International Journal of Humanoid Robotics
© World Scientific Publishing Company

3D simulations of humanoid soccer robots

EMANUELE MENEGATTI, GIOVANNI SILVESTRI, ENRICO PAGELLO

Department of Information Engineering,
University of Padua, Via Gradenigo 6/a
Padova, 36136, Italy
emg@dei.unipd.it

NICOLA GREGGIO

Scuola Superiore Sant'Anna, ARTS-Lab,
Pontedera (Pisa), 56025, Italy
nicola.greggio@arts.sssup.it

ANTONIO CISTERNINO, FEDERICO MAZZANTI

Group, Laboratory, Address
City, State ZIP/Zone, Country
second_author@group.com

ROSARIO SORBELLO, ANTONIO CHELLA

Group, Laboratory, Address
City, State ZIP/Zone, Country
rosario.sorbello@group.com, antonio.chella@group.com

Received Day Month Year
Revised Day Month Year
Accepted Day Month Year

Keywords: Humanoid Robotics; Simulation; Realistic Physical Properties; Kinematics.

1. Introduction

The simulation of real robots is one of the fundamental research areas in robotics for its many applications. Indeed, a realistic simulation allows researchers to develop and test programs in a virtual environment, without the physical presence of the robot. This may be extremely useful, especially, in situations in which there are many people working on different aspects of the same robot, like in the case of humanoid robots. In this situation, each person can work independently without interfering with the others. Moreover, simulators are very useful in education: Students can develop different motions and behaviors for the robot, without been physically in the laboratory or without the risk to damage the robot. Again, this is particularly true for humanoid robots which are complex units built with expensive electronic and mechanical parts.

However, create a realistic simulation of a humanoid robot is a challenging task due to the complex structure of the kinematic chain created by the robot limbs and body and due to the complex dynamics of robots with several degrees of freedom (DOF). In fact, even the simplest humanoid robots have easily more than 15 DOG. An additional complication is that usually humanoid robots have different kinds of actuators for the different joints. This requires in the simulator the need to specify different kind of joints or hinges and different parameters for the actuators controlling them (e.g. different torque or friction coefficients, for the different joints, like the elbow or the knee of the robot). Most of the large humanoid robot projects that involve big and expensive humanoid robot platforms have a dedicated simulator to simulate just that specific humanoid robot. Some examples are:

However, if one is interested in multi-robot humanoid teams, as in the soccer game, this kind of simulator is not appropriate. The simulation of multi-robot humanoid teams requires much more flexible simulators with the capability to simulate different robotic platforms that can interact each other. Moreover, the simulation of a soccer match between two teams of soccer robots requires to simulate several robots at the same time, in which the robots of the two teams are probably different (in terms of bodies and actuators, and might be also heterogeneous within the same team) and physically interact among them, with the ball, and with the other objects in the field. So far, most of the research on multi-robot systems used 2D simulators, just to cite few of them:

However, for simulating a multi-robot humanoid team the 2D simulation is not enough. It can be enough for wheeled robot, but robots with many degrees of freedom cannot properly simulated in a 2D world. Legged robots, and humanoid robots in particular, need a more complex simulator creating a faithfully 3D simulation. This was correctly understood several years ago by the RoboCup Simulation community that understood that to contribute to achieve the 2050 goal of beating a human soccer team with a robot soccer team they had to move from a 2D simulation XXX to a 3D simulation.

We are interested in simulators providing, not only realistic physical simulations, but also realistic rendering of the 3D scenes. This is because we are interested in simulating robots whose main sensor is a camera. If the simulator can provide a realistic rendering of the 3D scene and if it is possible to generate what is called an egocentric view of the scene, i.e. to render the scene as viewed by a camera mounted on the robot, it is possible to "close the loop"of the robot control directly in the simulation. In fact, it is possible not only to view the virtual robot moving in the virtual world, but also to run the image processing algorithm designed for the robot on the synthetic images generated by the virtual camera and to use the result of the image processing as input for the robot behavior control module in order to select the most appropriate command to be sent to the virtual robot.

