SimRobot – A General Physical Robot Simulator and Its Application in RoboCup*

Tim Laue, Kai Spiess, and Thomas Röfer

Bremer Institut für Sichere Systeme, Technologie-Zentrum Informatik, FB 3,
Universität Bremen, Postfach 330 440, 28334 Bremen, Germany
{timlaue, kspiess, roefer}@tzi.de

Abstract. This paper describes SimRobot, a robot simulator which is able to simulate arbitrary user-defined robots in three-dimensional space. It includes a physical model which is based on rigid body dynamics. To allow an extensive flexibility in building accurate models, a variety of different generic bodies, sensors and actuators has been implemented. Furthermore, the simulator follows an user-oriented approach by including several mechanisms for visualization, direct actuator manipulation, and interaction with the simulated world. To demonstrate the general approach, this paper presents multiple examples of different robots which have been simulated so far.

1 Introduction

When working with robots, the usage of a simulation is often of significant importance. On the one hand, it enables the evaluation of different alternatives during the design phase of robot systems and may therefore lead to better decisions and cost savings. On the other hand, it supports the process of software development by providing an replacement for robots that are currently not on-hand (e. g. broken or used by another person) or not able to endure long running experiments (e. g. learning tasks [1]). Furthermore, the execution of robot programs inside a simulator offers the possibility of directly debugging and testing them. This is a great benefit when working with platforms that do not offer any direct debugging facilities, e. g. the Sony AIBO robot.

In the past, several robot simulators have been developed with different main focus on complexity, accuracy, and flexibility. There are also differences in the possibility of creating and integrating own robot models and virtual environments. Some of the simulators are restricted to a two dimensional environment or are only approximating dynamics and realistic interaction of the robots with the environment. In the following, we give a short overview of current related works on robot simulators with accurate dynamics simulation for three dimensional environments and with relevance to the RoboCup domain. These will be compared with SimRobot.

* The Deutsche Forschungsgemeinschaft supports this work through the priority program "Cooperating teams of mobile robots in dynamic environments".

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 173–183, 2006.
© Springer-Verlag Berlin Heidelberg 2006

UCHILSIM [1] is a robot simulator developed by the University of Chile. It is specially designed for the RoboCup Four-legged League. The simulator contains dynamics simulation using the Open Dynamics Engine (ODE) [2], a graphics engine, and has a window based graphical interface. The environment and the robots are described in a VRML structure extended by nodes for simulator elements and physical attributes. It has interfaces to their UChile1 software package and their learning component. At the current stage, this simulator is rather specific for one RoboCup league.

Another 3-D simulator used in the RoboCup domain is Übersim [3], which has a focus on vision-centric robots in dynamic environments. It has a client/server based architecture, where clients communicate with the server over TCP/IP. The simulator also uses ODE for dynamics simulation. Own robots can be modeled by programming their structure in C classes. In the current release [4], only two sensors are predefined: a camera sensor and an inclinometer. At the moment there seems to be no graphical user interface for interacting directly with the robots or the environment during simulation time.

A more general 3-D multi-robot simulator with graphical interface and dynamics simulation is Gazebo [5], a part of the Player/Stage project [6]. This simulator has a large variety of sensors and comes with models of existing robots such as the Pioneer2DX or the SegwayRMP. The robots and sensors can be controlled by the Player server or controllers can be written using a library provided with the simulator. The simulated environment is described in XML files using several predefined elements and robots. It also offers the possibility of creating and integrating own robots as plug-ins but this has to be done by code-based modeling in C.

Webots [7, 8] is a commercial robotic simulator developed by the Cyberbotics Ltd. It has an ODE-based physics simulation and provides several robot models such as Sony Aibo, Khepera, or Pioneer2. The robots and the environment are described using the VRML standard for graphical models, extended by nodes for the Webots elements, sensors, and physical attributes. Controllers can be programmed in C++ or Java and connected to third party software through a TCP/IP interface.

In comparison with these simulators, the following features of SimRobot may be pointed out and will be described in this paper: The simulator is not limited to any special class of mobile robots[1]. By using an XML-based modeling language, users are enabled to specify robots and their environments completely without any additional use of other programming languages. A large set of body elements, actuators and generic sensors allows the free composition of arbitrary robot models. An important element of simulations, which is ignored in many cases, is the support of the work of the user. This is addressed by SimRobot by providing several possibilities of visualization and interaction with the simulated world.

