Theory and Applications of Mathematics & Computer Science

(ISSN 2067-2764)
http://www.uav.ro/applications/se/journal/index.php/tamcs

Theory and Applications of Mathematics & Computer Science 1 (2011) 17–33

SimTwo Realistic Simulator: A Tool for the Development and Validation of Robot Software

Paulo Costa[a,c,*], José Gonçalves[b,c], José Lima[b,c], Paulo Malheiros[a,c]

[a] *Faculty of Engineering of the University of Porto, DEEC, Porto, Portugal.*
[b] *Polytechnic Institute of Bragança, Department of Electrical Engineering, Bragança, Portugal.*
[c] *Robotics and Intelligent Systems research group from INESC PORTO, Portugal.*

Abstract

Many times the simulation environment is an ad hoc implementation done with the single purpose of testing the author's algorithm or methodology. It is not difficult to find that the accuracy of the simulation is low and mostly unable to reflect the real world. While, under certain circumstances, that can be a valid approach there is the need of a simulation environment where the focus is on the physical realism while retaining a lot of configurability so that it can be adapted to a lot of different uses. In this paper it is presented a simulation environment where multiple kinds of robots can be modeled. The robots can be modeled based on a network of physical bodies interconnected by joints that can be powered, or not, by electrical motors. The corresponding low level controllers and the high level decision or IA can also be implemented in the simulation environment. To achieve that, some established open source libraries like the Open Dynamics Engine are used. The parameters for the physical simulation are all accessible when defining the robot model and a very accurate motor model can also be implemented. There is also the ability to use external modules to implement the high level controllers.

Keywords: Robotics, Simulation, Sensors, Actuators.

1. Introduction

Simulation has established itself as an important tool in the mobile robotics field. The ability to test and develop robotic solutions in a virtual environment is widely used in research as well as in education, allowing the rapid development of robot software. In this context, having in mind the current needs of the authors, concerning both education and research it was developed SimTwo, being a versatile robot simulation environment that allows rapid test and design of differential, omnidirectional, industrial, humanoid robots, etc., developed in Object Pascal. SimTwo has a set of predefined components such as sensors and motors where specified models are inputed. The Open Dynamics Engine is used for the simulation of the rigid body dynamics. The robots look and behavior are defined in XML format files. The virtual world is represented using GLScene components (GLScene, 2010), these provide a simple implementation of OpenGL (OpenGL, 2010). The main motivation for the authors to develop a robot simulator was to have full control of the simulation features and its accuracy. The authors develop robot software and teach courses that use this tool as a teaching aid. They can add new features at any time and also improve the existing ones, never becoming limited by the features of the existing simulators (Gonçalves *et al.*, 2010).

For the simulator development it was used ODE (Open Dynamic Engine) (ODE, 2010), that provides an excellent real time time simulation, embedding it in a 3D application. ODE is an open source, high performance library for

*Corresponding author
Email addresses: paco@fe.up.pt (Paulo Costa), goncalves@ipb.pt (José Gonçalves), jllima@ipb.pt (José Lima), paulo.malheiros@fe.up.pt (Paulo Malheiros)

simulating rigid body dynamics. It is fully featured, stable, mature and platform independent with an easy to use C/C++ API. It has advanced joint types and integrated collision detection with friction. ODE is useful for simulating vehicles, objects in virtual reality environments and virtual creatures. It is currently used in many computer games, 3D authoring tools and simulation tools. There are other alternatives for physics Engines such as Box2D, Bullet, Chipmunk physics engine, SOFA (Simulation Open Framework Architecture), Tokamak physics engine and Jinngine (Java Physics Engine). It has also been made an effort to promote the use of different physics engines, for example using the Physics Abstraction Layer™(PAL), this provides a unified interface to a number of different physics engines. This enables the use of multiple physics engines within one application.

Some of the 3D simulators that are free and can be added new features are Gazebo Open Simulator and Simulator BOB. Gazebo is not open source although new Plugins can be uploaded with the user sensor and actuator modules. Open Simulator is open source and it can also be expandable through loadable modules to build complete custom configurations. Simulator BOB: 3D simulation environment for mobile robots is open source, with the goal to speed up its developing time, it is being developed for further research with artificial intelligence and autonomous mobile robots. Some of the commercial softwares available are Microsoft Robotics Studio, Webots by Cyberbotics and Marilou by Anycode (Michel, 2004).

There are also 2D simulation software suitable to develop software, mainly used in the area of artificial intelligence, not being important, for the robot software development, the interaction of the robot with a complex 3D world. Two examples of 2D simulators are Stage and Khepera simulator. Stage allows the simulation of a population of robots and the Khepera Simulator allows to develop controllers for the Khepera robot (Michel, 1996) (Wolfer & Rababaah, 2005) (Hassanzadeh *et al.*, 2008).

SimTwo is a free software that is not open source. Although it is not possible to upload new modules it is very flexible, providing a great variety of sensors and actuators and allowing to prototype and rapidly reconfigure different commercial and non commercial robots. One of the SimTwo important features is the ability of the use of remote clients that allow the development of robot software in different programming languages and commercial softwares such as MATLAB and LabVIEW. This feature is also available in both commercial and non commercial softwares, such Webots and Open Simulator. SimTwo has also a built in script controller that can communicate through serial port with micro-controllers and through Network with other applications. In section 2 is presented a description of the SimTwo simulator. In section 3 it is presented the modeling and simulation of a robot based on the Lego Mindstorms NXT Kit, where is given special focus to the actuator modeling and simulation. In section 4 some SimTwo applications examples are presented: an omnidirectional wheeled robot, a humanoid robot, and lighter-than-air vehicle. In the simulation of the wheeled robot it is presented how to develop the robot software with the purpose of its localization and navigation in a structured environment. The localization algorithm is achieved resorting to an Extended Kalman Filter, merging data provided from encoders (relative measurements) and distance data provided by infrared distance sensors. Finally some conclusions and future work are presented.

2. SimTwo – Realistic Simulator

SimTwo is a realistic simulation system that can support several types of robots. Its main purpose is the simulation of mobile robots that can have wheels or legs, although industrial robots, conveyor belts and lighter-than-air vehicles can also be defined. Basically any type of terrestrial robot definable with rotative joints and/or wheels can be simulated in this software. Figure 1 shows the software with all its main windows.

The dynamics realism in SimTwo is obtained by decomposing a robot in rigid bodies and electric motors. Each body behaviour is numerically simulated using its physical charatheristics: shape, mass and moments of inertia, surface friction and elasticity. It is also possible to define standard joints such as socket, hinge and slider which can be coupled with an actuator or a sensor.

SimTwo is an application with a *multiple document interface* (MDI) where all windows are under the "world view" window control, shown in Figure 1, exiting the simulator is done by closing this window. The "configuration" window offers control over several elements of the virtual scene. It is possible to define the controller timestamp and to configure the 3D world view (camera position, shadow visibility, etc.). Robots information is also displayed in this window, like its position and speed.

