simulator allows to accomplish this goal. The simulator, based on the Open Dynamics Engine and GLScene graphics library, provides instant visual feedback and allows the user to test any control strategy without damaging the real robot in the early stages of the development. The proposed simulator also captures some characteristics of the environment that are important and allows to test controllers without access to the real hardware. Experimental results are shown that validate this approach.

Keywords: Computer simulation, Digital control, Computer graphics, Dynamic behaviour, Kinematic control system.

1. INTRODUCTION

In recent years, studies of research in biped robots have been developed rapidly and resulted in a variety of prototypes that resemble the biological systems. Legged robots have the ability to choose optional landing points, an advantage to move in rugged terrains. Especially, two legged robots are also able to move in human environment since its structure is almost same with humans. Thus, studies about biped robots are very important (Suzuki, et al., 2006).
Furthermore, bipedal locomotion under influence of external disturbances is a challenging task for a humanoid robot. If disturbances are large enough, a fall might become unavoidable. Postural reflexes should minimize the number of falls (Renner, et al., 2006). If a fall happens, the robot must be able to detect it, to recognize its posture on the ground and to get back into an upright posture (Stückler, et al., 2006).
The simulator must also be able to measure the consumed energy providing a good efficiency planning. The planning for humanoid movements should result in minimum energy consumption, like it happens in the human body.

A screenshot of the developed simulator is shown in Fig. 1, where 3D scene shows the robot human-like, graphic shows the desired time variables and table shows the angle, angular speed and torque for each robot joint. There are several robot simulators, such as Simspark, Webots, MURoSimF and ADAMS, that provide a simulation capability. Meanwhile, the developed simulator allows to build and to test the low and high level controllers in a way that can be mapped with the reality, although with a minimal overhead (Browning, et al., 2003). Code migration from general realistic simulators to real world systems is the key for reducing development time of robot control, localization and navigation software.
The motivation of developing a realistic humanoid robot simulator is to produce a personalized and versatile tool that will allow in the future the production and validation of robot software reducing considerably the development time. This simulator deals with robot dynamics and how it reacts for several controller strategies and styles. This paper proposes a simulator for a humanoid robot and compares it to the real robot. The proposed simulator allows to design behaviours without access to the real hardware in order to carry out research on robot

control. This simulator is based on the Open Dynamics Engine (Smith, 2000). It is very helpful to design and develop movements which can be applied to the real robot.

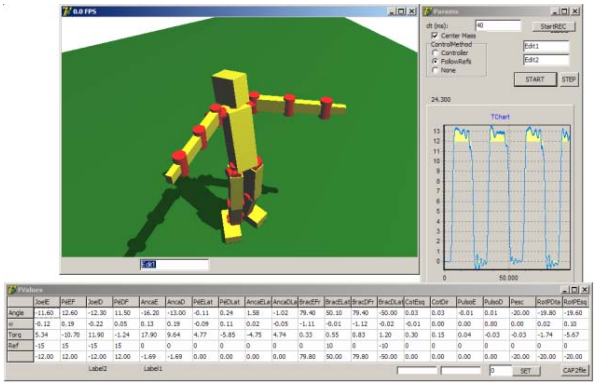


Fig. 1 Simulator screenshot.

The paper is organized as follows: Initially, the real robot, where mechanical design, communication and control application are described, is presented. Then, section 3 presents the developed simulator and how it can be programmed by user. A comparison between the real and the simulated robot is presented further in section 4. Finally, section 5 rounds up with conclusions and future work.

2. REAL HUMANOID ROBOT

The commercially available Bioloid (Tribotix, 2004) robot kit, from Robotis, is the basis of the presented humanoid robot. The overview of the proposed biped robot is shown in Fig. 2. It follows the dimensional rules of RoboCup (2007) and Humanoid League (2007).

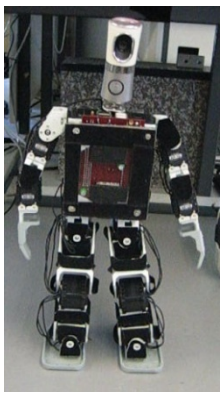


Fig. 2 Real humanoid robot standing.