Another feature we are interested in is the possibility to control the simulated robot in the virtual world exactly with the same code used to control the real robot in the real world. Additionally, in order to validate the simulation and the robot model, we would like to control both robots (i.e. the virtual and the real one) at the same time.

There are a fair number of 3D simulators which fulfill these requirements, and each of them presents advantages and disadvantages, in the following we report some of them. The ASURA RoboCup Software by the Asura RoboCup team [?] has been developed for the simulation of four legged robots. It allows to use strategies and sensors acquisition, but it lack in representing dynamic simulation, so giving a poor representation of the virtual environment. SimRobot simulator by Laue *et al.* [?], supports different models with a great flexibility of control and different body models, sensors, and actuators are available. Dynamics is simulated with the open-source physical engine Open Dynamic Engine (ODE) [?]. Also, UCHILSIM by Zagal and Ruiz-del-Solar [?] uses the ODE engine for the physical simulation. This project has been developed to became a standard framework for the AIBO robot simulation. A simulator able to simulate both AIBOs and humanoid robots is `MuRoSimF` [?]. `MuRoSimF` is very interesting for soccer matches simulations mainly for two reasons: it provides single robot sensors simulation (in order to close the control loop in simulation) and it provides a scalable level of detail and complexity of the kinematics and dynamics which can be tailored to the requirements of a specific simulation (even chosen individually for each simulated robot). Thus, the user can trade off between available computational resources and simulation fidelity. However, so far this software is not available and the rendering of the simulated camera is not very realistic. Webots [?] is a commercial software for robotic simulation. It uses the ODE engine. It has an extensive library of actuators, sensors and robots. In addition, the mechanical features of the robots are well defined and the available libraries allow to control the virtual robot and the real robot with the same code. However, it lacks in the quality of the 3D graphical representation of the virtual environment and being a commercial robotic simulator its cost is not affordable by everyone. Another popular simulator in the robotics community is the toolkit Player - Stage - Gazebo. Player is the server coordinating the simulations, Stage is the 2D rendering client and Gazebo is the 3D rendering client. Player/Stage is well suited for 2D simulation and several groups are using it around the world for research and education. However, the 3D rendering is quite simple and does not have good rendering performance. In particular, the creation of the virtual scene is quite difficult and the 3D rendering is not realistic.

Our final choice was for USARSim (Urban Search And Rescue Simulator) [?]. USARSim provides a great fidelity in visual 3D rendering and realistic simulation of physics in the virtual environment exploiting the rendering and the physical engines of a commercial computer game called Unreal Turnment. In addition, we choose USARSim because there is a large open-source community that is developing and maintaining it, especially people from the RoboCup Rescue community [?], [?], [?], [?], [?]. So far, USARSim has been used to simulate wheeled or tracked robots, the only legged robots available were AIBO robots and a non realistic model of the

Sony QRIO[?]. To the best of our knowledge the model we created in USARSim was the first model of a commercially available humanoid robot (nowadays, other models are spawning like the Robonova model presented in [?]). While working on this project Microsoft released its development suite for robotics called *Robotics Studio*. Robotics Studio provides an integrated software development kit for programming real robots and it provides a simulator to test the software before mounting it on a real robot. Considering the major role of Microsoft in the computer world and the huge effort they are putting in this product in term of software development, promotion in the robotics community and even commercial advertising, we think Robotics Studio will have a strong impact on the robotics community, even because it is free of cost for reasearch and educational purposes. Therefore, we think it is mandatory to compare Robotics Studio simulator with the best simulators already there.

In this paper, we describe how we created the model of a commercial humanoid robot platform called Robovie-M by VStone ltd. for two general purpose robotic simulators which are easy available for the research communit: USARSim and Microsoft Robotics Studio. This paper describes how we created each of the two robot models from scratch, which are the difficulties we encountered, and which are advantages and disadvantages of the two simulators and their ability to faithfully simulate the motion of a real humanoid robot.

The remainder of the paper is organized in this manner. In Section ....XXX XXX...

