Two Humanoid Simulators: Comparison and Synthesis

Nima Shafii, Luis Paulo Reis, Rosaldo J. F. Rossetti
Artificial Intelligence and Computer Science Laboratory
Department of Informatics Engineering
Faculty of Engineering of the University of Porto - FEUP
Porto, Portugal
{*nima.shafii, lpreis, rossetti* }*@fe.up.pt*

Abstract— In this paper, an overview of two humanoid simulation platforms is provided: Simspark, a 3D Robocup simulator and the robotics simulator SimTwo. Although these two simulators have different background, today they share the same humanoid robot model, namely the Albaderan NAO robot. According to the fact that developing reliable and robust biped locomotion and low level humanoid behaviors are still challenging tasks, simulation has an important role in improving humanoids movement development approaches. In this paper the two humanoid robotic simulators will be compared in face of simulating Humanoid low level behaviors. The comparison is based on identifying the same role and locomotion approaches. The results show that Simspark is closer to reality than SimTwo.

Keywords-Bipedal Locomotion; Humanoids Simulation; Soccer Humanoid Robots.

I. Introduction

Humanoids robots try to mimic human-like behaviors and movements. While wheeled robot locomotion is not adapted to many human environments, such as stairs and areas littered by many obstacles, humanoid robots are able to avoid different shapes of obstacles, and attain postures that are more desirable. Therefore, by using biped locomotion, humanoid robots can function and perform their tasks easier than wheeled robots in areas designed for people. According to the fact that biped locomotion is similar to human movement, this similarity causes people to interact with humanoid robots easier as well. Although humanoid robots have lots of motivations behind them, the control of humanoid walking and running is still challenging and has not been solved [1]. Beside, biped locomotion is known as one of the most complicated robotics tasks in the field.

The idea behind a humanoid simulator is to develop a virtual agent capable of thinking and acting. Therefore, acquired knowledge from simulation can be transferred to real robots. Simulation is an easier way to develop biped locomotion methodologies. It makes performing gait optimization [2] and learning phase [3] easier and more efficient. It is also not reasonable to test the algorithm directly on the real humanoid robot since learning needs lots of iterations and humanoids always are very expensive and hard to troubleshoot in the case of falling or breaking. Therefore, Humanoid simulators must emulate the real world in the best possible way. To make this feasible, it is necessary to construct accurate and reliable models of the real robots. On the other hand, physical Simulators always contain some simplifications when it intended to model the real world, meaning the result of simulation and reality is usually not the same [4]. This problem is called the reality gap in the field of evolutionary robotics [5].

Researchers always try to find a simulator which has the least differences between its result and reality [6]. In this case of study, some researchers try to compare different simulators and find the differences between simulators in order to find exact causes of differences between them. It can also lead them to obtain the best model of a simulator that is closer to the reality. Performance of humanoids simulation depends on many factors related to modeling the physical environment and each simulator uses its specific architecture such as physics engine and methods for discretizing time. A clear approach to compare efficiency of simulating humanoid does not exist yet.

In 2009, Shivaram and Stone compared three humanoid robotics platforms [7]. They compared the simulators by analyzing results of robots walking with the same robots model and same approach of walking. For example, they compared the speed of walking in simulators with the speed of walking in reality, so as to conclude which simulator is closer to the real world.

In this paper we choose two different humanoid simulators, simTwo [8] and Simspark (RoboCup 3D league Simulator) [9], which simulate the same model of humanoid robot, NAO [10]. They will be overviewed, then; a methodology will be presented to compare them in order to guide us to find out the better simulator. By implementing the methodology on these simulators, we intend to generate some results that will allow us to carry out appropriate comparison. Finally, comparing these results can lead us to find the exact causes of differences to find out which of these simulators is better to use as a humanoid test bed.

The rest of the paper is organized as follows. Section 2 presents an introduction to the humanoid robot model and platform. Section 3 and 4 present respectively the RoboCup 3d league simulator and the SimTwo simulator. Section 5 presents our methodology for comparing the simulators. Section 6 presents the implementation and results. The paper concludes with some final remarks and pointers for future work.

