Realistic Behaviour Simulation of a Humanoid Robot

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

***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 proposed accurate simulator allows to fulfil this goal. The simulator is based on the Open Dynamics Engine and GLScene graphics library, providing instant visual feedback. User is able to test any control strategy without bringing damage to 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 and simulator results are presented in order to validate the proposed simulator.**

I. Introduction

In last 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, and two legged robots are also able to move in human environment. So, studies about biped robots are very important [1]. Locomotion under influence of external disturbances is a challenging task for a humanoid robot once if disturbances are large enough, a fall might become unavoidable. Closed loop controllers should minimize the number of falls [2] and if a fall happens, the robot should detect it, and get back into an upright posture [3]. On the one hand, simulation is a powerful tool for speeding up the control software development. On the other hand, developing new control software for robots can be a difficult and challenge task. The ability to rapidly prototype software, within a simulation environment, can be of great benefit to develop robot control if the resulting software can be easily transfered from simulation to the reality. Therefore, the simulator must capture the most important environment characteristics; however, developing simulators with high-fidelity dynamic models that can be simulated in real-time is a non trivial problem [4]. 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. Joints angles and torques limits must also be handled. A screenshot of the developed simulator is shown in Fig. 1, where a 3D scene shows the robot human-like, a graphic shows the desired time variables and a table shows the angle, angular speed and torque for each robot joint.

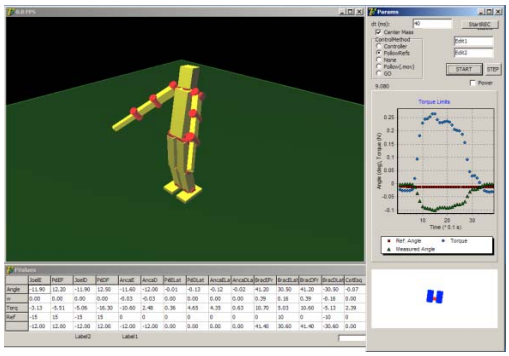


Fig. 1. Simulator screenshot.

There are several robot simulators, such as Simspark, Webots and MURoSimF, that provides a humanoid 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 some overhead. Code migration from general realistic simulators to real world systems is the key for reducing development time of robot control, localization and navigation software. Due to the complexity of robot, world, sensors, and actuators modelling it is not an easy task to develop such simulator. 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, based on the Open Dynamics Engine [5], for a humanoid robot and compares it to the real robot. The proposed simulator allows to design and test behaviours without access to the real hardware in order to carry out research on robot control.

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 controller issues. 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.

II. Real Humanoid Robot

The commercially available Bioloid [6] robot kit, from Robotis, is the basis of the used humanoid robot. The

José L. Lima and José C. Gonçalves are assistant professors in the Electrical Engineering Department of Polytechnic Institute of Bragança. {jllima, goncalves}@ipb.pt.

Paulo G. Costa and A. Paulo Moreira are Auxiliar Professors in the Department of Electrical Engineering and Computers in Faculty of Engineering of University of Porto. {paco, amoreira}@fe.up.pt.

overview of the proposed biped robot is shown in Fig. 2. The modified robot, defers from the original kit, following the dimensional rules of RoboCup [7] Humanoid League [8].

Fig. 2. Real Humanoid robot standing.

A. *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 2 *kg* and its height is 38 *cm*.

B. *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 [9]. 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 each individual servo can be addressed or a broadcast can be sent. These limitations should be placed in the simulator as presented in final sections.

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. 3.

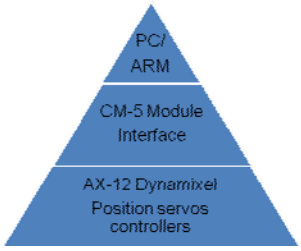


Fig. 3. Interface layer.

The original firmware was replaced in order to get higher performances and low level controller achieve. At the next layer, target angle and moving speed for the individual joints are generated from a personal computer or from an embedded system. Each broadcast communication takes about 12 ms. Therefore, the fastest allowed control and sampling rate of the servo motors is about 83 Hz.