The "code editor" offers an *integrated development environment* (IDE) for high-level programming based in Pascal language, this is the main tool in this simulator. The control algorithms are directly compiled in this window, a message

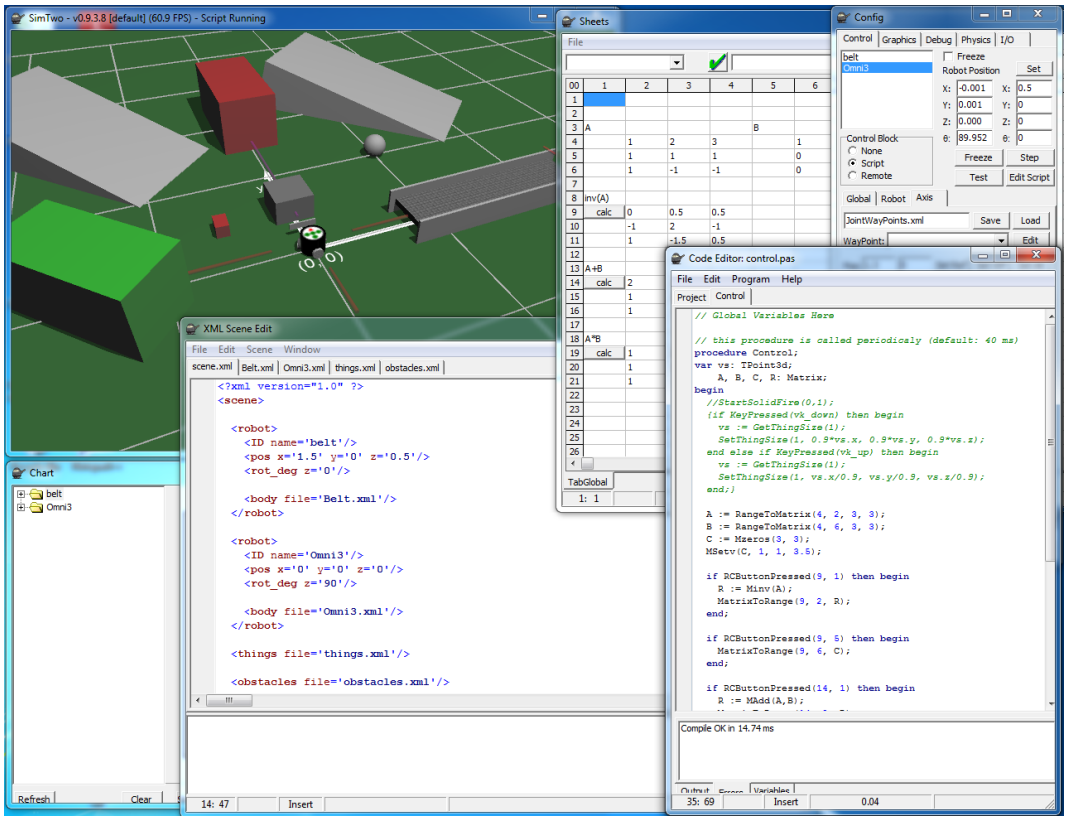

Figure 1: SimTwo software. Application windows clockwise from top left corner: world view, spreadsheet, configuration, code editor, scene editor and chart.

appears in the bottom of the page informing of a successful compilation or the existence of coding errors. The control script is started from this window where the resulting robot movement is visible in the main window, any changes to the control script requires this to be stopped and recompiled.

Debugging the control algorithm is possible using the “chart” and the “spreadsheet” window. In the “chart” it is possible to plot all variables available for every robot, such as its position, motors speed and current, etc. In the “spreadsheet” window it is possible to define “edit cells” as well as “button cells” for specific operations. This window becomes a customizable form window, this is the equivalent to a graphic application.

The scene implementation is done by editing several XML format files, these files are definable in the “scene editor” window. A scene in SimTwo can have “robots”, “obstacles” and “things”, as shown in Figure 2.

A main scene file (scene.xml) defines the robots in use and their specific construction file. Each robot is defined by various solids (cuboid, cylinder and sphere) connected through joints (slider, socket and hinge). The shell elements are solids without mass, these do not modified the robot physical properties but are an essential part for collision simulation. A robot can also have sensors, these provide information from the environment surrounding the robot.

The scene objects are the “obstacles” and “things”, these are very similar in definition (both are defined solely by solids) but while the “obstacles” are imovable in case of collision the “things” are not. If a robot colides with a “thing” these will react accordingly to their mass definition. A scene can also have sensors, these are static relative to the world as opposed to a robot sensor which gives information relative to its corresponding robot.

In Figure 3 it is presented the flux of information of a SimTwo controller. The controller has different levels that are updated at different rates, being presented its default values. The presented default values can be changed by the user depending on the application that is being simulated. The artificial intelligence controller is updated by default at 40 ms, being a common rate to accomplish real time requisites in mobile robotics challenges. The motor controller is updated by default at a 10 ms rate and its model output is updated by default at 1 ms rate. The default values

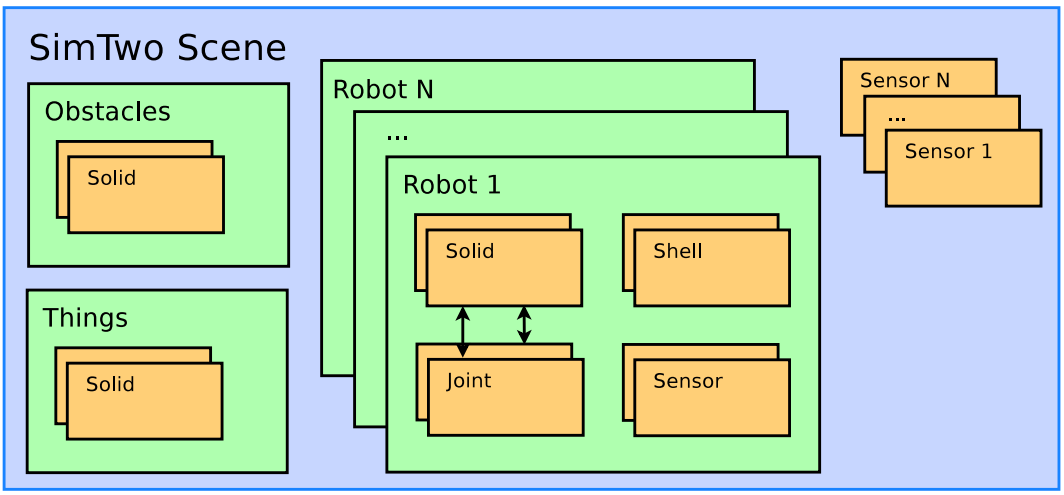

Figure 2. SimTwo Scene structure

are typical values that were chosen having in mind the dynamics of the world and motor models in mobile robotics, although user can change this values depending on the specif dynamics and real time requisites of its simulation.