2. The robot Robovie-M

The robot we modeled is Robovie-M (version 2) by Vstone [?]. This is sold as commercial construction kit to build a small humanoid robot. Its dimensions are 290x240x65mm, with a complexive weight of 1.9 Kg. It has 22 DOF (degrees of freedom), and 22 Sanwa servomotors: 6 for each inferior art (legs), 4 for each superior art (arms), for the trunk. The characteristics of the servomotors are in Table 1.

Motor	Torque	Speed	Size
Hyper ERG-VB	13 Kg x cm (6V)	60°/0.1s (6V)	39x20x37.4 mm
SPEC-APZ	4 Kg x cm (4.8V)	60°/0.2s (4.8V)	39x20x35.5 mm

Table 1. Robovie-M: Technical characteristics of the servomotors.

The robot is sold without a camera, and its original control board does not support a camera device. We added a control board called IT+R-Core by IT+Robotics based on a Intel XScale PXA270 at 520MHz fitted with 64MB SDRAM and 32MB FLASH memory. We built head-frame to support the camera and we fixed it to the

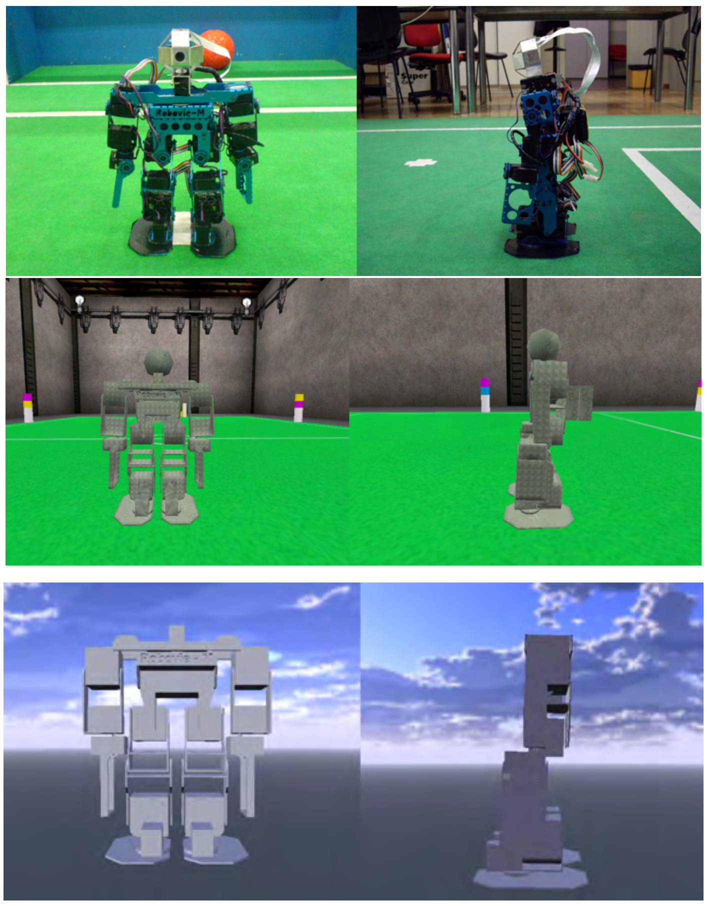


Fig. 1. (top) The modeled robot Robovie-M by Vstone; (middle) the USARSim model of Robovie-M; (bottom) the MS Robotics Studio model of Robovie-M.

shoulders of the robot, Fig. **??**. The power for the control board and for the motors is prvided by five batteries of 1.2 V and 2300 mA.

3. The USARSim Simulator

USARSim provides a high fidelity simulation at a low cost. The current version of USARSim is based on the Unreal Engine 2 game engine released by Epic Games with Unreal Tournament 2004 [?]. Buying the game (for approximately 20 Euro), it is possible to obtain the engine needed to run the simulator, together with a complete game development framework. The Unreal Editor allows to rapidly create the models of new objects and new environments. The Unreal Script, which is an ad-hoc script language, allows to define the behavior of the objects in the virtual environment even through a TCP/IP network since Unreal Engine 2 has been developed for the development of networked multi-player 3D games. Robot control programs can be written using one of the following tools: the GameBot interface, the MOAST System [?], the Player interface, or the Pyro middleware [?].