II. Humanoid Robot Model and Platform

A humanoid robot is a robot with its overall appearance based on a human body and that is able to stand and move on its own two legs. The number of their joint actuators indicates the number of Degree of Freedom (DOF). Like humans, humanoid's body moves in three planes, including transverse (axial), frontal (coronal) and sagittal. Sagittal plane indicates the vertical plane running from front to back and divides the body into left and right sides. Frontal or Coronal plane is a plane perpendicular to the sagittal plane, running from side to side and dividing the body into front and back.

In the NAO robot model, 12 DOFs are embedded in its legs. There are three DOFs in each legs moving in the sagittal plane: one in the hip, one in the ankle and one at the knee. There is also one actuator in the hip, which can turn and move the leg in transverse plane. In addition to DOFs of legs, the model has 10 DOFs in the upper parts of its body. Fig. 1 illustrates a schematic view of a humanoid robot. As it was described, DOFs 1, 3, and 4 move on Sagittal plane, DOFs 2 and 5 move on Frontal plane and DOF 6 moves on transverse plane.

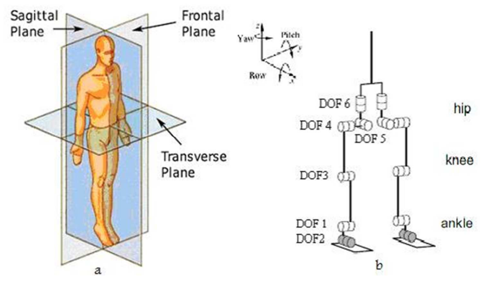


Figure 1. A. Human body planes, B. NAO joint's configuration

III. Robocup 3D League Simulator

The Robocup Simulator is a generic simulation platform for physical multi-agent simulations and it is used in the RoboCup 3D simulation league currently. This simulator has been developed by the RoboCup community since 2006. It is designed as a flexible application framework and intends to be a generic simulator, capable of simulating anything, from the launch of a projectile for academic purposes to a big soccer game for scientific research purposes. The framework facilitates exchanging single modules and extending the simulator [11]. The Simulator consists of two important parts: Server, and Monitor.

A. Server

The server is responsible to handle connections from the agents; to receive and to process messages, send reply messages to the agents. The server architecture is illustrated in Fig. 2.

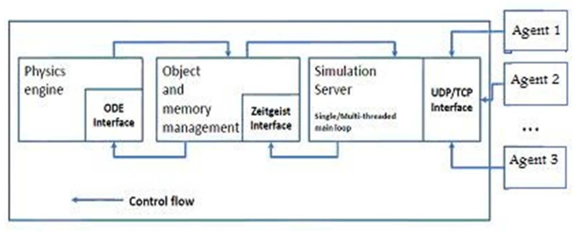


Figure 2. Robocup Simulator Server Architecture [11]

The simulator uses the Open Dynamics Engine (ODE) [12] to simulate the physical environment. ODE is a physical simulation engine, which allows simulating the system's dynamics and the physical properties of the simulated objects. It provides advanced joint types and integrated collision detection with friction. ODE is particularly useful for simulating objects in virtual reality environments. It is cross-platform and provides a user-friendly C/C++ Application Programming Interface (API).

RoboCup simulator is very well-known because it can be seen as a multi-agent environment enabling different types of experiments. Several agents can connect to the server simultaneously. Its object and memory management is based on Zeitgeist [11]. It is a framework for handling data objects and functional components of a system in a uniform way, which strictly follows the object-oriented paradigm using C++ programming language.

The core of the simulator is a simulation server. It receives the messages with actions from the agents, performs the simulation operations and sends a reply message back to the agent with the environment information. The simulation engine acts as a server, handling the messages of the agents and replying on to other messages. In Each cycle of the simulation, agents send a message to the server containing information about their effectors (e.g. joints). The message from the server to the agent contains temporal information and specific information from the application domain (which is soccer). This information includes game state (play mode, time and current result), and information of the preceptors of the robot (e.g. joints, gyroscopes, foot sensors, vision information). The messages are constructed using a LISP-like format.