2.1 Mechanical design

The presented humanoid robot is driven by 19 servo motors: 6 per leg, 3 in each arm and one in the head. Three orthogonal servos set up the 3DOF (degree of freedom) hip joint. Two orthogonal servos form the 2DOF ankle joint. One servo drives the head (a vision camera holder). The shoulder is based on two orthogonal servos allowing a 2DOF joint and elbow has one servo allowing 1DOF. The total weight of the robot (without camera and onboard computer) is about 2kg and its height is 38 cm. The 3D modelled system composed by 19 DOF is presented in Fig. 3 where left picture shows the front view and the right one shows the back view. Each link has its weight at the position of its COG (centre of gravity).

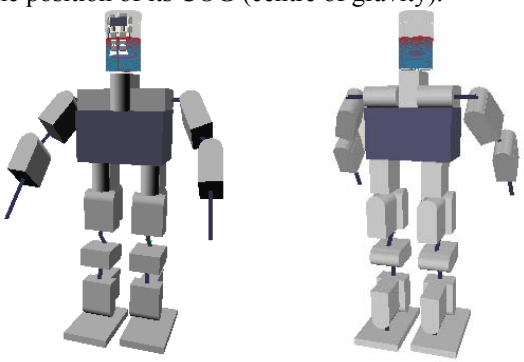


Fig. 3 Modelled robot.

This model, based on GLScene (GLScene, 2000) is used just for visualization of the real robot and has nothing to do with the dynamics simulation. GLScene introduction is further presented in Section 3.

2.2 Communication architecture

Multiple layers that run on different time scales contain behaviours of different complexity. The lowest level of this hierarchy, the control loop within the Dynamixel actuators (AX-12), has been implemented by Robotis (Behnke, et al., 2006). Each servo is able to be programmed with not only the goal position, the moving speed, the maximum torque, the temperature and voltage limits but also with the control parameters. This communication layer is based on a 1Mbps half-duplex serial bus where individual servo can be addressed or a broadcast can be sent.

At the next layer, an interface unity CM-5 module, based on an Atmel ATMega128 microcontroller, allows a communication interchange. It receives messages from the upper layer and translates them to the servos bus. Answers from servos are also translated and sent back to the upper layer as presented in Fig. 4.

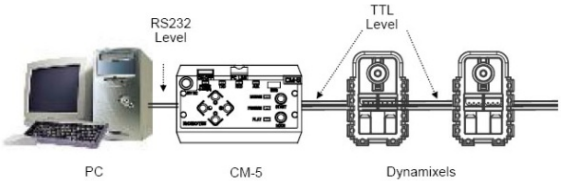


Fig. 4 Interface layer.

At the next layer, target angles for the individual joints are generated from a personal computer or from an embedded system. The layers diagram is presented in Fig. 5.

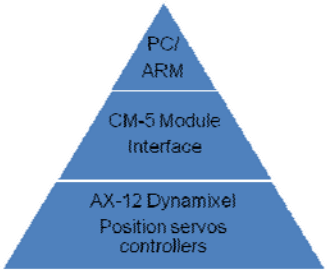


Fig. 5 Layers diagram.

Each broadcast communication takes about 12 ms. Therefore, the fastest allowed control rate and sampling servo motors states is about 83 Hz.

2.3 Behaviour and control

Perception assumes a major role in an autonomous robot, and must be therefore reliable or abundant (Santos, et al., 2006). For this robot, the following perception was planned:

1. Joint position.
2. Joint speed.
3. Joint motor torque.

As a future feature, the following perception support was also planned:

1. Body orientation based on accelerometers.
2. Feet force sensing (Kagami, et al., 2004).

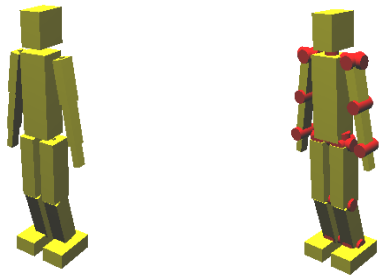
As a first approach, an open-loop system can be used (accelerometers and feet force information disabled). This can be done sending pre-programmed joint angles and angular speeds for each joint. Walk and stand up movements can be achieved. The closed loop control can be done resorting to COG estimation.