USARSim was initially developed for extremely realistic simulation of Urban Search and Rescue (USAR) robots and environments, in particular for the reference test arenas developed by the National Institute of Standard And Technology (NIST) [?] tasks. However, it has been used also as a research tool for the study of human-robot interaction (HRI) and multi-robot coordination, and now for its realistic simulation and high quality rendering its is used also in other fields of robotics. Using USARSim, problems like animation, and virtual environments rendering are automatically solved. More robotics-specific tasks of modeling platforms, control systems, sensors, interface tools and environments are accelerated by the advanced editing and development tools integrated with the game engine. With the current release of the simulator come various environmental models, different models of commercial and experimental robots, and sensor models. There are some validation of the accuracy of USARSim [?], [?], [?], [?],[?].

An important feature of USARSim, which is derived from the game, is that the scene can be observed by using egocentric view (first-person view, i.e., the robot view) or external camera view (third-person view).

4. USARSim model of Robovie-M

The virtual model of the Robovie-M robot has been developed using the program 3DStudio. We measured the sizes and weights of each single part, including the servomotors, of the Robovie-M as they come out of the box. Then, we drawn the virtual model of the robot with Adobe 3DStudio. The 3DStudio model was exported as *.ASE files (one file for each part). A .ASE file is a generic mesh file, which can be recognized and imported by the Unreal Editor, as a static mesh into a (*.usx) file to generate the correct visual rendering of each part. The physical properties of the model (such as mass, friction, and inertial tensor) has been described with the Unreal Script language, using a different script file for each of the different robot's part (for a total of 23 parts, which reduce to 13 because of the left-right symmetry).

Fig. 1 shows the final model of Robovie-M in USARSim.

In order to correctly scale the robot within the USARSim virtual environments, we adopted the following proportion $4mm = 1uu$. The uu is the measure unit used in USARSim. For example, the high of the Robovie-M is 290mm that corresponds to 72.5uu in the virtual scene of USARSIm.

To export the model in the Unreal Editor we established the hierarchy that defines the joints of each part of the robot (Fig. 3). This hierarchy is not only conceptual, but also spatial, meaning that it establishes the contact points of the various parts. These parts will be subsequently connected in USARSim by using the KDHinge joint [?].

Before importing the model under USARSim we applied the textures to the

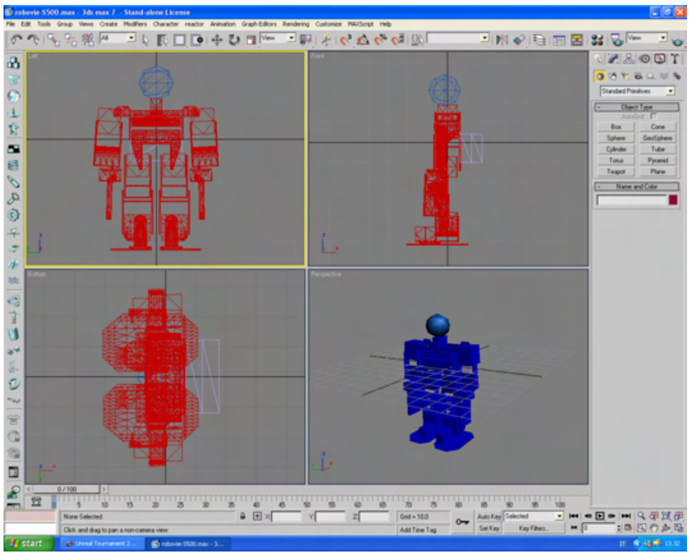


Fig. 2. Robovie-M model created with 3DStudio.