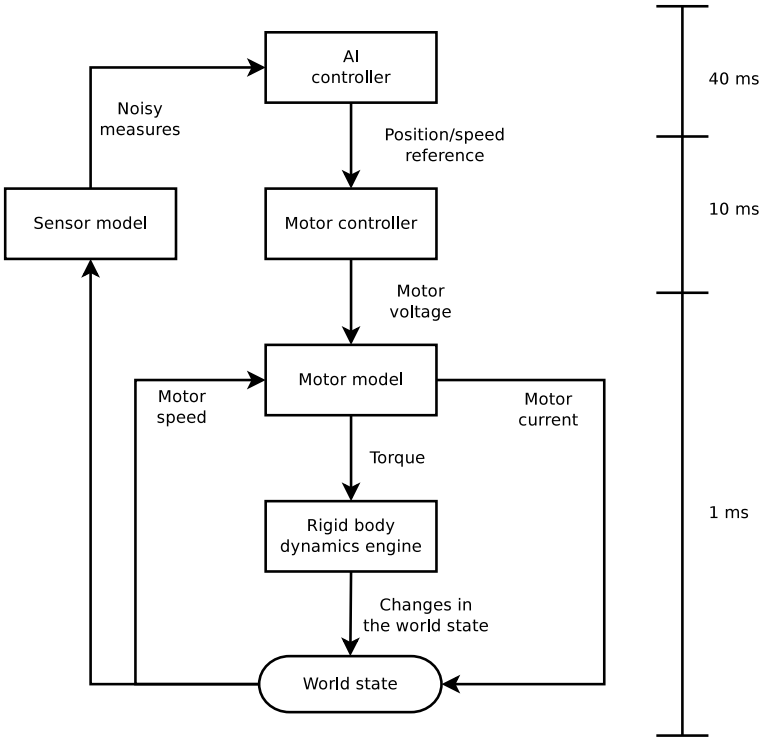

Figure 3. SimTwo controller cycle

The artificial intelligence level is provided with sensor information extracted from the world state. After some decisions and calculations related with control, localization and navigation the controller provides to the motor controller its inputs. The motor model provides the torque information, forcing the Rigid Dynamics Engine to react and consequently forcing the World state (Sensor, obstacles and robot positions) to change.

3. Modeling and simulation of a Lego Mindstorms Robot

Lego Mindstorms is a powerful educational tool (Gawthrop & McGookin, 2006) (Lund & Pagliarinis, 2000), being used by the authors in many activities. It has been used to teach mobile robots introductory concepts to undergraduate and graduate students and in several demonstrations. The undergraduate students while attending summer courses prototype their own robots and develop robot software based on the Lego Mindstorms educational software. The graduate students while attending robotics classes also prototype their own robots, with the difference that the developed robot software is done resorting to high level programming languages. It has also been used in demonstrations with the goal of captivate and inform undergraduate students, concerning the areas that involve technology.

In this section it is described the modeling of an NXT based robot and how it can be simulated using SimTwo, giving a special attention to the actuator model and simulation (Campion *et al.*, 1996) (Khosla, 1989) (Olsen & Petersen, 2001). It is not presented the sensor modeling and it is not presented the comparative between the simulated and the real robot, this results can be found in (Gonçalves *et al.*, 2009*b*). The presented approach does not replace the training with hardware but is an important complement. Students can develop and test robot controllers even at their homes, without the need of having access to hardware. This situation is important because usually students have access to hardware only when they are attending classes or when they have tutorial classes at the robotics laboratory. The simulation is also important to monitor some interesting variables, that are much more difficult to access in the real world. As a direct benefit the robot behavior can be monitorized in a more accurately way.

The Lego modularity makes the rapid prototyping of different robot configurations easier. This easiness presents itself as an extra motivation for the persons who are taking their first steps in the world of mobile robotics. The Lego parts allow connectivity, suppressing the need of using glue or screws. It is also an ecological tool because although it is not easy to recycle plastic, the Lego parts can be reused. It is a modular tool, it is possible for the same part to be a different thing depending on the application, motivating those who are using it and it is presented at a relatively low cost. Added to all the presented advantages of the Kit Lego Mindstorms, simulation can also be a very important teaching aid.

The modeled and simulated prototype, presented in Figure 4, is a differential robot. Its movement is controlled by varying the velocity of each wheel independently.

Figure 4. Driving Base of the Lego Mindstorms NXT

Design behavior without real hardware is possible due to a physics-based simulator implementation. The dynamic behavior of the simulated robot is computed by ODE Open Dynamics Engine, a free library for simulating rigid body

dynamics. The simulator architecture is based on the real Lego NXT robot. The body masses and dimensions are followed in order to build a simulated robot like the real one.

3.1. Actuator modeling

The Lego Mindstorms NXT kit provides three servomotors with built-in rotation sensor and a gear ratio of 1:48. A Lego Mindstorms NXT servomotor is shown in Figure 5.

Figure 5. Lego Mindstorms NXT servomotor.

The servomotor can be resumed to a DC motor model, presented in Figure 6, where U_a is the converter output, R_a is the equivalent resistor, L_a is the equivalent inductance and e is the back emf (electromotive force) voltage as expressed by equation (3.1).

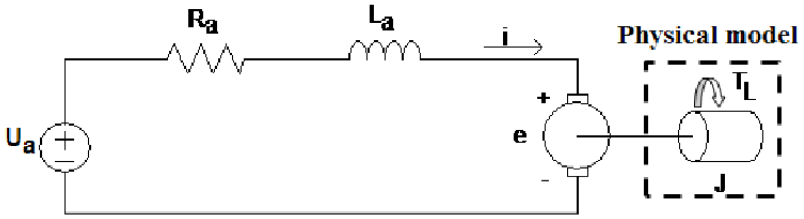

Figure 6. DC motor electric model.

$$U_a = e + R_a i_a + L_a \frac{\partial i_a}{\partial t} \tag{3.1}$$

The motor can supply a torque T_L and the load has a moment of inertia J that will be computed by the physical model ODE. Current i_a can be correlated with the developed torque T_d through equation (3.2) and the back emf voltage can be correlated with angular speed through equation (3.3), where K_s is a motor parameter that can be found by an experimental setup as presented in subsection 3.1.1 (Bishop, 2002).

$$T_d(t) = K_s i(t) \tag{3.2}$$

$$e(t) = K_s \omega(t) \tag{3.3}$$

In fact, the real developed torque (useful) that will be applied to the load (T_L) is the developed torque subtracted by the friction torque (T_c).

$$T_L = T_d - T_c \tag{3.4}$$

3.1.1. DC motor model measurements:

It was used the NXT Lego Mindstorms motor as the base of the simulator. The R_a and L_a values can be directly measured (R_a=7.6 Ω and L_a= 4.88 mH). The K_s motor parameter can be found by an indirect measure. For several angular speeds, it can be measured the emf voltage while the motor is in open circuit. Table 1 presents the data for 7 measures.

Table 1. Speed and emf voltage.

ω (rotations/min)	ω (rad/s)	e (V)	k=e/ω
0	0	0	
66	6.91	3.4	0.491
115	12.04	6.0	0.498
155	16.23	7.9	0.487
195	20.42	9.8	0.479
233	24.39	11.9	0.487
265	27.75	13.85	0.499

Figure 7 shows graphical data of the K_s line and its trend line. The average value for K_s (line slope) is about 0.4919 V.rad/s. It is possible to observe that the error in the origin for the presented approximation it is negligible, being only 0.0218 V. Despite being a small error it is possible to increase the precision in the K_s parameter estimation, forcing the trend line to pass in the origin. The obtained value for K_s is nearly 0.4908 V.rad/s.