C. *Behaviour and Control*

Perception assumes a major role in an autonomous robotics, and must be therefore reliable or abundant [10]. For this robot, the joint position, speed and torque perception was planned. For enlarge the closed loop control, as a future feature, the accelerometers information and feet force sensing perceptions were also planned [11]. At the present, the real humanoid robot is not equipped with accelerometers and for similarity the simulator's sensors were not included in the simulation.

As a first approach, an open-loop system can be used (accelerometers and feet force information are 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 developed software that communicates with CM-5 module and controls the robot is presented in Fig. 4.

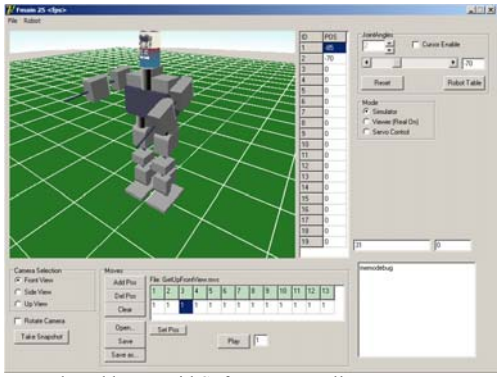


Fig. 4. Developed humanoid Software controller.

III. 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 [4]. For real-time simulation, an accurate but fast simulation engine must be used. ODE, Open Dynamics Engine [5] 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 graphic routines are based on open 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 [12].

A. *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. ODE joints, 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.

GLScene is used to render the 3D graphics appearance enhancing visualization including shadows, textures, projections, illuminations and it also provides depth perception.

B. *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. 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. 5 a) and b), that shows the used symbology and some examples of possible trajectories. It is necessary to calculate the angle equation that result in the desired conditions.

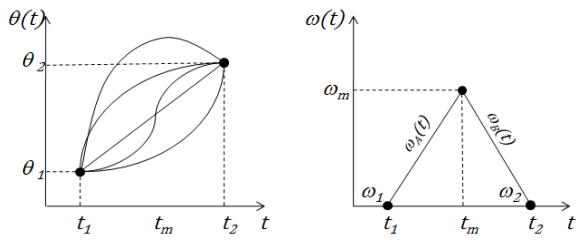


Fig. 5. Joint angles and angular speeds (actual θ_1, $\omega_1$and desired θ_2, ω_2).

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. 5 b) 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 covered angle can be determined through the integral of the angular speeds as presented in equation (1).

$$\theta_2 - \theta_1 = \int_{t_1}^{t_m} \omega_A(t)dt + \int_{t_m}^{t_2} \omega_B(t)dt \qquad (1)$$

Equation (1) gives the desired value for ω_m presented in equation (2).

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

Then, angle reference equation, for each *i* joint, can be described in equation (3) for $t_1<t\leq t_m$ and in equation (4) for $t_m<t\leq t_2$.

$$\theta_i^{ref}(t) = \theta_1 + \omega_1 \cdot t + \frac{1}{2}\cdot\frac{\omega_m-\omega_1}{t_m-t_1}\cdot t^2 \qquad (3)$$

$$\theta_i^{ref}(t) = \theta_{(t=t_m)}^{ref} + \omega_2 \cdot t + \frac{1}{2}\cdot\frac{\omega_2-\omega_m}{t_2-t_m}\cdot t^2 \qquad (4)$$

The presented equations (1 to 4) 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.

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 (5), where $\theta_i^{ref}(t)$ is calculated by equations (3) and (4) and $\omega_i^{ref}(t)$ is the interpolation for $\omega_i(t)$. Gains constants K_i^θ and K_i^ω depend on each joint due to its submitted effort.

$$T_i(t) = K_i^\theta \cdot \left(\theta_i^{ref}(t) - \theta_i(t)\right) + \\ + K_i^\omega \cdot \left(\omega_i^{ref}(t) - \omega_i(t)\right) \qquad (5)$$

A detailed graph (with a small angle scale), presented in Fig. 6, shows how the low-level controller follows the θ_i^{ref}(t) (Ref controller) guiding θ_i(t) (Measured angle) to the requested angle (Angle ref.), based on equation (5). The simulator closed loop control frequency is the same as used in 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 robot 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 [13].