3. OPEN DYNAMICS ENGINE SIMULATION

Design behaviour without real hardware is possible due to a physics-based simulator implementation. The physics engine is the key to make simulation useful in terms of high performance robot control. Although there are a number of open source simulation engines available, most focus on producing fast pseudo realistic simulations for use in computer games. These engines are therefore fast, but produce motions that look good as opposed to being accurate. In contrast, there exist a number of simulation engines for rigid body motion that are unusable for simulating the mechanical interactions of rigid parts (Browning, et al., 2003). For real-time simulation, an accurate but fast simulation engine must be used. ODE, Open Dynamics Engine (Smith, 2000), checks these requisites. As an open source rigid body simulation engine, developed by Russell Smith, has reached a maturity level ensuring that produced code is stable. It is essentially a simulation library that provides support for rigid body motion, rotational inertia and collisions treatment where the world to be simulated is built. It also allows to use open GL (graphics library) routines to render the 3D simulated environment. The open GL routines are based on GLScene library. It provides visual components and objects allowing description and rendering of 3D scenes in an easy, no-hassle, yet powerful manner. It has grown to become a set of founding classes for a generic 3D engine with RAD (Rapid Application Development) in mind (GLScene, 2000).

3.1 Humanoid Construction

A complex humanoid model can be avoided due to the ODE usage. Humanoid body simulator construction is based in body masses and joint connections. Each body mass imitates the servo motors and connection pieces weights from the real robot as presented in Fig. 6a). ODE joints, presented in Fig. 6b) by cylinders, imitate the servo motors axis movements and must be defined its types, angles and torques limits. Joint types are typically a hinge that allows both bodies to be connected and roll such as arms and forearms, femur and leg. A more complex joint must be introduced when there are two or more degrees of freedom between two bodies. It happens when two servo motors are physically combined. A universal joint solves the problem allowing a two bodies connection to roll on two axes. As example, presented in the simulator, these joints connect trunk and arms, trunk and legs, legs and feet. This simulator has one more degree of freedom for each arm than the real robot: its wrist. User can deactivate this joint and it behaves like forearm prolongation.



a) b)

Fig. 6 ODE and GLScene humanoid construction.

GLScene is used to render the 3D graphics appearance enhancing visualization.

3.2 Humanoid low-level controller

This controller accepts, for each servo, angles and angular speeds from a higher level, with a desired period *T* (example: 1 second) that can be defined by user. The main objective of this controller is to build and to follow the trajectories established by angles and angular speeds requirements. The low-level controller finds the intermediate trajectories that take joints to the desired states and follow them. Let suppose that for $t=t_1$ (actual time) it is measured angle θ_1 and angular speed ω_1, and for $t=t_2$ (next period *T*) it is desired position θ_2 and angular speed ω_2, as illustrated in Fig. 7 and Fig. 8, where some examples of possible trajectories are shown. It is necessary to calculate the angle equation that result in the desired conditions.

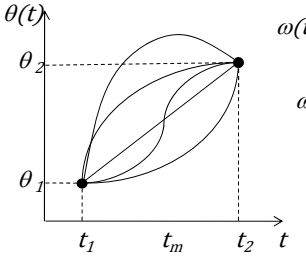


Fig. 7 Joint angles (actual θ_1 and desired θ_2).

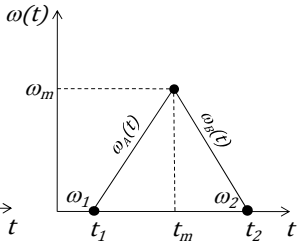


Fig. 8 Joint speed (actual ω_1 and desired ω_2).

Assuming a constant angular acceleration, angular speed will follow a linear equation and the ω_m (for t_m instant) must be determined. The t_m instant is the middle of t_1 - t_2 period, t_m=½(t_1+t_2), as illustrated in Fig. 8 as a first approach. As future work, t_m can be chosen having in mind maximum acceleration minimization. By this way, angular reference and angular speed equations can be found as a smooth movement, following the desired conditions.