To simulate rigid body dynamics, SimRobot also uses ODE, since this engine has a wide variety of features and has been used successfully in many other

[1] Admittedly, SimRobot will not support the special domains of underwater robotics and aircrafts.

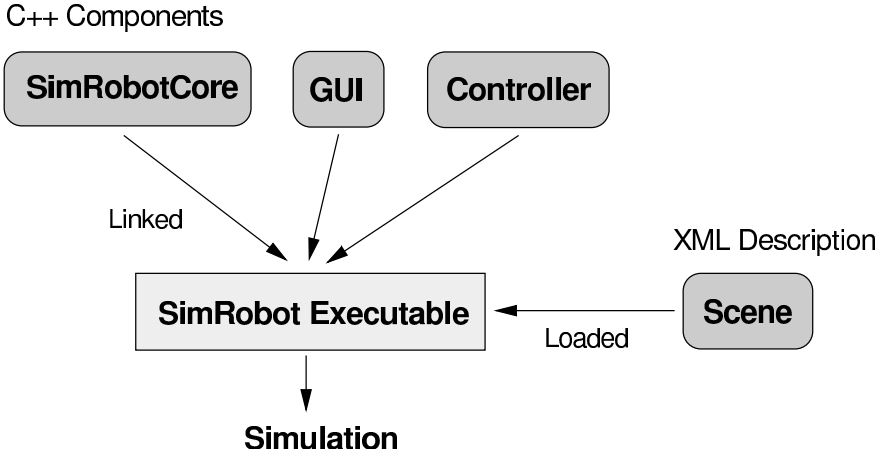


Fig. 1. The modules of SimRobot and their dependencies

projects. The visualization as well as the computation of imaging sensor data is based on OpenGL, because this industry standard offers the best performance on modern hardware on different platforms.

Previous works on this simulator project have been a kinematic robot simulation [9] and a preliminary release (without dynamics) for the GermanTeam in the Sony Four-legged Robot League [10].

This paper is organized as follows: Section 2 describes the general architecture of SimRobot, Section 3 shows several special features of the user interface, the Sections 4 and 5 describe the elements which SimRobot is able to simulate, some applications of the simulation are shown in Section 6, the paper ends with the conclusion in Section 7.

2 Architecture

2.1 Components of the Simulator

As shown in Fig. 1, SimRobot consists of several modules that are linked to one single application. This approach, which is different from many other client/server-based simulation concepts, has been chosen because it offers the possibility of halting or stepwise executing the whole simulation without any concurrencies. It allows also a more comprehensive debugging of the executed robot software.

The main components of SimRobot are:

SimRobotCore. The simulation core, which may also be qualified as engine or kernel, is the most important part of the application. It models the robots and the environment, simulates sensor readings, and executes commands given by the controller or the user. Even most parts of the visualization are integrated into the simulation core.

```
<Hinge name="subWheelAxis">
    <AnchorPoint x="0.022" y="0" z="0"/>
    <Axis x="0" y="1" z="0"/>
    <Elements>
        <Cylinder radius="0.006" height="0.004">
            <Rotation x="90" y="0" z="0"/>
            <Appearance ref="VeryDarkGray"/>
            <PhysicalAttributes>
                <Mass value="0.02"/>
            </PhysicalAttributes>
        </Cylinder>
    </Elements>
</Hinge>
```

Fig. 2. An excerpt from a scene description using the `RoSiML` language. A sub-wheel of a Small Size robot (see Sect. 6) is modeled via a hinge joint and a cylinder.

The kernel is platform independent. It is connected to a user interface and a controller via a well-defined interface. This enables an easy porting to other platforms as well as the embedding into other applications. A previous version had been used in the *RobotControl* software of the GermanTeam [10]. At the moment, the current kernel becomes integrated in the Linux framework of our Small Size team (see Sect. 6).

GUI. The user interface is responsible for the display of information (e. g. different views of the simulated scene) and for the interaction with the user. It is described in more detail in Sect. 3.

Controller. The controller implements a sense-think-act cycle. In each simulation step, it is called by the simulation, reads the available sensors, plans the next action, and sets the actuators to the desired states. A controller which is suitable for the modeled scene has to be provided by the user. Normally, it contains the control software of the simulated robots, but it may also be left empty.