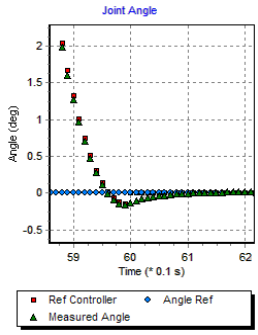


Fig. 6. Detailed joint angle low-level controller.

As a final result, presented in Fig. 7, the left arm joint angle $\theta_i(t)$ (Measured angle) follows the θ_i^{ref}(t) (Ref controller).

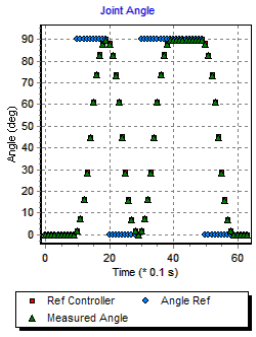


Fig. 7. Joint angle controller.

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.

C. *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 [14], with two kinds of inverted pendulums [15], using Gravity-compensated inverted pendulum mode [1], or Zero Moment Point (ZMP) pattern generation [16]. 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 centre of gravity, COG, can be determined and hip angles controller allows to guarantee the desired stability. The closed loop control is done resorting to COG calculation as presented in next sections. The ZMP estimation is based on COG.

IV. SIMULATION AND REAL ROBOT BEHAVIOUR RESULTS

This chapter presents a comparison between the simulator and the real robot behaviour. As the simulator was developed having in mind the real robot, there is a direct mapping between both sides. Equilibrium, torques and angles limitations, stand-up movements and current consumption were successfully tested and shown in next subsections.

A. *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. The robot ZMP is estimated by the distance between the centre of mass (CM) and the floor contact convex hull. The simulator draws, in real-time, the ZMP and the centre of mass. As examples, presented in Fig. 8, two body positions, push-up a) and standing c), result in the ZMP drawn at the right side, b) and d).

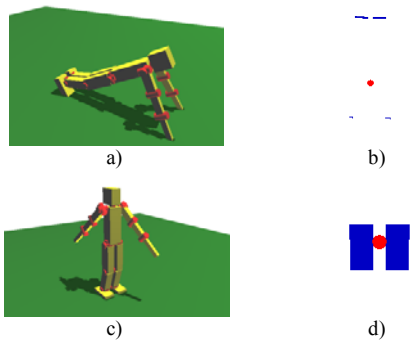
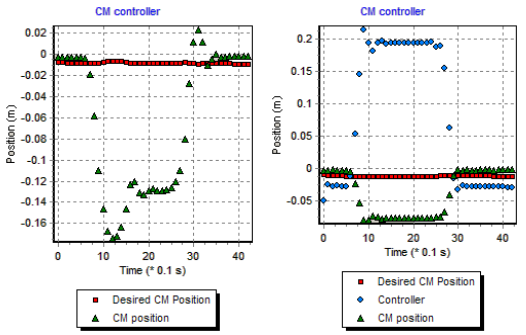


Fig. 8. ZMP drawing.

ZMP can also be used to generate movements patterns ZMP that allows a smooth and soft motion [16]. Using the ZMP to control the haunch, allows to improve the equilibrium issue. The centre of mass estimation and the desired centre of mass position deviation can control the haunch joint through a PD controller. Figures 9 a) and b) present the centre of mass position during a frontal arm elevation without and with the

controller. Of course this issue allows to correct only a small gap that improves the stability. The desired CM position was the geometric centre of feet for each direction. As result, the centre of mass remains more stable and in some cases avoids a robot falling.



a) b)

Fig. 9. CM controller data graph.

B. *Angles and Torques limitations*

As it is well known, real systems deal with limited signals, opposite of virtual and simulated applications. For simulator to have the appearance of the real robot, joint angles and servo motors torques must be limited to the real values. As result, Fig. 10 presents an arm elevation angle limitation (120 degrees) for a reference of 200 desired degrees. The controller tries to push the joint angle to the desired angle but the joint limit forbids it.