The angular speed equation $\omega_A(t)$ for $t_1<t\leq t_m$ is presented in (1) and the angular speed equation $\omega_B(t)$ for $t_m<t\leq t_2$ is presented in (2).

$$\omega_A(t) = \frac{\omega_m-\omega_1}{t_m-t_1}\cdot t + \omega_1 - \frac{\omega_m-\omega_1}{t_m-t_1}\cdot t_1 \quad (1)$$

$$\omega_B(t) = \frac{\omega_2-\omega_m}{t_2-t_m}\cdot t + \omega_2 - \frac{\omega_2-\omega_m}{t_2-t_m}\cdot t_2 \quad (2)$$

The covered angle can be determined through the integral of the angular speeds as presented in equation (3).

$$\theta_2 - \theta_1 = \int_{t_1}^{t_m} \omega_A(t)dt + \int_{t_m}^{t_2} \omega_B(t)dt \quad (3)$$

Equation (3) gives the desired value for ω_m presented in equation (4).

$$\omega_m = \frac{(\omega_1+\omega_2)\cdot(t_1-t_2)+4\cdot(\theta_2-\theta_1)}{2\cdot(t_2-t_1)} \quad (4)$$

Then, angle reference equation, for each *i* joint, can be described in equation (5) for $t_1<t\leq t_m$ and in equation (6) for $t_m<t\leq t_2$.

$$\theta_i^{ref}(t) = \theta_1 + \omega_1\cdot t + \frac{1}{2}\cdot\frac{\omega_m-\omega_1}{t_m-t_1}\cdot t^2 \quad (5)$$

$$\theta_i^{ref}(t) = \theta_{(t=t_m)}^{ref} + \omega_2\cdot t + \frac{1}{2}\cdot\frac{\omega_2-\omega_m}{t_2-t_m}\cdot t^2 \quad (6)$$

The presented equations (1 to 6) define the *T* period references generator.

The same equations can be applied in order to get a closed loop system with a smaller period, *T'* of 40 ms.

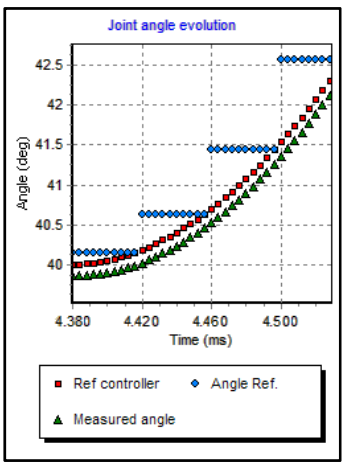


Fig. 9 Detailed joint angle low-level controller.

The initial and the final state are the calculated references, θ_i^{ref}. Having the desired equation of angle and angular speed for each joint, a proportional controller can be applied to follow $\theta_i(t)$ and $\omega_i(t)$. In the simulator, the low-level controller output is the torque (T_i) to be applied on each *i* joint and can be found by equation (7), where $\theta_i^{ref}(t)$ is calculated by equations (5) and (6) and $\omega_i^{ref}(t)$ is calculated by equations (1) and (2). Gains constants K_i^θ and K_i^ω depend on each joint due to its submitted effort.

$$T_i(t) = K_i^\theta\cdot\left(\theta_i^{ref}(t) - \theta_i(t)\right) + K_i^\omega\cdot\left(\omega_i^{ref}(t) - \omega_i(t)\right) \quad (7)$$

A high detailed graph (with a small angle scale), presented in Fig. 9, shows how the low-level controller follows the $\theta_i^{ref}(t)$ (Ref controller) guiding $\theta_i(t)$ (Measured angle) to the requested angle (Angle ref.), based on equation (7).

The simulator closed loop control frequency is the same of the real robot, but higher frequencies can be tested once there is no RS-232 communication limits. The simulator step frequency, f_{sim}, is 4 kHz, the ODE calculus frequency of physics movements. Closed loop control frequency is lower than f_{sim} and synchronous with the 3D visualization updating based on GLScene.