Scene. The specification of the robots and the environment, in the context of SimRobot named as *scene*, is modeled via an external XML file and loaded at runtime. It is described in the following section.

2.2 Specification of Robots and Their Environment

The use of an external specification language allows the modeling of scenes without any modifications or extensions of the source code of the simulator. Thus the modeling process becomes simpler and people without programming skills are also enabled to use the simulator.

Together with researchers from the Fraunhofer Institute for Autonomous Intelligent Systems, the specification language `RoSiML` (Robot Simulation Markup Language) [11] has been developed (see example in Fig. 2). It is a part of a joint effort to establish common interfaces for robot simulations. The aim is to exchange components between different simulators and to allow the migration of robot models among simulators without any complicated adaptions.

The language by itself has been specified in XML Schema. This is a popular choice, since XML is supported by a variety of editors and there also exist many ready-to-use parsers. The cascaded structure of XML documents is also quite suitable to reflect the structure of scenes inside the simulator, as it uses the scene graph approach from computer graphics to organize and process all elements.

3 User Interface

The user interface (Fig. 3) of SimRobot has been designed to allow as much visualization and interaction as possible as well as to be flexible enough to handle

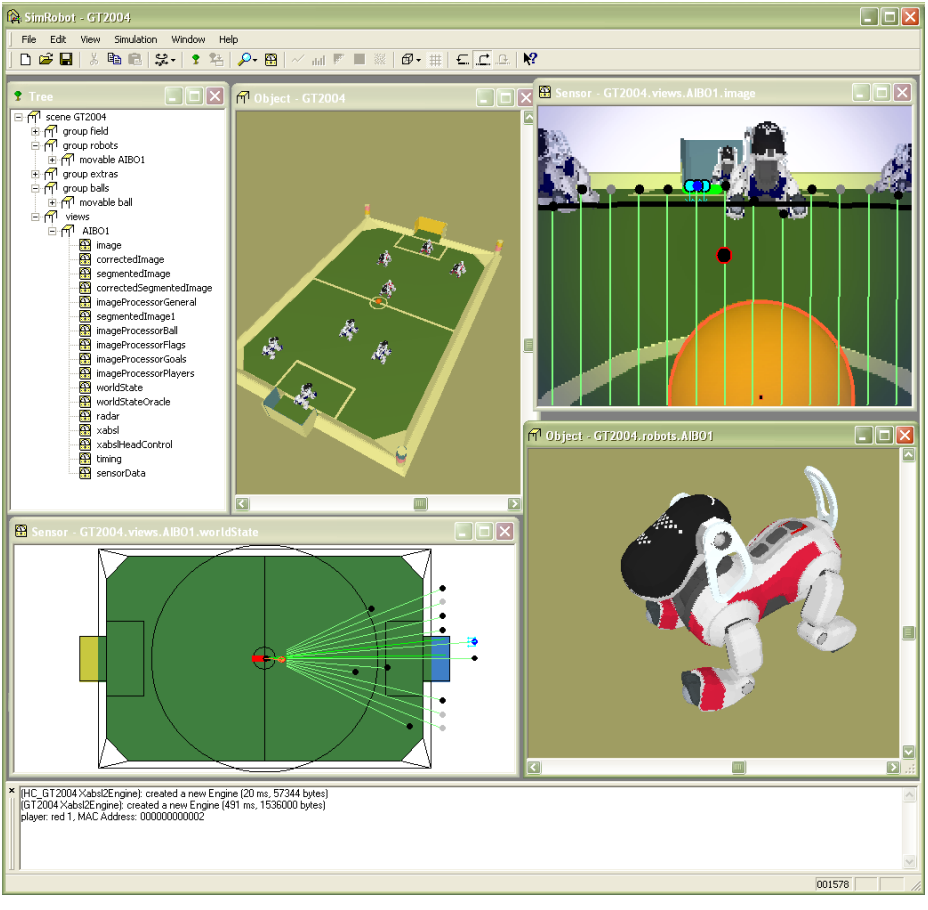


Fig. 3. The user interface of SimRobot while simulating the German Team 2004. The internal frames show (from left to right, top to bottom): the object tree, a view of the whole scenario, a simulated image of an AIBO camera including overlays from the image processor of the simulated software, a user defined view of a robot's world model, a close view of a single robot, and the console.