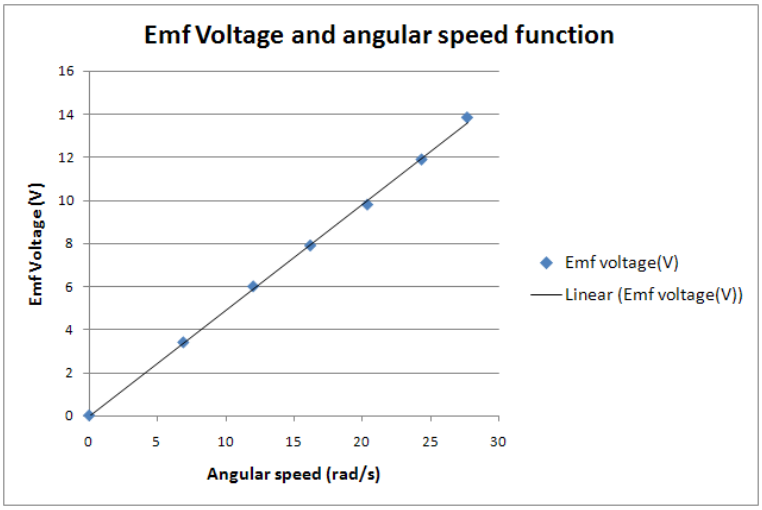

Figure 7. K_s value for DC motor model.

In order to calculate the friction it was obtained the motor angular velocity for different voltages at its terminals. After a linear regression it was obtained equation (3.5).

$$\omega = 1.9248v - 0.2519 \tag{3.5}$$

Where ω is the angular velocity in rad/s and v is the voltage at the motor terminals.

Assuming that the developed torque (T_d) is equal to the torque generated by the static friction (T_c) it is possible, resorting to the linear regression presented in equation (3.5), to obtain the necessary voltage to equal the static friction (v_c), by replacing by zero. For this situation it was not necessary to include in the model the viscous friction, because it only exists for angular velocities different from zero. As there is no movement there is no voltage generated by the emf and as the current has no variation it is possible to obtain the produced torque by the static friction resorting to

equations (3.1) and (3.2), resulting in equation (3.6). The torque due to the static friction has a numerical value of 3.558E-4 Nm.

$$T_c = K_s \frac{v_c}{R_a} \tag{3.6}$$

After obtaining the static friction there is only left, to complete the motor model, to estimate B, the parameter that relates the viscous friction with the motor angular velocity, as shown in equation (3.7).

$$T_\omega = B\omega \tag{3.7}$$

Placing the motor spinning without load the developed torque will given by equation (3.8).

$$T_d = T_c + B\omega \tag{3.8}$$

As the developed torque is proportional to the current as shown in equation (3.2) it is possible to conclude, from equation (3.8), that the current can be given by equation (3.9).

$$i_a = \frac{T_c + B\omega}{K_s} \tag{3.9}$$

The voltage applied to the motor terminals can be given by equation (3.1), but as in steady state the current variations are small, it can be obtained equation (3.10) for the motor terminal voltages.

$$U_a = K_s\omega + R_a i_a \tag{3.10}$$

where e was replaced by $K_s\omega$ as exemplified in equation (3.3).
From equations (3.9) and (3.10) it is obtained equation (3.11).

$$\omega = \frac{U_a - \frac{R_a T_c}{K_s}}{\frac{R_a B}{K_s} + K_s} \tag{3.11}$$

Minimizing the sum of the absolute error between the real angular velocity and the obtained by the model (equation (3.11)), it is obtained a numerical value for B (1.92E−3).
The estimated motor parameters are shown in Table 2.

Table 2. Motor parameters

Parameters	Value SI units
K_s	0.4908
T_c	3.558E-4
B	1.92E−3
R_a	7.6
L_a	4.88E−3

The different obtained motor models can be observed in the graphics shown in Figure 8, where *rot* is ω (the motor angular velocity).

3.1.2. DC motor nonlinearities:

In a way to map the reality closer, where variables cannot assume all values, the model must have some limitations. The first one, the voltage applied to the supply terminals U_a. This voltage should be limited to the batteries voltage. Further, current i should be limited once it is related to the torque through equation (3.2). Figure 9 shows the limitations presented in the servomotor model.

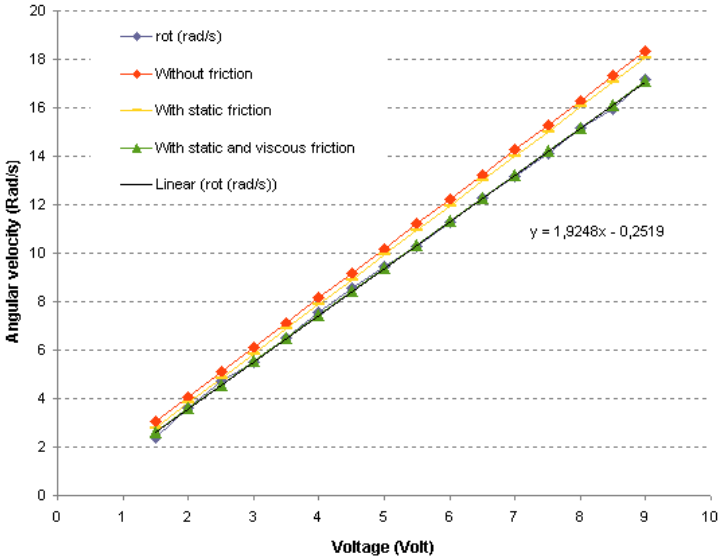

Figure 8. Different obtained models.

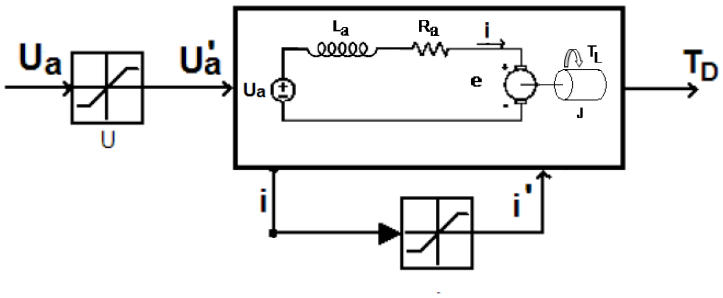

Figure 9. Servomotor model nonlinearities.

3.2. Robot simulation based on the SimTwo

The robots look and behavior are defined in XML format files. The virtual world is represented using GLScene components, these provide a simple implementation of OpenGL. The NXT robot was defined using boxes and cilynders as shown in Figure 10. These are the objects used by the ODE to determine the collisions and friction. SimTwo allows the replacement of the solids by any 3DS model which gives a realistic look of the robot as shown in Figure 11.