As robotics soccer is a challenge in a highly dynamic environment, the robot controller must be updated as fast as possible. As an example, if the ball has a speed of 2 m/s and if the lag time is 100 ms, the ball will travel a distance of 20 cm between two sampling instants, compromising the controller performance (Gonçalves, et al., 2007).

As a final result, presented in Fig. 10, the left arm joint angle $\theta_i(t)$ (Measured angle) follows the $\theta_i^{ref}(t)$ (Ref controller).

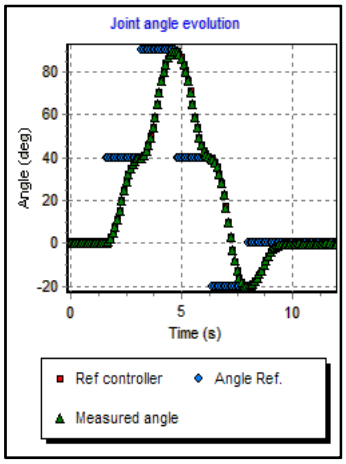


Fig. 10 Joint angle controller.

The *Ref controller curve* is overlapped by the *measured angle* due to its proximity. Presented *Angle Ref* to be followed is actualized every second.

The low-level controller if fully implemented in the developed simulator, leaving a high level controller freedom to calculate trajectories, joint angles, angular speeds and finally perception listening.

3.3 High-level controller

A higher level controller generates the desired joint states, similar to the real robot control loop, that establish the robot simulator movements based in its current position (and equilibrium when in closed loop method). As first case, open loop, joint angles and angular speeds should be sent to the robot. These joint sequences can be saved in a file and shared with the real robot. Walk and stand up routines can be achieved. Furthermore, there are several related works in literature on methods for walk pattern planning. It can be applied on a slippery surface (Park, et al., 2001), with two kinds of inverted pendulums (Park, 1998), using Gravity-compensated inverted pendulum mode (Suzuki, et al., 2006) or Zero Moment Point (ZMP) pattern generation (Kajita, et al., 2006). Perturbation analysis should also be implemented such as joint measures corrupted by noise or collisions applied to the robot simulating a real crash between humanoid and an object.
To maintain dynamic equilibrium during walk and stand up movements, robot needs information about contact force, its current and desired motion. The solution to this problem relies on a major concept, the ZMP as presented in next subsection. The COG can be determined and hip angles controller allows to guarantee the desired stability.

3.4 Zero Moment Point visualization

The Zero Moment Point (ZMP) specifies the point with respect to which dynamic reaction force at the contact of the foot with the ground does not produce any moment, i.e. the point where total inertia force equals zero. ZMP is important in order to guarantee and measure the robot equilibrium.

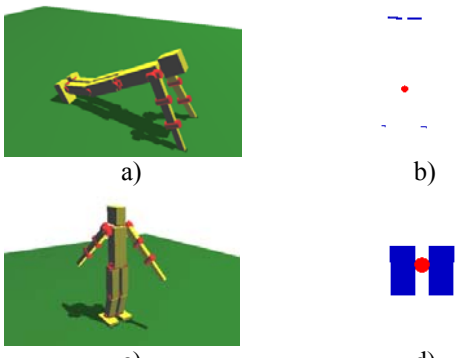


Fig. 11 ZMP drawing.

The robot equilibrium is measured by the distance between the centre of mass (CM) and the ZMP convex hull. The simulator draws, in real-time, the ZMP and the centre of mass. As examples presented in Fig. 11, two body positions, push-up a) and standing c) result in the ZMP drawn at the right side, b) and d). ZMP can also be used to generate movements patterns ZMP that allows a smooth and soft motion (Kajita, et al., 2006).

4. SIMULATION AND REAL ROBOT BEHAVIOUR RESULTS

This chapter presents, in a short way, a comparison between simulator and real robot behaviour. Stand-up movements were successfully tested and shown in next subsection.