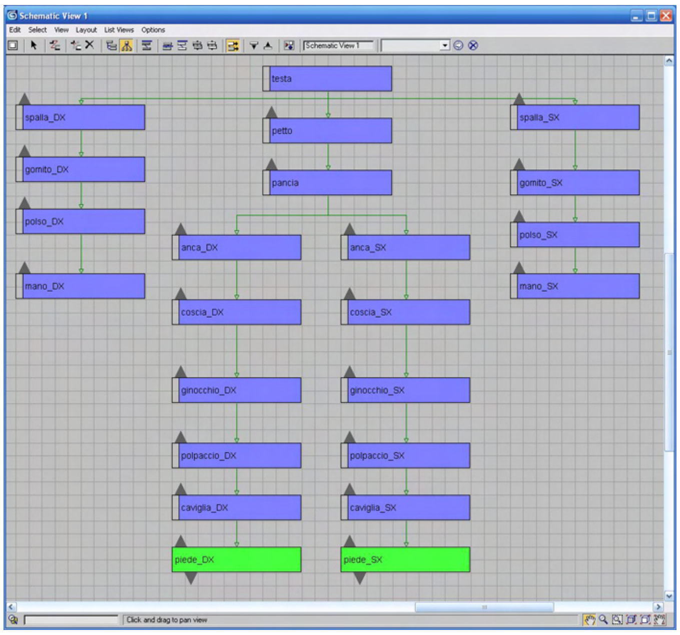


Fig. 3. 3DStudio model: Hierarchic scheme of the Robovie-M.

model. The simulator allows to choose among different "suits" (called *skins*) with which loading the model. Nevertheless, the grey skin that is seen in Fig. 4 is a part of the default textures provided with the Unreal Editor.

4.1. *Physical parameters*

The calibration of the physics parameters of the simulation is the most tricky task. In fact, wrong parameters result in a simulation with little realism. However, the realism is not the only issue that has to be considered. Also the the simulation efficiency has to be taken into account. Each single part can be characterized by[a]:

- *KMass*: Its mass.
- *KInertiaTensor*: The inertia tensor that indicates how such a mass is distributed within the space.
- *KFriction*: A friction coefficient used to model the Coulomb's friction. The force needed to move an object will be given by the multiplication of its weight with this coefficient.
- *KRestitution*: A elasticity's coefficient, used to determine the resilience of bodies after a bump. The module of the velocity of an object after a collision with another object will be calculated multipling the module of the velocity before the impact with this coefficient.

The masses of the single parts have been obtained by measuring them as explained before. The calculation of the inertia tensor and of the center of mass (COG) was made by the physical engine Karma [?] of USARSim using the collision primitives. With except for the head, for all the other parts it was been sufficient a simple box as a collision primitive. We verified the COGs calculated by Karma with the ones of the single parts calculated with 3DStudio, the error (never greater than 1uu) has been considered acceptable.

In Fig. 4 there have been represented the collision primitives used for the Robovie-M, while in purple it can be seen the centre of mass of the part.

The friction coefficient has been chosen experimentally by tuning the simulation on the reality. We performed some tests to determine the appropriate friction coefficient. The outer body parts have 0.5 friction, the foot has 0.8 with the carpet on the floor and the rest of parts have a 0 KFriction coefficient.

The elasticity coefficient has been set to zero for all the parts, because of the very little elasticity of the aluminum that composes the robot.

The properties of the KDHinge joints have been derived from the servomotors' specifics (tab. 1):

- *MotorTorque*: It is the torque moment of the joint given by torque*g/100 (where torque is the servomotor's torque in kg*cm and g is the gravity

[a]We will use the USARSim's notation in oder to describe the physical parameters of the robot's model.

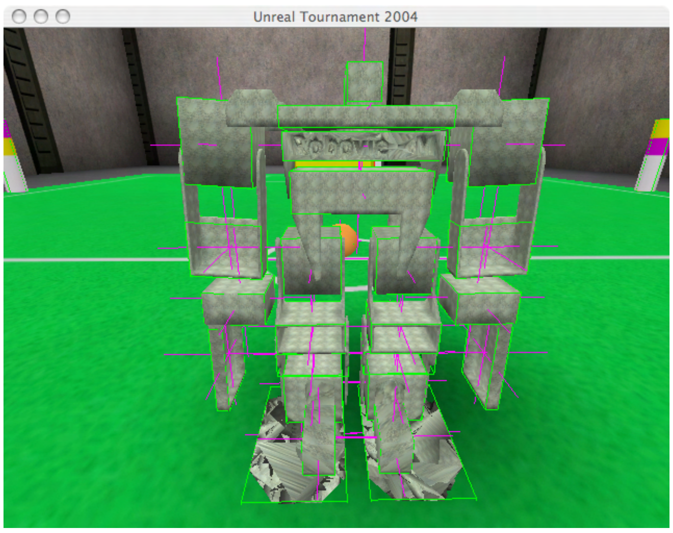


Fig. 4. 3D Model: Collision primitives.