simulations of any different kind of environments. Therefore a tree of all objects of the scene is the starting point for all user operations. Each node of that tree may be selected to open a view for that kind of object. In case of actuators (e. g. a hinge joint), a control for direct manipulation is opened. For sensors, several different visualization modes are implemented. Through this concept, it is also possible to open several views of arbitrary subsets of the scene graph (as shown in Fig. 3). These views offer a zoom, rotation, panning and different grades of detail as well as the possibility to switch between the appearance and the physical model of single objects, as shown in Fig. 4. Furthermore, it is possible to interactively drag and drop and rotate objects inside the scene or to apply a momentum to an object (e. g. to let a ball roll). This is quite useful to arrange different settings while testing e. g. a robot behavior.

To add own views to a scene (as e. g. the world state in Fig. 3), an interface for user-defined views has been implemented which enables the definition of own debug drawings from inside the controller.

Other elements of the user interface are an editor for the scene description files and a console for text output from the controller.

4 Physics and Actuators

As aforementioned, ODE is used for simulating rigid body dynamics. Therefore, the set of simulated objects results from the abilities of that engine. Nevertheless, we had to implement several extensions to fulfill the requirements of our general approach.

4.1 Rigid Bodies

For the design of robot shapes and the environment, several rigid bodies (mostly adopted from ODE) may be used. The basic bodies are: Box, Cylinder, Capped-Cylinder and Sphere. Each of them has divers attributes describing its appearance (e. g. color) and physical behavior (e. g. mass or friction).

This set has been extended by the so-called ComplexShape. A body of that class has a graphical representation based on a number of geometric primitives and a physical representation based on a set of basic bodies which may approximate the shape. This approach allows the use of sightly detailed elements with an accurate dynamic and collision behavior. The parts of the AIBO model in Fig. 4 have been described by ComplexShape objects.

4.2 Actuators

To achieve a high grade of flexibility in creating robots, SimRobot supports various kinds of joints corresponding to the joints provided by ODE. There are simple rotational joints with one axis or two perpendicular fixed axes, a translational joint, a ball and socket joint and a simple wheel suspension like joint with free rotation about one axis and rotation and compression along the other axis. The joints can connect two movable rigid bodies or one body with

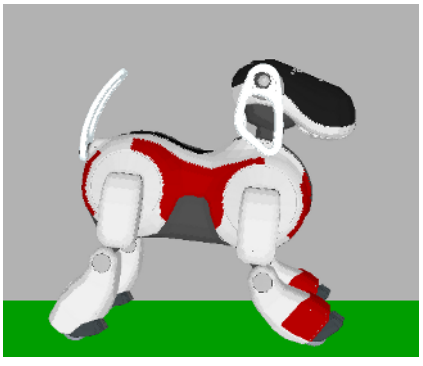
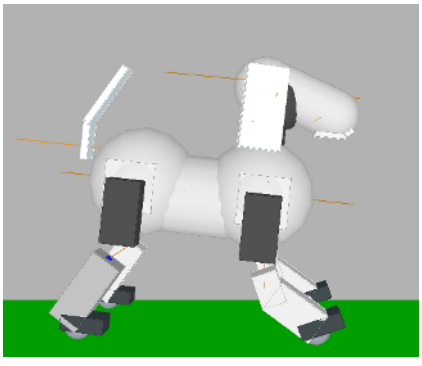
a)  b) 

Fig. 4. A Sony AIBO model consisting of a set of ComplexShape objects. a) The graphical representation with details such as toes, tricot elements and LEDs on the head. b) The physical representation consisting of simple boxes, spheres, and capped cylinders.

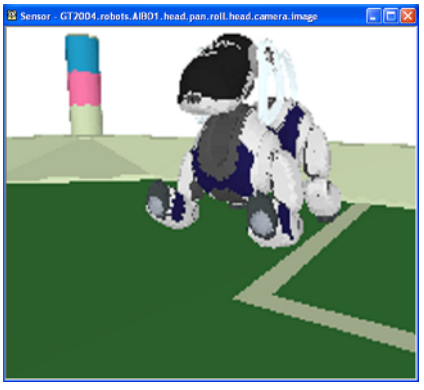
the static environment. The range of motion of a joint can be limited. Joints can be unpowered or a motor can be attached to each axis. Due to the fact that the motor provided by ODE has no specific controller, we implemented a P and a PID controller to simulate servo motors. Additional controllers can easily be integrated, if necessary. Besides the servo motor, a simple velocity controlled motor is also provided. In ODE, joints are frictionless, i. e. a pendulum will never stop swinging if no collision occurs. So we implemented a simple friction model for damping motion in unpowered joints.