4.1 Getting back on two feet - movements

As robot posture depends on external disturbances and on its equilibrium, a fall might occur. If it happens, the robot must be able to recognize its posture on the ground, usually supine or prone. A stand up routine must be initialized in order to place the robot standing (Stückler, et al., 2006).
The simulator and the real robot stand up movements comparison is made in Fig. 12 and Fig. 13.

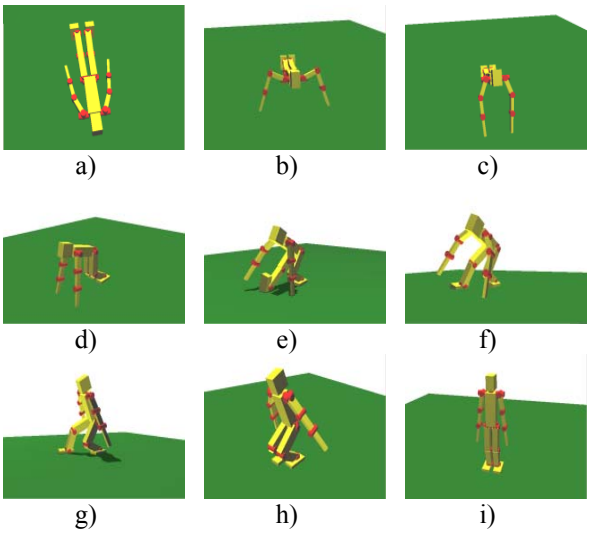


Fig. 12 Simulator stand-up movements.

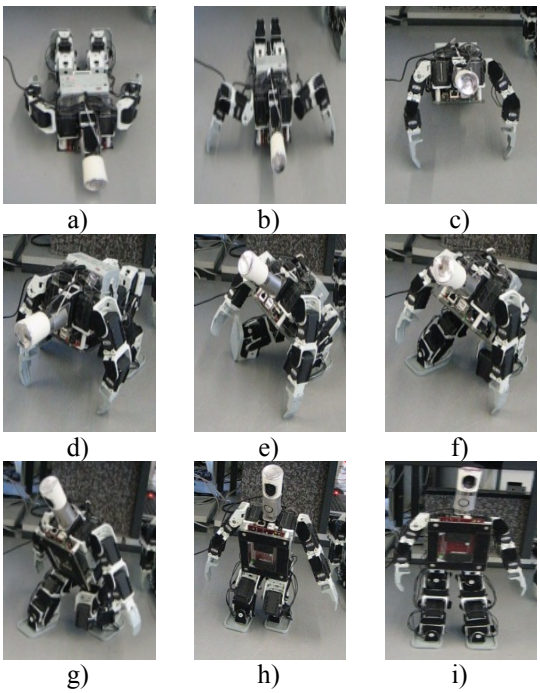


Fig. 13 Real robot stand-up movements.

4.2 Getting back on two feet – current consumption

The humanoid robot is powered by onboard batteries which restrict the available energy to a defined limit. So, the planning for humanoid movements should result in minimum energy consumption. Simulator expects the consumed current, $\tilde{I}$, based on the joints torque efforts, T_i, as presented in equation (8), where N is the number of servo motors, I_{supply}^{SV} is the supply current of servo motors even with no torques, I_{supply}^{CM5} is the supply current of control module and K is a generic gain that can be found through some experimental results.

$$\tilde{I} = K \cdot \sum_{i=1}^{N} T_i + N \cdot I_{supply}^{SV} + I_{supply}^{CM5} \qquad (8)$$

As result, both currents consumptions, measured in the real robot and estimated by the simulator during a stand-up movement are presented in Fig. 14 and Fig. 15.

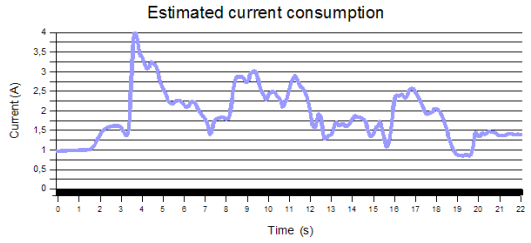


Fig. 14 Simulator stand-up estimated current consumption.