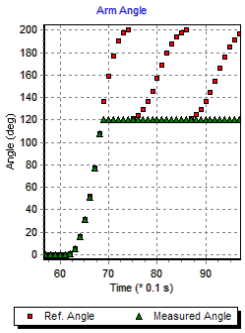
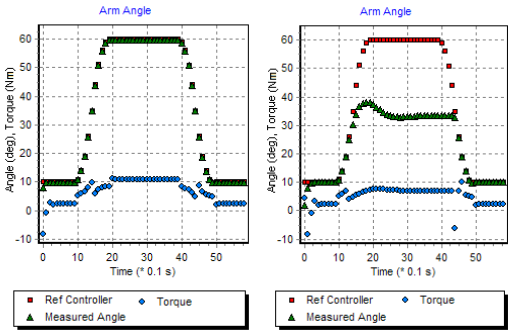


Fig. 10. Angle limitation.

As the torque is the basis to control the whole body and its joints, the simulator can limits its values to the desires constants, appearing as a real robot. Figure 11 a) presents an arm elevation joint from 0 to 60 degrees step and its return reference signal, the measured angle and the applied torque that allows to obtain the desired conditions. Figure 11 b) presents the same desired conditions but a torque limit of *7 N* is now presented. The arm elevation is now damaged and is not able to go beyond 35 degrees. A viscosity constant must be introduced in simulation otherwise joint angle will be unstable, unlike reality as presented in Figure 12. The viscosity can be added as presented in equation 6 where T_A is the torque to be applied, T_{MAX} is the torque limit, K_v is the viscosity constant and ω_i is the angular speed for each i joint.

$$T_A = T_{MAX} - K_v \cdot \omega_i \quad (6)$$



a) b)

Fig. 11. Torque limitation with friction.

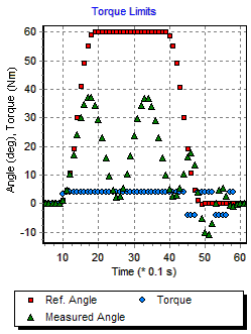


Fig. 12. Torque limitation.

C. *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 [3].

The simulator and the real robot stand up movements comparison is made in Fig. 13 and Fig. 14. The same set of parameters was applied in simulation and real robot with a small arrangement.

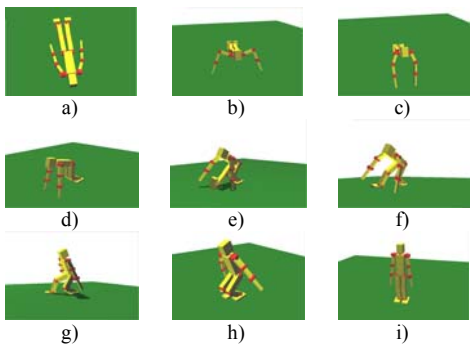


Fig. 13. Simulator stand-up movements.

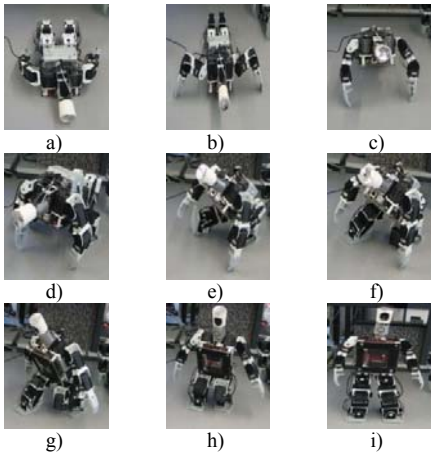


Fig. 14. Real robot stand-up movements.

D. *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 (7), 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.

$$\tilde{I} = K \cdot \sum_{i=1}^{N} T_i + N \cdot I_{supply}^{SV} + I_{supply}^{CM5} \qquad (7)$$

As result, both currents consumptions, measured in the real robot and estimated by the simulator during a stand-up movement are presented in Fig. 15 and Fig. 16. The similar appearance of graphics hints the accuracy of the simulator but clearly some tuning is still required to achieve a better match.