4.3 Automatic Generation of Object Compounds

In SimRobot it is possible to specify complex rigid bodies through using the provided simple elements without explicitly summarizing them with additional XML tags in the scene description or performing additional calculations by the user. The structure of the scene tree is used for this automatism. In the scene tree, all elements of a subtree beneath a joint down to the leaf nodes or to other joints are treated as one single physical body. All elements in this compound are stuck together and behave as one single object. For correct dynamical behavior, the masses, centers of masses and inertia tensors of all contained objects are combined to a single rigid body. The geometrical representations of all combined objects build the collision behavior of the compound object. In addition, single objects or object groups can be excluded of the above described automatism of combining and are treated as independently moving objects or compound objects.

5 Sensing

SimRobot realizes sensor simulation via a set of generic sensor classes, i. e. it does not include specific sensors such as a special laser range finder or similar devices. The available sensor classes are:

a) 

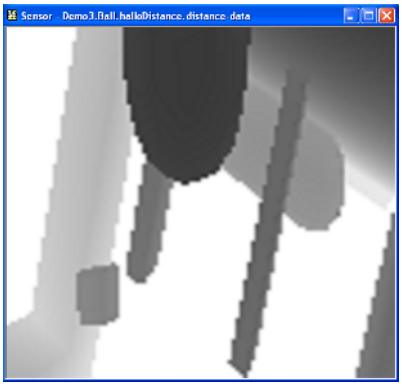
b) 

Fig. 5. The results from two different sensors: a) An image from a simulated AIBO camera and b) a depth image inside a demo scene mainly consisting of boxes and cylinders

Camera. The camera sensor generates a two-dimensional array of RGB pixels which have a color depth of 24 bits (one example is shown in Fig. 5a). Aside from the standard perspective projection, it is also possible to use a spherical projection with an equal angular distance between all pixels. To speed up the process of image generation significantly, SimRobot is able to support hardware accelerated off-screen rendering. This is a feature which several manufacturers implement on their graphics hardware nowadays.

Distance Sensor. This sensor is quite equal to the camera, but instead of pixels, it returns distances gained from the depth buffer of rendered images. Due to its generic approach, there exist several applications for this sensor: the generation of depth images (Fig. 5b), simulating a laser range finder (Fig. 6b), or being modeled as a single value PSD sensor in an AIBO robot.

Bumper. For detecting the collision of objects, e. g. to model a touch sensor, the so-called Bumper has been implemented. Unlike all other sensors, it is not a special object in the scene graph. Furthermore, each body or group of bodies may be assigned to be collision sensitive. This allows the creation of arbitrarily shaped touch sensors. The information about collisions is directly gained from the dynamics engine. As an addition, it is also possible for the user to interactively provoke the sensing of a collision. We have used this feature e. g. for pressing the buttons of simulated AIBO robots.

Actuator State. There also exist interfaces for inquiring the current states of actuators. This includes the angles of joints as well as the velocities of motors.

6 Applications

As aforementioned, SimRobot has been used by the GermanTeam to simulate the robots and the environment of the Sony Four-legged League. In previous years, a kinematic version of SimRobot has been used, which needed several work arounds