acceleration given in m/s^2). This was set to 6.4.

- *MotorSpeed*: It is the angular speed of the joint, expressed in rad/s. This was set to 4.46
- *HingePropGap*: It is the error parameter used to make the joint fixed to the set angle ?. It is expressed in UnrealUnit (uu) and the conversion is: One degree = 182 uu. This was set to 3600.
- *KAngularDamping*: It is a parameter, similar to the KFriction one, which indicates a slew contrary to the rotation, with an angular moment that is proportional to the angular speed. It is just a kind of friction for the "activation" of the joint. This has been set to zero.

While the two first values can be obtained from the servomotors' specifics directly, the others have been calculated experimentally by performing comparative tests between the real robot and the virtual robot varying these parameters. However, a USARSim's limitation is that it is not possible to define the parameters for the joints singularly, so there have been set unique values for all the joints (defined in USAR.ini file).

5. Validation of the USARSim simulator

First, we analyzed USARSim qualitatively assessing the simulation of the humanoid robot. The physical engine of Unreal Tournament Karma ?, has an internal engine that solves a set of linear equations for each simulation's step. Karma is a library of the USARSim's MathEngine. Karma uses the Lagrange's multiplying method to model the rigid bodies. Within this model, the effect of the joints is modeled by forces that act to maintain the joint. To calculate these forces it is solved a set of linear equations using the linear prediction method (LPC). Kea is the solver of these equations. It calculates, at the end of each time-step, as the forces are applied to satisfy the joints. Kea, solves the differential equations with a certain degree of approximation that may be different every time. So the result may be different every time.

5.1. *Setting the simulator's parameters*

Since Unreal Engine 2 and Unreal Script are configurable and finely tunable, we made some experiments and comparisons in order to define the best values with which to set them in order to create a realistic simulation (for instance one can set the parameters for friction and gravity). It is clear that without a precise simulation, an error in the initial phases of the robot's movements will propagate and will increase during the simulation, affecting the final results negatively. For this reason, before performing any experiment, we visually compared the real robot's behavior with the simulated one in order to define the best values of the needed parameters. We considered that the most important ones, and consequently the ones that a researcher who would like to reproduce our work has to know, are the *PenetrationScale* and the *ContactSoftness*. Marco Zaratti explained them in [?].

PenetrationScale [default: 1.0]. We set this value as: 5.

ContactSoftness [default: 0.01] (karma units). We set this value as: 0.001.

5.2. *Comparison between a RR's walk and a VR's walk*

In addition to the visual confrontation, USARSim can return time and spatial coordinates of the simulated objects. We choose to analyze the real and virtual robots' walked distance and we rescaled the time spent by the virtual robot. We analyzed and to compared the coordinates of the center of gravity (COG) of the robot in the frames of reference of the world. We generated a motion sequence to make 8 steps along a straight line. This motion has been written on the RR and on the VR and we repeated this motion 10 times (both for the RR and the VR). Again, we compared the time and the coordinates of the robot in the real and in the simulated world. We placed the RR on the floor, in our laboratory. Then we started the RR from a fixed starting point and we made a video with a digital camera, which allows a time reference. The ground-truth position of the RR was given by a flexible ruler placed along its trajectory. We let the robot walking straight.