4. SimTwo Application Examples

The presented examples are in the field of edutainment robotics. Edutainment robotics plays an important role in education due to the inherent multi-disciplinary concepts that are involved, motivating students to technological areas. It also plays an important role in research and development, because it is expected that the outcomes that will emerge here, will later be transfered to other application areas, such as service robots and manufacturing.

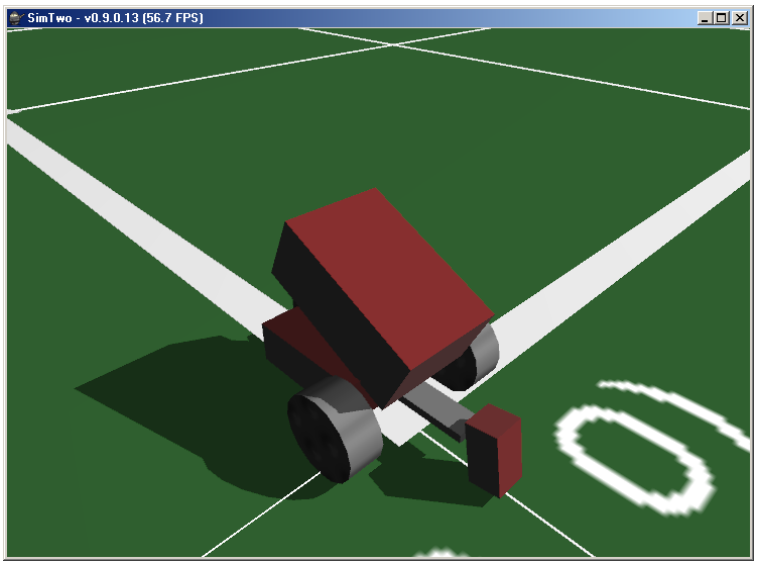

Figure 10. NXT robot built in the SimTwo

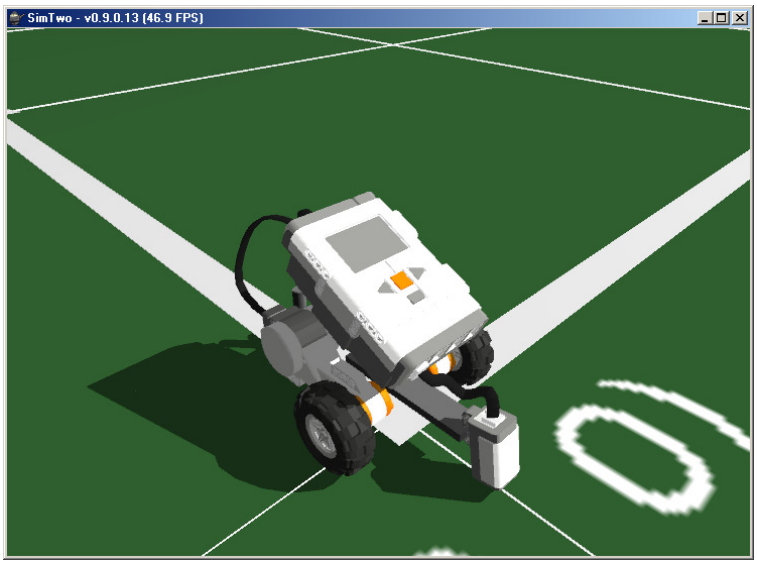

Figure 11. NXT robot built in the SimTwo with 3DS models

4.1. Simulation of an omnidirectional wheeled robot

This section presents the modeling and simulation of a wheeled mobile robot designed to operate in structured environments and how to develop robot software based on its simulation. It is presented the localization and navigation in a Fire Fight Robot Contest arena of an omnidirectional mobile robot equipped with brushless motors and infra-red distance sensors (Williams *et al.*, 2002) (Leow *et al.*, 2002) (Loh *et al.*, 2003) (Muir & Neuman, 1987). The used control architecture is generic and can be applied to several control challenges other than mobile robotics. The robot controller was implemented resorting to the simulator, using a remote client that communicates with the simulator using the UDP protocol, as shown in Figure 12. It is also possible to control robots using the embedded script functionality. The developed code can be migrated to the real robot, making the simulation useful, as presented previously in (Gonçalves *et al.*, 2009*a*).

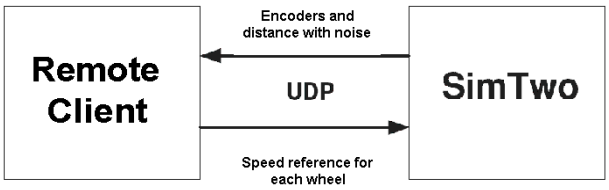

Figure 12. Simulator communicating with remote client.

The localization and navigation software is developed resorting to the simulation shown in (Figure 13). The simulator provides to a remote client the distance sensors data with noise as modeled in (Gonçalves *et al.*, 2008) and the encoders data. The remote client executes the localization and navigation algorithms and returns the speed references for each wheel to the simulator.

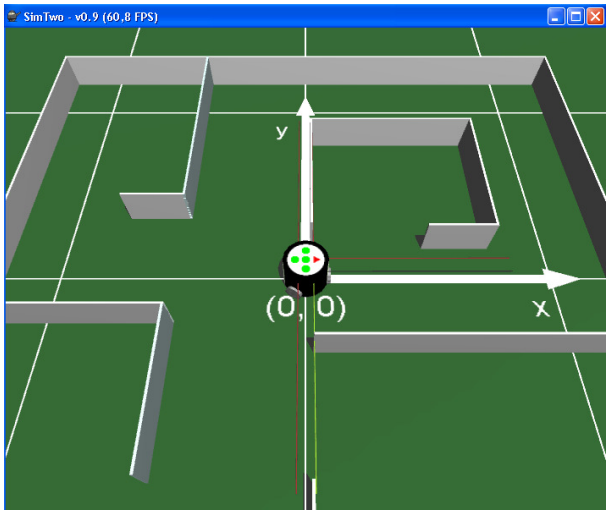

Figure 13. Robot simulator snapshot.

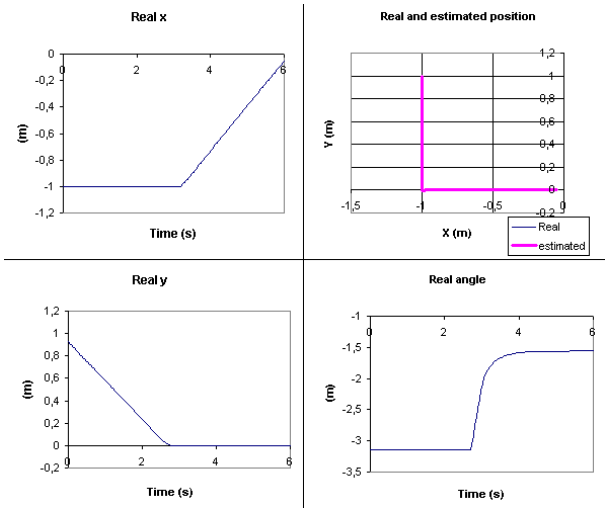

Figure 14. Robot trajectory.