B. Monitor

The monitor provides a simple graphical interface that allows the user to watch a simulation. Simulations may be watched in real-time, but it is also possible to play simulation log files offline, for post processing assesment. In the particular case of humanoid soccer simulation, it provides additional information such as the team names and the game time, play mode and results. Several shortcut keys can be used to change camera views, to drop the ball, and carry out other useful operations.

IV. SimTwo Architecture

In 2006, SimTwo [8] was developed by Dr. Paulo Costa in the University of Porto. It focused on preparing a simulated environment to test and develop mobile robot approaches and applications.

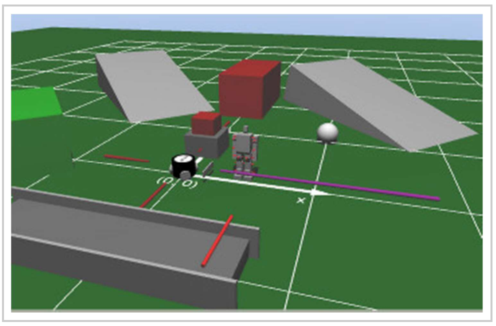


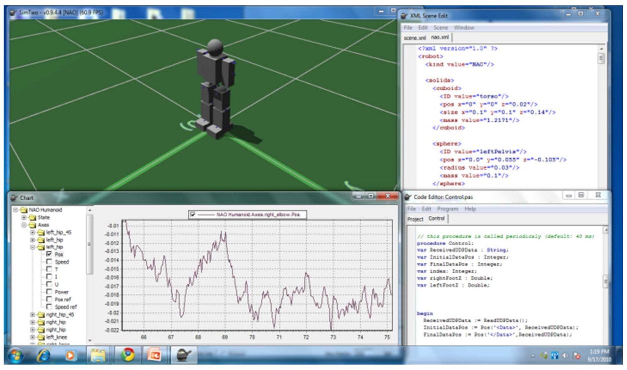
Figure 3. Snapshot from SimTwo environment

The SimTwo system is a realistic simulation where anyone can implement various types of mobile robots with different configurations including the different body mass and shape and any types of joints, which can be described with a mixture of classic joints and wheels. Therefore, in addition to simulate wheeled robots, humanoid robots can also be simulated. Recently, the model of NAO robots was also added to this simulator. In Fig. 3, the environment of SimTwo is shown

The realism of the dynamics implemented in SimTwo is achieved by dividing a robot in to a system of rigid bodies and electric motors. The "mechanics" associated with bodies is numerically simulated considering their physical shape, and mass moments of inertia, friction and elasticity of the surfaces. Like the Robocup simulator the Physical simulation is performed by using ODE, which was described in the previous section. Certain joints are designed as shaft gimbals typically and as pipeline explicitly. Each of them can have its associated drive system and sensors.

The drive system may consist of a DC motor and gearbox. The DC motor model contains several elements such as nonlinear saturation of the applied voltage, current limit and friction. To define the environmental configurations in the simulator, SimTwo can design any types of obstacles which has different shape and role such as slops, stairs, cubes and so on. Obstacles and robots can also be moved easily during the simulation by dragging mouse on them.