We took the time necessary to complete the 8-step walking task by the real robot. During the experiment, we evaluated the position of the robot every 25 seconds, in order to plot the behavior of the robot, in terms of speed, trajectory, and distance run as functions of the time. This was achieved thanks to the flexible ruler placed along the robot walking direction. Then, we loaded the VR in the Unreal environment, and we did the same thing. USARSim gives the values of the speed, time, and position of the COG of the robot every thread cycle. USARSim uses a "heavy" graphical engine. This latter consumes several processor's and memory's resources, causing slow down during the simulation. This surely affects the time spent by the VR in performing its 8-step walking task. We simulated the tests without running any other application, so giving the most resources to the computer as possible. We used an Apple PowerBook G4 with 768 MB of RAM memory, nVidia GeForce FX Go5200 64 MB of VR, and a G4 1.5 GHz PowerPc processor, with the OS X 10.4 operative system.

Then we compared the data of the RR with these of the VR. These graphs have been obtained by averaging the results of our 10 testing walks of the RR and our 10 testing walks of the VR.

We made the VR and the RR walking along a straight direction, which is represented in these graphs as the x-coordinate. The plot in Fig. 5.2 represents the average of ten cases during which the two robots are walking along a straight direction (x-axis). Then, the plot in Fig. 5.2 shows the quadratic error, intended as the difference of the y-values of the trajectory coordinates. The y-values represent the values of the deviations of the robots' trajectories from the given (imposed, represented by the x-axes) direction, along its orthogonal axes. In each case the error has been calculated as the difference between the VR values and the RR values, respectively.

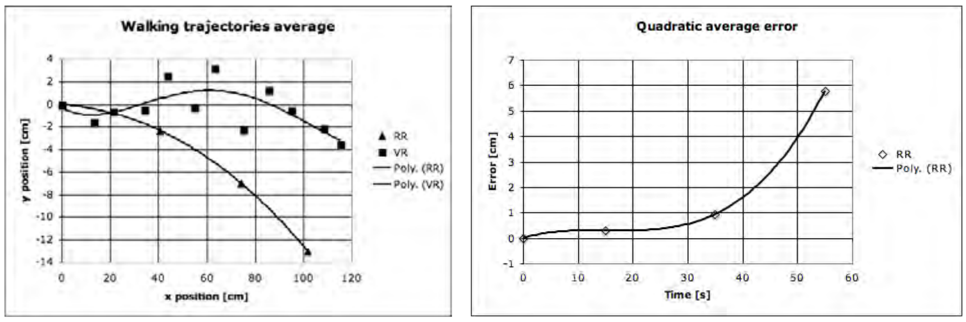


Fig. 5. VR's vs RR's walking progression

5.3. *Examination of the fluidity of the 3D representation of the environment*

As an additional experiment, we tested the performance of the USARSim in terms of frames per second. Since using more than one robot is necessary specific in order to use the simulator to test virtual soccer games in which there are presented more than one robot. Using more than one robot, the computational complexity of the simulation increases, than the performances decreases, in terms of scenes fluidity and speed. We tested the performance of USARSim using frame speed (fps) frames per second. USARSim directly return this information to the user. In addition, it allows also to store it in a file log, with the other information (robot's speed, time, etc.). Two tests have been performed: One with one VR and the other with two VRs. In each test the robots were still fixed during the first 7 s, and they moved after that.

The plot in Fig. 5.3 shows the rendering of USARSim during, firstly a one VR simulation (blue line), and then a two-robot simulation (red line). The physics performance has been obtained redirecting the visualization to a map's corner. With this expedient, it has been possible not to have to visualize polygons and textures, which need to be rendered. The plot in Fig. 5.3 shows the physics performance of the simulator. As in the previous plot we simulated a one-VR and a two-VRs condition, with the robots in a fixed position (again before the 7 s) and during a moving condition (again after the 7 s).