Odometry and infra-red distance sensors data fusion was achieved applying an extended Kalman filter. This method was chosen because the robot motion equations are nonlinear and also because the measurements error probability distributions can be approximated to Gaussian distributions (Choset *et al.*, 2004) (Thrun *et al.*, 2005). As an example, a robot trajectory produced in the simulator is shown in Figure 14. The pose estimate error and its variance are shown in Figure 15.

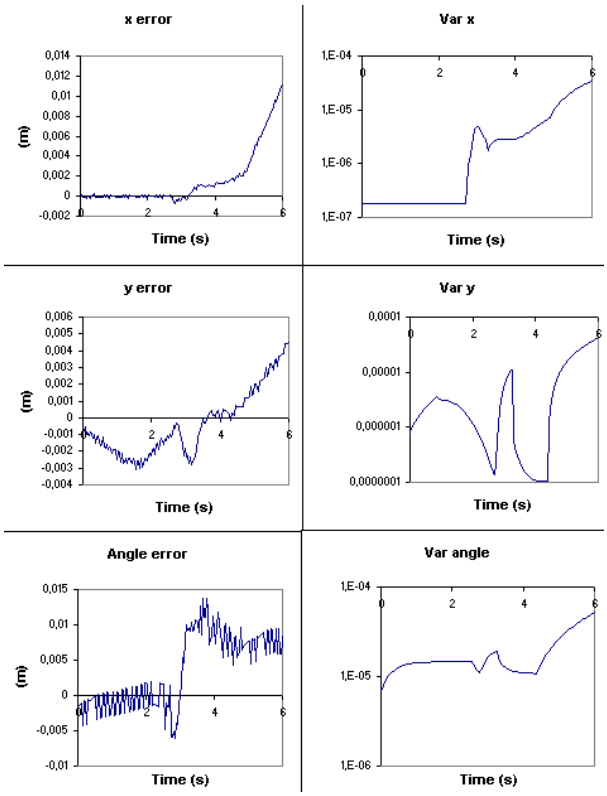

Figure 15. Pose estimate error and variance.

4.2. Simulation of a Humanoid Robot

In last years, research studies in biped robots evolved 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 environments. So, studies about biped robots are very important (Suzuki & Ohnishi, 2006) (Zagal *et al.*, 2009). A humanoid robot dynamic model is a complex mathematical matter, however, it can become easy if it is used a simulator.

Nowadays, there are several simulators that own the humanoid robot simulation capability. Meanwhile, this simulator allows to simulate different types of robots and allows the access to the low level behaviour, such as dynamical model, friction model and servomotor model in a way that can be mapped to the real robot, with a minimal overhead. The main purpose of Simtwo support humanoid simulations capability is to allow to develop new methods of control the robot (i.e. walk, get-up movements) and implement new issues that further can be applied to the real robot. There are several humanoid robots kits available. The commercially available Bioloid robot kit, from Robotis, served as the basis for the humanoid robot and the proposed biped robot is shown in Figure 16 and Figure 17 shows the humanoid robot in the Simtwo simulation environment. The presented humanoid robot is driven by 19 servo motors (AX-12): six per leg, three 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 on board computer) is about 2 kg and its height is 38 cm.

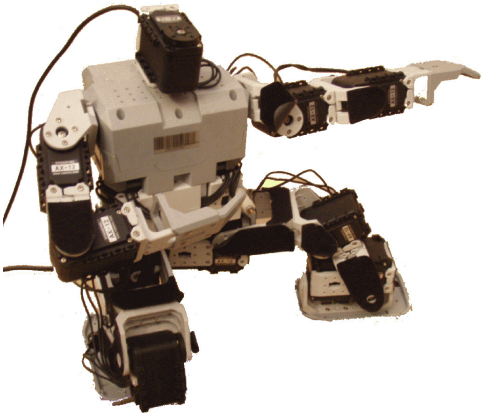

Figure 16. Real humanoid robot (Robotis).

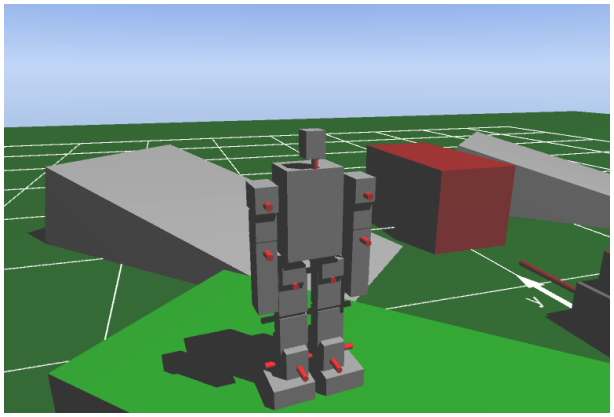

Figure 17. Simtwo Humanoid simulation.

The humanoid robot can be described through bodies and joints. Each body mass simulates 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. Joints types can be classified as hinges or universal joints: a hinge that allows both bodies to be connected and roll such as arms and forearms, femur and leg; a universal 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 allows two bodies to roll on both axes. As example, presented in the simulator, these joints connect trunk and arms, trunk and legs, legs and feet.

The humanoid servomotors were modeled and results presents a comparison between real and simulated servos. The friction constants can be found resorting to a free fall of an arm (Figure 19) and actuators constants (state-space controller gains) allows to obtain a comparison between real and simulated humanoid servos step response, as presented in Figure 20.

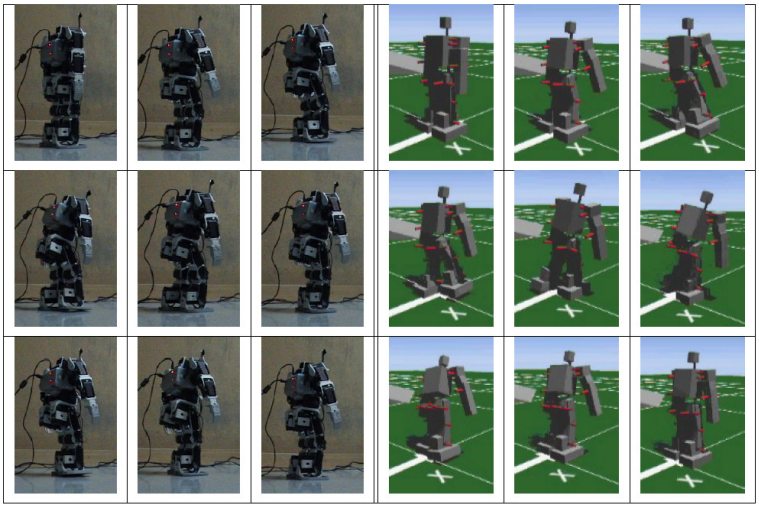

Figure 18. Real and Simtwo humanoid walk comparison.

A way to validate the humanoid simulation model is to apply the same control signal to both robots and to analyze the behavior. Predefined trajectory states, that allow robot to walk, are based on the Zero Moment Point (ZMP) method

and are well described in literature (Kajita *et al.*, 2006) (Zhang *et al.*, 2008) (Meghdari *et al.*, 2008). Figure 18 shows the sequence during walk movements for both robots (real at left and simulator at right).