To define and implement the environmental configuration and robot's model, SimTwo has different tools, which allows developers to import their configuration as XML scripts. In order to show the results of the simulation, a graphical viewer based on OPEN-GL is also implemented. A debugger tool is also developed as a graphical viewer, which can draw any graphs and trajectories. In Figure bellow tool-boxes of the SimTwo can be found. The two windows on the right.show the XML-based environment as well as the visual viewer and debugger on the left.



```
<?xml version="1.0" ?>
<robot>
  <kind value="NAO"/>

  <solids>
    <cuboid>
      <ID value="torso"/>
      <pos x="0" y="0" z="0.02"/>
      <size x="0.1" y="0.1" z="0.14"/>
      <mass value="1.2171"/>
    </cuboid>

    <sphere>
      <ID value="leftPelvis"/>
      <pos x="0.0" y="0.055" z="-0.105"/>
      <radius value="0.03"/>
      <mass value="0.1"/>
    </sphere>
```

```
// this procedure is called periodicaly (default: 40 ms)
procedure Control;
var ReceivedUDPData : String;
var InitialDataPos : Integer;
var FinalDataPos : Integer;
var index: Integer;
var rightFootZ : Double;
var leftFootZ : Double;



begin
  ReceivedUDPData := ReadUDPData();
  InitialDataPos := Pos('<Data>', ReceivedUDPData);
  FinalDataPos := Pos('</Data>',ReceivedUDPData);
```

Figure 4. View of SimTwo tool boxes

SimTwo has good capabilities to simulate actuators and to define obstacles and environment features, but it does not have adequate architecture to manage several agents to connect to the simulator as a multi-agent environment.

V. METHODOLOGY TO COMPARE THE SIMULATORS

Such as in [7], our comparison approach is based on a benchmark method. In this method, results and comparisons of running a computer program are achieved by running a number of standard tests and trials against it, in order to assess the relative performance of an object. To choose our standard tests, we run the same humanoid scenarios in different simulators. Therefore, a humanoid scenario must have the same dependencies and characteristics, including:

- Same approach for modeling agents' behavior cycle;
- Same approach for controlling robots;
- Same machine specification to run the simulation.

In this paper two humanoid scenarios will be proposed, which are called Vibrating and Walking. Following the characteristics and dependencies of these scenarios will be explained.

A. *Modeling of Agent's Behavior Cycle*

Performing agent's tasks and scenarios are related to agent's behavior architecture, which is implemented inside the agent. This architecture must be defined as a cycle, since agents must receive sensor data from the simulator, then process it and try to make decision based on it and after that, send an action back to the simulator. The figure below shows the architecture of our agent behavior cycle.

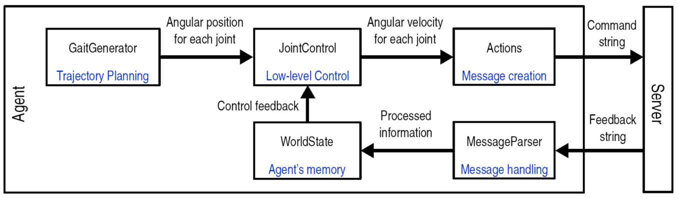


Figure 5. Agent behavior cycle architecture

Sensors from the simulator are received as a string. So, a message parser unit is needed to interpret it and prepare needed information for producing world state. According to the fact that our agent will be implemented in two different simulators, therefore the sensor's string and message parser unit will be different. Because our comparison needs to use the same decision-making structure, it is very important to prepare the same information from sensory input. World state unit processes all information from the world which is needed for agents to perform its decision. In different simulators, the world state of the agent must also be the same. For example coordinate system of joint's angle is not the same in different simulators, therefore these inherent differences must be solved and considered by the world state unit.

In this study, humanoids low-level movements have been chosen as our scenarios, we divided decision-making unit in two separate units: gait generator and joint control. Following related to each scenarios, the model of these two units will be explained.

B. *Modeling of robots walking (First Scenario)*

Based on the work referenced in [7], the main and first scenario which is presented for comparison between simulators is walking, since walking uses lots of actuators and it is known as a very complex motion. According to the fact that all of the physical futures can influence the performance of its simulation directly, it is very hard to simulate biped locomotion, especially some features like friction between surface and robots.

A Biped locomotion approach based on human motion captured data and Truncated Fourier Series is used in this study. In 2007 Truncated Fourier Series Formulation (TFS) method was used as a gait generator in bipedal locomotion [13]. Recently, an optimized gait generator based on TFS was implemented on a simulated humanoid robot and TFS less parameters were also reduced by 2 dimensions (down to 6 dimensions) [14] [15]. According to the fact that this model had least parameters and more simple approach compare to other newer approaches, it is considered for the first scenario. In the following its detail will be explained more.