The similar appearance of graphics hints the accuracy of the simulator but clearly some tuning is still required to achieve a better match.

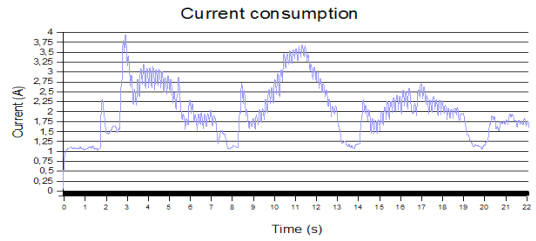


Fig. 15 Real robot stand-up current consumption.

5. CONCLUSION AND FUTURE WORK

The presented results allow to validate the proposed realistic simulator: the power consumption and stand-up routines simulation were achieved successfully in the robot simulator, making the simulation very realistic.

As future work, walk and ball dribbling movements should be developed, based on described simulator, and migrated to the real robot with a minimum overhead. Furthermore, the high level programming can be made resorting to a text based script window in order to allow users to create their own control programs: a script window that accepts Pascal language code should be developed. User can implement several controllers and movements planning where real-time results are presented.

REFERENCES

Behnke, S., Schreiber, M., Stuckler, J., Renner, R. and Strasdat, H. (2006). See, Walk, and kick: Humanoid robots start to play soccer. *International Conference on Humanoid Robots*. IEEE, Genova, Italy.

Browning, B. and Tryzelaar, E. (2003). Übersim: A Multi-Robot Simulator for Robot Soccer *Autonomous Agents and Multi-Agent Systems*. - Australia.

GLScene (2000). http://glscene.sourceforge.net.

Gonçalves, J., Pinheiro, P., Lima, J., Costa, P. (2007). Tutorial introdutório para as competições de futebol robótico, *IEEE Latin-American Learning Technologies Journal*. **Vol. 2**, N 2, pp. 63-72.

Humanoid League (2007). http://www.humanoidsoccer.org/.

Kagami, S., Takahashi, Y., Nishiwaki, K., Mochimaru, M. and Mizoguchi, H. (2004). High-speed matrix pressure sensor for humanoid robot by using thin force sensing resistance rubber sheet, Proceedings of IEEE Sensors, IEEE Xplore.

Kajita S., Morisawa, M., Harada, K., Kaneko, K., Kanehiro, F., Fujiwara, K., Hirukawa, H. (2006). Biped Walking Pattern Generator allowing Auxiliary ZMP Control, *International Conference on Intelligent Robots and Systems*, Beijing, China.

Park, J. and Ohung, K. (2001). Reflex Control of Biped Robot Locomotion on a Slippery Surface, *IEEE International Conference on Robotics & Automation*, Seoul, Korea.

Park, J. and Kim, K. (1998). Biped Robot Walking Using Gravity-Compensated Inverted Pendulum Mode and Computed Torque Control, *IEEE International Conference on Robotics & Automation*, Leuven, Belgium.

Renner, R. and Behnke S. (2006). Instability detection and fall avoidance for a humanoid using attitude sensors and reflexes, *International Conference on Inteligent Robots and Systems*, Beijing, China.

Robocup, (2007). http://www.robocup.org/.

Santos, V. and Silva, F. (2006). Design and Low-Level Control of a Humanoid Robot Using a Distributed Architecture Approach, *Journal of Vibration and Control*, SAGE Publications, **vol. 12**, pp. 1431-1456.

Smith, R. (2000). Open Dynamics Engine http://www.ode.org/.

Stückler, J., Schwenk J. and Behnke S. (2006). Getting Back on Two Feet: Reliable Standing-up Routines for a Humanoid Robot, *9th International Conference on Inteligent Autonomous Systems*, pp. 676-685, Tokyo, Japan.

Suzuki, T. and Ohnishi, K. (2006). Trajectory Planning of Biped Robot with Two Kinds of Inverted Pendulums, *12th International Power Electronics and Motion Control Conference*, Portoroz, Slovenia.

Tribotix (2004). http://www.tribotix.com/index.html.