It is possible to observe that both robots exhibit a very similar behavior. Previous works present the validation in other way: energy consumption on get-up movements (Lima *et al.*, 2008) and angles from real and simulator joints are compared (Lima *et al.*, 2009). These facts prove that humanoid simulator environment acts as a real robot.

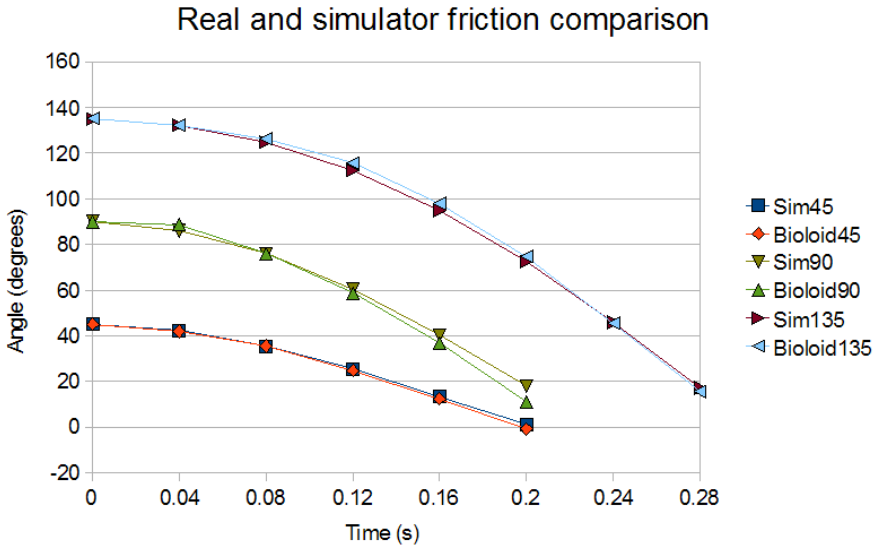

Figure 19. Real and simulator friction comparison.

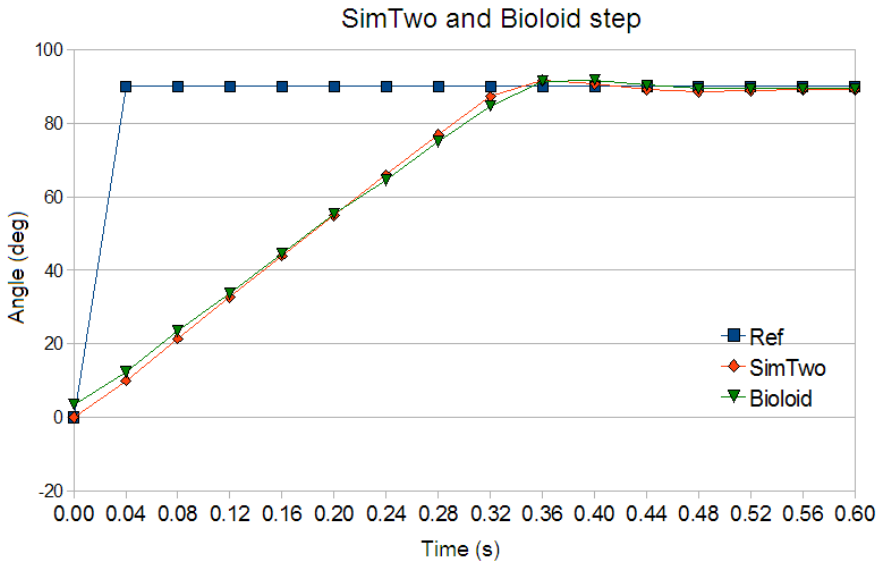

Figure 20. Simtwo and Bioloid step.

4.3. Simulation of a Lighter-than-air vehicle

One of the most interesting items in this simulator is an object that can elevate itself in the air. It is possible to create balloons and airships, as show in Figure 21.

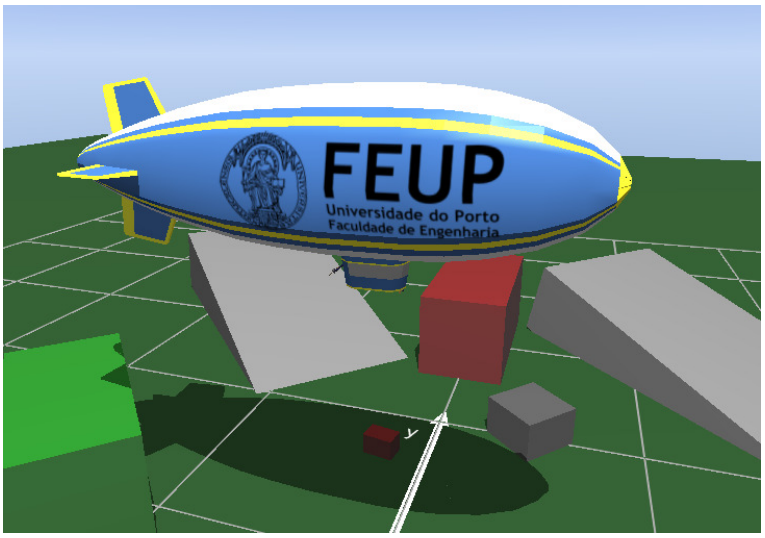

Figure 21. Blimp in SimTwo.

This type of vehicles is centered in one solid with a "buoyant" property, this is an upwards force that simulates the properties of the lifting gas. For more realism it is possible to define drag coefficient that is used to quantify the air resistance suffered by the vehicle. For movement this airborne vehicles require the use of propellers which was added to the SimTwo simulator.

This lighter-than-air application was specifically used for educational purposes in a microcontroller course unit. The students were required to model and control a real lighter-than-air vehicle which was a model of the famous "Goodyear Blimp", shown in Figure 22. A block diagram of educational experiment is shown in Figure 23.

Figure 22. Lighter-than-air vehicle class presentation.

A model was defined using a lift solid as the "balloon" and a second solid defining the control gondola in the correct size and position. The students were required to weigh these model elements for tuning the model and set the correct buoyancy parameter for air balance. Two propellers define the forward/backward movements as well as

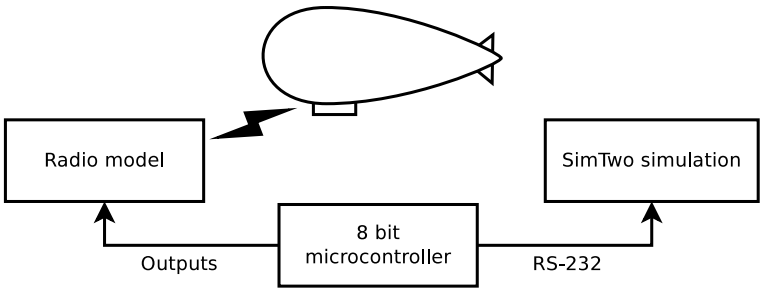

Figure 23. Zeppelin educational experiment.

steer the airship, a third propeller, placed horizontally below the gondola, allows the upwards/downwards lift. The horizontal and vertical speed can be defined in SimTwo tuning the motor parameters after the measurement of the same speeds in the real vehicle. A 3DS skin was added to make its appearance more realistic.