In this approach, movements of three DOFs in each leg, which are moved in sagittal plane, will be calculated. Foot in sagittal plane was also kept parallel to the ground by using ankle joint in order to avoid collision. Therefore ankle trajectory could be calculated by hip and knee trajectories and ankle DOF parameters were eliminated.

In this model, legs joint angular trajectories in sagittal plane are divided in two parts; the upper portion and the lower portion. The TFS for generating each portion of hip and knee trajectories are formulated below.

$$\begin{aligned}
\theta_h^- &= \sum_{i=1}^{n} B_i.\sin(iw_h t) + c_h, w_h = \frac{2\pi}{T_h} \\
\theta_h^+ &= \sum_{i=1}^{n} A_i.\sin(iw_h t) + c_h, w_h = \frac{2\pi}{T_h} \\
\theta_k^+ &= \sum_{i=1}^{n} C_i.\sin(iw_k t) + c_k, w_k = w_h \\
\theta_k^- &= c_k \geq 0
\end{aligned} \tag{1}$$

In these equations, C_h is offset of hip trajectory and C_k is offset of knee trajectory. The plus (+) sign represents the upper portion of walking trajectory and the minus (-) shows the lower portion. $i=1$ and A_i, B_i, C_i are constant coefficients for generating signals. The h and k index stands for hip and knee respectively. C_h, C_k are signal offsets and T_h is assumed as a period of hip trajectory. Considering the fact that all joints in walking motion have equal movement frequency and stride rates is statistically equal, the equation $w_k = w_h = \frac{2\pi}{T}$ can be concluded. While Left leg is considered as supporting leg, the variation of its knee angle is so minute that can be assumed fixed, This duration of walking is named knee lock phase and $\theta_k^- = c_k \geq 0$.

The amount of shift phase of the two leg trajectories signal is one half of the period of each signal so by producing trajectory of one leg the other leg's trajectory can be calculated. The trajectories for both legs are identical in shape but are shifted in time relative to each other by half of the walking period.

The walking scenario must be implemented as the gait generator unit, in the same way. For generating proper walking trajectory, a robot must learn and find the best value of the parameters which leads a robot to walk more stably and robustly. According to the fact that gait generator will produce angular trajectories, Control unit must control the joint actuators along their references. In the next section we will introduce our optimization method which is used in this project.

C. Modeling of Vibration skill (second scenario)

In this scenario, in order to detect differences of actuators simualtion, we have chosen the elbow joint, which has the least friction with surface. Therefore, environmental factors have less influent on simulating its movements. After the, in order to find the differences better, a scenario with the biggest change of power of elbow's actuator is presented.

The algorithm of the scenario will be presented below. It commands a robot to move its elbow to reach 50 degree and command to go to the opposite side, -50 degree in the next simulation step and after that repeats the scenario again. Due to a quanta of simulation step is reasonably small, 0.04 and 0.02 second in SimTwo and Simspark respectively, to reach to this angle a robot must utilize the highest power of its elbow's actuator.

```
if changeStep=true then
  begin
        SetAxisPosRef (0, GetAxisIndex
              (0,'right_elbow', 0),-50);
        changeStep:=false;
  end
  else begin
        SetAxisPosRef (0, GetAxisIndex
              (0,'right_elbow', 0),50);
        changeStep:=true;
  end;
```

This scenario will be implemented as the trajectory generator unit of the second scenario with the same configuration on both simulators. As a controlling unit, a PD controller with the same specification will be used.

VI. Implementation and Results

To compare SimTwo and Simspark both of them were tested on machines which have the same machine specification. The specification of machines are Pentium IV 3 GHz Core 2-Duo with 2 GB of physical memory. Scenarios considering the same specification are implemented on the simulators. As it was mentioned before, joints angular trajectories produced by each scenario are used by a simulated robot to test it. In order to utilize them, all individual robot's joints should attempt to drive towards their target angles using proportional derivative (PD) controllers. Same configuration of the PD must be applied for each different simulator, In this project, the P factor was assumed to be 200 and the D was assumed to be 2. In the following, implementation result of two scenarios in different approaches will be discussed.