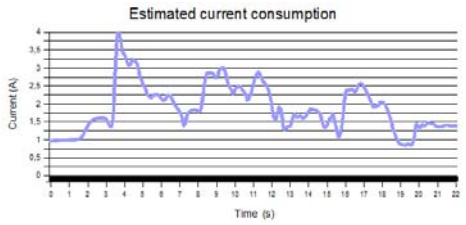


Fig. 15. Simulator stand-up estimated current consumption.

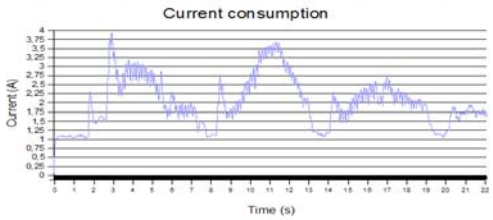


Fig. 16. Real robot stand-up estimated current consumption.

V. Conclusion and Future Work

In this paper a humanoid simulator, based on a dynamics engine and the friendly appearance of 3D scenes, is presented and compared to a real robot. The presented results allow to validate the proposed approach: 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 tested. 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

[1] Suzuki, Tomoyuki, and Kouhei Ohnishi. “Trajectory Planning of Biped Robot with Two Kinds of Inverted Pendulums.” 12th International Power Electronics and Motion Control Conference. Portoroz, Slovenia: IEEE, 2006

[2] Renner, Reimund, and Sven Behnke. “Instability detection and fall avoidance for a humanoid using attitude sensors and reflexes.” International Conference on Inteligent Robots and Systems. Beijing, China, 2006. 2967-2973.

[3] Stückler, Jörg, Johannes Schwenk, and Sven Behnke. “Getting Back on Two Feet: Reliable Standing-up Routines for a Humanoid Robot.” 9th International Conference on Inteligent Autonomous Systems. Tokyo, Japan, 2006. 676-685.

[4] Browning, Brett, and Erick Tryzelaar. “Übersim: A Multi-Robot Simulator for Robot Soccer.” Autonomous Agents and Multi-Agent Systems. Australia, 2003.

[5] Smith, Russell. Open Dynamics Engine. 2000. http://www.ode.org/.

[6] Tribotix. 2004. http://www.tribotix.com/index.html.

[7] Robocup. 2007. http://www.robocup.org/.

[8] Humanoid League. 2007. http://www.humanoidsoccer.org/.

[9] Behnke, Sven, Michael Schreiber, Jörg Stückler, Reimund Renner, and Hauke Strasdat. “See, Walk, and kick: Humanoid robots start to play soccer.” International Conference on Humanoid Robots. Genova, Italy: IEEE, 2006.

[10] Santos, Vítor M. F., and Filipe M. T. Silva. “Design and Low-Level Control of a Humanoid Robot Using a Distributed Architecture Approach.” Journal of Vibration and Control, 2006: 1431-1456.

[11] Kagami, S., Y. Takahashi, K. Nishiwaki, M. Mochimaru, and H. Mizoguchi. “High-speed matrix pressure sensor for humanoid robot by using thin force sensing resistance rubber sheet.” Sensors, 2004. Proceedings of IEEE, 2004: 1534-1537, Vol.3.

[12] GLScene. 2000. http://glscene.sourceforge.net.

[13] Gonçalves, José, Pedro Pinheiro, José Lima, and Paulo Costa. “Tutorial introdutório para as competições de futebol robótico.” IEEE Latin-American Learning Technologies Journal , 2007: Vol 2, Núm 2, 63-72.

[14] Park, Jong Hyeon, and Ohung Kwon. “Reflex Control of Biped Robot Locomotion on a Slippery Surface.” IEEE International Conference on Robotics & Automation. Seoul, Korea, 2001.

[15] Park, J.H. Kim, K.D. “Biped Robot Walking Using Gravity-Compensated Inverted Pendulum Mode and Computed Torque Control.” IEEE International Conference on Robotics & Automation. Leuven, Belgium, 1998.

[16] Kajita, Shuuji, et al. “Biped Walking Pattern Generator allowing Auxiliary ZMP Control.” International Conference on Intelligent Robots and Systems. Beijing, China: Proceedings of the 2006 IEEE/RSJ-, 2006. 2994-2999.