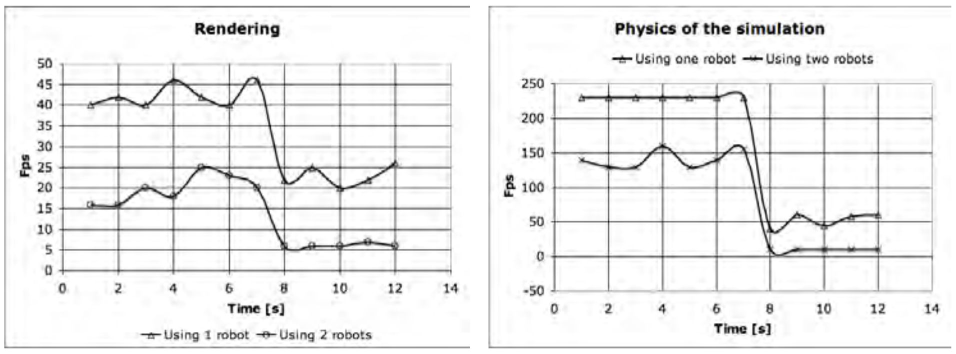


Fig. 6. USARSim performance

5.4. *Performance of a penalty kick*

In this test, the robot has to perform 8 steps and after that to kick a ball placed on the penalty kick spot in a regular RoboCup field. We placed the real robot in the laboratory and tested three different kick penalty situations. As in the previous experiment, we placed a flexible ruler along the RR walking direction, in order to obtain the exact values of position, in terms of x and y coordinates. Then, we tested the time required to perform this task. We made different movies with a digital camera, to document these tests. At the link *"http://www.dei.unipd.it/˜emg/downloads/penaltyComparison.wmv"* there is the movie, representing the RR and the VR, during the performing of the same kick penalty. In Fig. 5.4 there is shown an example of the egocentric view of the robot during a walk: Here there is the robot in a single frame, while performing the penalty kick, from an egocentric view. In Fig. 5.4 there is shown an analogous scene, but viewed from an exocentric (external) view.

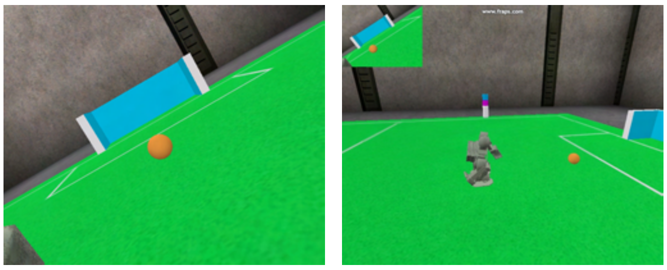


Fig. 7. The different views allowed by USARSim. The scenes represent different frames during the VR's performance of a kick penalty

6. Microsoft Robotics Studio

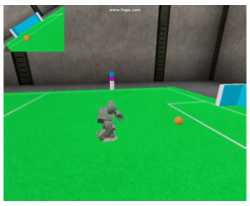
7. Microsoft Robotics Studio model of Robovie-M

8. Validation of the Microsoft Robotics Studio simulator

9. Discussion

10. Conclusion

11. Future work



Emanuele Menegatti received his/her M.S. and Ph.D. degrees from the University of Name, in Year1 and Year2, respectively. From Year3 to Year4, he/she was Researcher/Engineer/... at the Institute of Name, and worked on the project of Title. From Year5, he/she was Associate Professor, and became full Professor in Year6 at the University of Name. Now, he/she holds Position at the Faculty of XXX of the University of Name. He/she is also the President/Director of Organization/Society.

First-Author is the author of over X technical publications, proceedings, editorials and books. His/her research interests include Mind topics, Body topics and Application topics (see IJHR's flyer). He/she is the coordinator of the collaborative research network on Humanoid Robots at the University of Name, Country. He/she is an active member of the Y Societies, and was the General Chair of Z Conferences in Country, and the Co-Chair of Z Conferences in Country.

Nicola Greggio received his M.Sc. degree in Electronic Engineering from the University of Padova, Italy, in 2005. Currently he is a Ph.D. Student in ICT and Robotics Engineering in the class of Experimental Sciences from the Scuola Superiore S.Anna, Pisa, Italy. From 6/1/2006 to 12/31/2006, he also served as research assistant with the Institute IT+Robotics, Vicenza, Italy, where he was responsible for research projects in the area of Soccer Humanoid Robots.

His research interests include Artificial Intelligence topics, BioRobotics topics and Rehabilitation Engineering topics (see IJHR's flyer).