A. walking (First Scenario)

According to first scenario which was shown in the previous section, the best parameters to generate joints angular trajectories for bipedal locomotion must be found. According to [14], for this kind of optimization problem, Genetic Algorithms (GA) can be used. Therefore on RobocupSim, first Robots try to find the best Parameters by using GA. Then results of it will be used for the walking robot which is simulated on the SimTwo. By extracting the relation between motor's command and Joint's angle, and comparing speeds of walking and joints angular trajectories extracted from simulation, comparison can be done.

1) *Simspark Results*

For using GA as an optimizer in Simspark, the cross over rate and mutation rate are set to 0.8 and 0.06, respectively. Population for each generation is 100 and selection method is roulette wheel. Termination condition is to have a generation counter greater than 28. Therefore the GA requires 2800 trials to find appropriate TFS parameters.

After 9 hours, since starting GA on the machine, generation exceeded to 28 and the robot could walk 34.5m in 77 s with average body speed of around 0.45 m/s and time period of each step was about 0.41s. GA led the robot to learn how to walk straight, Fig. 6 exhibit biped locomotion is obtained from GA search and in Fig. 7 angular trajectory generated by TFS after learning process is shown. Followed trajectories based on the learned trajectory is also shown in this figure.

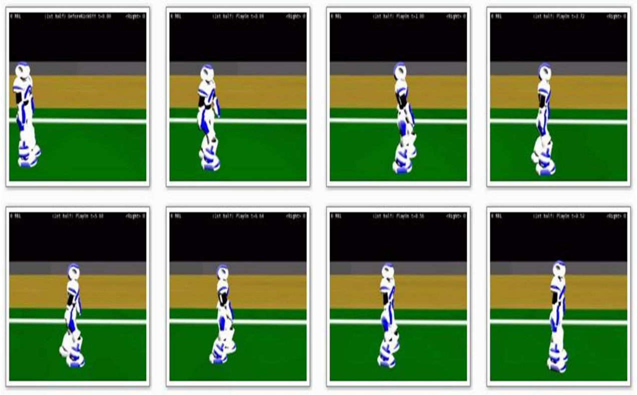


Figure 6. Walking is obtained by GA on the Simspark

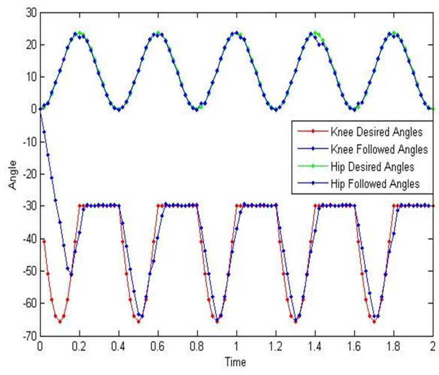


Figure 7. Angular Trajectory generated by learned TFS (Desired angle) and followed one by controller (Followed angle) for left hip and left knee of simulated Nao robot

2) *SimTwo Results*

The values of the walking parameters achieved by RobocupSim are used for the walking scenario, which is simulated in SimTwo. According to the fact that the Humanoid in SimTwo uses the same controller's configuration and gait generator model, therefore the robot's model must produce same walking trajectories and the results of the walking must be the same in both simulators. But, after implementing first scenario on SimTwo, the humanoid could not walk. This result shows differences between these two simulators. Fig. 9 shows the humanoid robot felt down during its walking in the SimTwo environment. As an additional test, we allowed the humanoid robot to walk above the ground. This scenario can allow a robot to test its walking without environmental friction and ground force. Fig 9 also shows the robot which is walking above the ground. Results were very interesting since showing that the robot can generate and follow the exact joints angular trajectories, which were produced by the model on Simspark. Fig. 10 shows the angular trajectory of the left hip which is performed by SimTwo when robot is walking scenario above the ground.