The main objective of the proposed educational experiment consisted in developing a 8 bit microcontroller application. The trajectory was defined by an open loop sequence of movements such as front, back, rotate left, rotate right, up and down which should be tested in SimTwo prior to using the real vehicle. The microcontroller sent the movement commands to SimTwo using a RS-232 serial connection thus validating the inputed trajectory. The same commands were sent to a radio module which relayed them to the airborne vehicle.

5. Conclusions and future work

The presented simulation environment has already been used on some educational settings where the students tested their control algorithms on a model before porting them to real robots. That avoided costly mistakes and increased the students exposure to the robots, even if some of that time was on a virtual robot. The SimTwo capabilities where also stressed on some research on humanoid robots that lead to a phd and some master thesis. That drove the development of SimTwo as the current research is still doing. While the initial focus was on mobile robots with omnidirectional wheels and humanoid robots, the capabilities where extended so that modeling robotic manipulators, car like robots, lighter than air vessels and machines with conveyor belts is already possible. There is an effort underway to implement more sensors with careful modeling from real ones. Also some physical devices like a mechanical differential are planned. The present library of already modeled robots is being expanded. While SimTwo is Wine friendly and able to run on a Linux x86 PC some effort is planned to make it truly cross platform.

References

Bishop, R. (2002). *The Mechatronics Handbook*. CRC Press, New York.

Campion, G., G. Bastin and B. Dandrea-Novel (1996). Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *IEEE Transactions on Robotics and Automation* **12**(1), 47–62. 1042-296X.

Choset, H., K. Lynch, W. Burgard, L. Kavraki and S. Thrun (2004). *Principles of robot motion*. MIT Press.

Gawthrop, P. and E. McGookin (2006). Using lego in control education. In: *Proceedings of the 7th Ifac Symposium on Advances in control education*.

GLScene (2010). http://glscene.sourceforge.net/wikka/HomePage.

Gonçalves, J., J. Lima, H. Oliveira and P. Costa (2008). Sensor and actuator modeling of an realistic wheeled mobile robot simulator. In: *Proceedings of the 13th IEEE International Conference on Emerging Technologies and Factory Automation*.

Gonçalves, J., J. Lima, P. Malheiros and P. Costa (2009*a*). Code migration from a realistic simulator to a real wheeled mobile robot. In: *9th Conference on Autonomous Robot Systems and Competitions, Castelo Branco, Portugal*.

Gonçalves, J., J. Lima, P. Malheiros and P. Costa (2009*b*). Realistic simulation of a lego mindstorms nxt based robot. In: *Scheduled for presentation during the Invited CCA Session "LEGO based Control Education and Prototyping in Robotics, Mechatronics and Embedded Systems, IEEE Multi-conference on Systems and Control", Saint Petersburg, Russia*.

Gonçalves, J., J. Lima, P. Malheiros and P. Costa (2010). Fostering advances in mechatronics and robotics resorting to simulation. In: *Proceedings of the 10th IFAC Workshop on Intelligent Manufacturing Systems, Lisabon*.

Hassanzadeh, I., H. Ghadiri and R. Dalayimilan (2008). Design and implemention of a simple fuzzy algorithm for obstacle avoidance navigation of a mobile robot in dynamic environment. In: *Proceeding of the 5th IEEE International Symposium on Mechatronics and its Applications (ISMA08)*.

Kajita, S., M. Morisawa, K. Harada, K. Kaneko, F. Kanehiro, K. Fujiwara and H. Hirukawa (2006). Biped walking pattern generator allowing auxiliary zmp control. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2994-2999, Beijing, China*.

Khosla, P. (1989). Categorization of parameters in the dynamic robot model. *IEEE Transactions on Robotics and Automation* **5**(3), 261–268. 1042-296X.

Leow, Y. P., Low K. H. and Loh W. K. (2002). Kinematic modelling and analysis of mobile robots with omni-directional wheels. In: *Seventh lnternational Conference on Control, Automation, Robotics And Vision (lCARCV'O2)*. Singapore.

Lima, J., J. Gonçalves, P. Costa and A. Moreira (2008). Realistic behaviour simulation of a humanoid robot. In: *8th Conference on Autonomous Robot Systems and Competitions, Aveiro, Portugal*.

Lima, J., J. Gonçalves, P. Costa and A. Moreira (2009). Humanoid realistic simulator: The servomotor joint modeling. In: *Proceedings of International Conference on Informatics in Control, Automation and Robotics, Milan, Italy*.

Loh, W. K., K. H. Low and Y. P. Leow (2003). Mechatronics design and kinematic modelling of a singularityless omni-directional wheeled mobile robot. In: *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*. Vol. 3. pp. 3237–3242.

Lund, H. and L. Pagliarinis (2000). Robocup jr. with lego mindstorms. In: *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, San Francisco, CA, IEEE*.

Meghdari, A., S. Sohrabpour, D. Naderi, S. Tamaddoni, F. Jafari and H. Salarieh (2008). A novel method of gait synthesis for bipedal fast locomotion. *Journal of Intelligent and Robotic Systems* **53**(2), 99–202.

Michel, O. (1996). *Khepera Simulator version 2.0*. User Manual.

Michel, O. (2004). Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems* **1**(1), 39–42.

Muir, P. and C. Neuman (1987). Kinematic modeling for feedback control of an omnidirectional wheeled mobile robot. In: *Proceedings 1987 IEEE International Conference on Robotics and Automation*. Vol. 4. pp. 1772–1778.

ODE (2010). http://www.ode.org/.

Olsen, M. M. and H. G. Petersen (2001). A new method for estimating parameters of a dynamic robot model. *IEEE Transactions on Robotics and Automation* **17**(1), 95–100. 1042-296X.

OpenGL (2010). http://www.opengl.org/.

Suzuki, T. and K. Ohnishi (2006). Trajectory planning of biped robot with two kinds of inverted pendulums.. In: *Proceedings of 12th International Power Electronics and Motion Control Conference, Portoroz, Slovenia*.

Thrun, S., W. Burgard and D. Fox (2005). *Probabilistic robotics*. MIT Press.

Williams, R., B. Carter, P. Gallina and G. Rosati (2002). Dynamic model with slip for wheeled omnidirectional robots. *IEEE Transactions on Robotics and Automation* **18**(3), 285–293. 1042-296X.

Wolfer, J. and H. Rababaah (2005). An integrated khepera and sumo-bot development environment for assembly language programming. In: *Proceeding of the International Conference on Engineering and Computer Education*.

Zagal, J., J.Delpiano and J. Solar (2009). Self-modeling in humanoid soccer robots. *Robotics and Autonomous Systems*.

Zhang, L., C. Zhou and R. Xiong (2008). A lie group formulation for realtime zmp detection using force/torque sensor. In: *Proceedings of the 11th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, pp. 1250-1257, Coimbra, Portugal*.