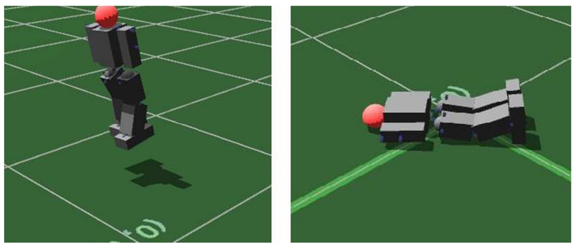


Figure 8. B) Robot walking above the ground **A)** Robot felt in during walking

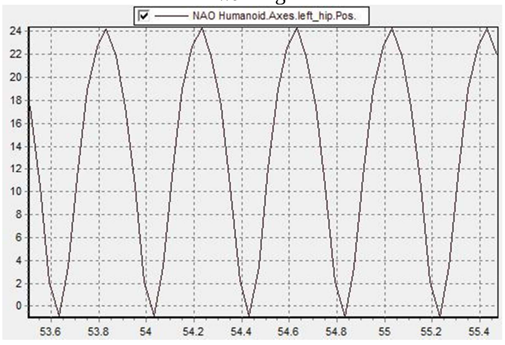


Figure 9. Hip Angular Trajectories, generated by walking above the ground

In conclusion, the results of testing first scenario on both simulators were different. It has also proved that different approaches are used for modeling and simulating actuators especially in the case of facing surface friction and ground forces, because without these environmental elements, generated angular trajectories were the same in both.

B. *Vibration Scenario*

According to the fact that both simulators achieved different results in the manner of simulating actuators, vibration scenario was very useful in order to study the differences between simulators. It uses the biggest change of actuator each time. It has been implemented on both humanoid simulators with the same controller unit characteristics. In SimTwo, simulation results are not very realistic, since it shows the elbow actuator is very powerful an it can move the elbow to reach 10 degree in duration of one step and in the next simulation step (0.04 s) it can reach -10 degree. Therefore, in this situation the simulated actuator of elbow can create a big torque, which can rotate the elbow 500 degree/sec. It means that the simulated actuator in SimTwo needs to use high power or voltage to produce this huge torque. Therefore, it seems not to be a real simulation since NAO robot has a small lithium battery to prepare this high power. Fig. 10 shows a snapshot from SimTwo while robot is executing the second scenario. Fig. 11 also shows the angular trajectory of elbow simulated on SimTwo.

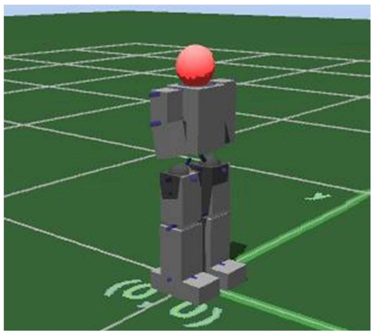


Figure 10. NAO executes the Vibration on SimTwo. It is also shown the elbow moves with big variations.

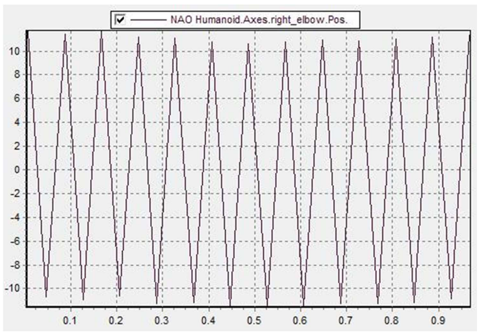


Figure 11. Angular trajectory of elbow with in duration of 1 sec. simulated in SimTwo

However, variation of degree on SimTwo was very big, but implementation of the vibration scenario on Simspark shows the angle of elbow changes varied less than four degree for each step. It means that its actuator can produce power for moving the elbow with the maximum speed of 100 degree/second and it is not utilizing the big power. Therefore, simulated actuator by Simspark is closer to reality than SimTwo. However in the time that this paper has been written, group that developed the SimTwo project was working on it and already achieved better simulation results. So, it is expected that the new SimTwo version may achieve results similar to those achieved today by the RoboCup simulator.

VII Conclusion

In this paper, the RoboCup 3D league Simulator and SimTwo were reviewed concerning their capabilities to simulate humanoid robots. In order to compare them, two different scenario based on benchmark approaches were presented. After implementing these scenarios, the results showed differences in their behaviors mostly in the case of simulating actuators. Analyzing the results led us to conclude that the 3D league RoboCup simulator is closer to reality than SimTwo.

According to the fact that SimTwo is very interesting in what concerns its tools and global facilities, turning it into a more realistic simulator may be a key point for its more global use. In addition, it can be interesting to use the results of these two simulation studies on a real robot to find out which of them is more near to reality through a new set of experiments.

Acknowledgements

The first author is supported by FCT under grant SFRH/BD/66597/2009. This work has been partially funded by FCT Project ACORD - Adaptative Coordination of Robotic Teams (PTDC/EIA/70695/2006).

References

[1] E. Westervelt, “Ask The Experts: Biped Walking,” IEEE Control System Magazine, pp. 20 -23, 2003.

[2] J. E. A. Bertram, “Constrained optimization in human walking: cost minimization and gait plasticity,” The Journal of Experimental Biology, Vol. 208, pp. 979-991, 2005.

[3] K. Wolff, J. Pettersson, A. Heralic, and M. Wahde, “Structural evolution of central pattern generators for bipedal walking in 3d simulation,” Proceedings of the 2006 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2006), pp. 227–234, 2006.

[4] J. C. Zagal, J. Ruiz, “Combining Simulation and Reality in Evolutionary Robotics,” Journal of Intelligent and Robotic Systems, Vol.50(1), pp. 19-39, 2007.

[5] S. Koos, J.-B. Mouret, “Crossing the reality gap in evolutionary robotics by promoting transferable controllers,” Proceedings of the 12th annual conference on Genetic and evolutionary computation, pp. 119-126, 2010.

[6] T. Laue, M. Hebbel, “Automatic Parameter Optimization for a Dynamic Robot Simulation,” L. Iocchi et al. (Eds.), RoboCup 2008, LNAI 5399, pp. 121–132, 2009.

[7] S. Kalyanakrishnan, T. Hester, M. Quinlan, Y. Bentor ,P. Stone, “Three Humanoid Soccer Platforms: Comparison and Synthesis,” Proceedings of the RoboCup International Symposium, LNCS/LNAI Springer, pp.140-152, 2009.

[8] J. Boedecker and M. Asada, “SimSpark – Concepts and Application in the RoboCup 3D Soccer Simulation League,” Autonomous Robots, 2008, pp. 174-181.

[9] J. Boedecker, “Humanoid Robot Simulation and Walking Behaviour Development in the Spark Simulator Framework,” Artificial Intelligence Research University of Koblenz, 2005.

[10] “NAO Albedaran,” available at: http://www.aldebaran-robotics.com/en

[11] O. Obst, M. Rollmann, “Spark - a generic simulator for physical multi-agent simulations.” In: LNCS. V. 3187, G. Lindemann, I. J. Timm, R. Unland, 2004, Springer.

[12] R. Smith, ODE _ Open dynamics engine, www.ode.org, 2008.

[13] L. Yang, C. M. Chew, A.N. Poo, “Adjustable Bipedal Gait Generation using Genetic Algorithm Optimized fourier Series Furmulation,” In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4435-4440, 2006.

[14] N. Shafii, M.H. Javadi, B. Kimiaghalam, “A Truncated Fourier Series with Genetic Algorithm for the control of Biped Locomotion,” In Proceeding of the 2009 IEEE/ASME International Conference on advanced intelligent Mechatronics, pp. 1781-1785, 2009.

[15] N. Shafii, L.P. Reis, and N. Lau, “Biped Walking using Coronal and Sagittal Movements based on Truncated Fourier Series,” RoboCup 2010: Robot Soccer World Cup XIV, Javier Ruiz-del-Solar, eds., 2011, pp. 324-335.