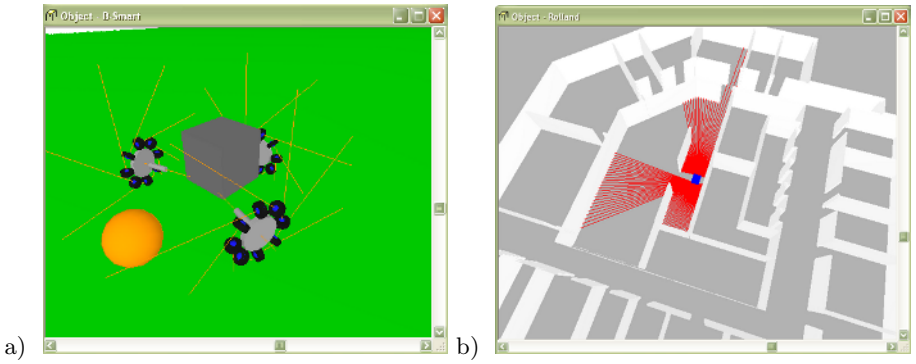


Fig. 6. Applications of SimRobot: a) The physical representation of a Small Size platform with an omnidirectional drive. The lines around the wheels denote the axes of the sub-wheels. b) A simulated office environment with a navigating robot. The lines starting at the robot denote the rays of its laser range finders.

to simulate correct body postures e. g. when executing kick motions. Due to the absence of a collision handling, the ball had also to be moved manually. Together with the physical simulation, a new model of the AIBO ERS-7 robot, which is shown in Fig. 4, has been created. The appearance was kept, but all elements are now additionally represented by rigid bodies, which are connected with adequate joints to allow an accurate simulation of all 20 degrees of freedom. This model enables a direct execution of the desired motions as well as an accurate handling of collisions with other robots or the ball.

A completely different kind of robots is used in the RoboCup Small Size League. To allow fast and flexible motions, wheel-based robots with omnidirectional drives are used. These consist of three or four wheels which are arranged in a triangle or a rectangle respectively. Each of these wheels is surrounded by a set of small passive sub-wheels which enable the robot to move sidewards. This approach has also found its way into the Middle Size League. To integrate SimRobot into the environment of our Small Size team B-Smart [12], a platform with such a drive has been modeled, as shown in Fig. 6a. The XML description of a single sub-wheel has already been presented in Fig. 2. Though having a completely different structure than the previously modeled walking robot, the platform has been simulated successfully, showing the expected motion behavior when driving around.

An application outside the RoboCup domain is the simulation of *Rolland* - The Bremen Autonomous Wheelchair [13]. This robot has a differential drive and is equipped with two laser range finders which are connected to a standard notebook which executes the control software. Rolland performs navigation tasks in office environments, as shown in Fig. 6b. Since this robot currently has a two dimensional model of the world and is programmed to never collide with other objects, the physical simulation is of minor usefulness for this platform. Nevertheless, it demonstrates the application of SimRobot in a large-scale environment and the usage of distance sensors.

7 Conclusion and Future Works

This paper presented SimRobot, a robot simulator which supports rigid body dynamics and the simulation of a variety of sensors and actuators. Through its generic concept and the use of a special modeling language, it is able to simulate arbitrary user-defined robots without any modifications of the simulator itself. This has been shown by means of several examples of completely different robot platforms which have been simulated successfully. The flexible approach of the user interface has been able to offer a variety of visualizations and options for interaction for each simulated environment.

The German Team as well as the B-Smart team will use this simulator in 2005. It will also be made available to other RoboCup teams, as described in the following section.

Among other things, future works will concentrate on the simulation interface standardization efforts described in Sect. 2.2. These will include generic plug-in interfaces for sensors and actuators as well as the definition of a common controller interface.

Availability of SimRobot

The simulator presented in this paper will be released under an open source license in the near future. The most current available version, which does not include the dynamics engine but most other features, has been released as a part of the German Team 2004 code release [14]. An up-to-date binary version for Microsoft Windows which includes several examples is also available [15]. A release of a Linux version is planned.

Acknowledgements

The authors would like to thank all people who contributed to the development of SimRobot by directly providing code or by testing and reporting problems, the authors of the Open Dynamics Engine for making available the basis for our physical simulation, and the researchers from Fraunhofer AIS for their contributions to the Robot Simulation Markup Language.

References

1. Zagal, J.C., del Solar, J.R.: UCHILSIM: A Dynamically and Visually Realistic Simulator for the RoboCup Four Legged League. In: RoboCup 2004: Robot Soccer World Cup VIII. Lecture Notes in Artificial Intelligence, Springer (2004)
2. Smith, R.: Open Dynamics Engine - ODE (2005) www.ode.org.
3. Go, J., Browning, B., Veloso, M.: Accurate and flexible simulation for dynamic, vision-centric robots. In: Proceedings of International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'04). (2004)
4. Go, J., Browning, B., Veloso, M.: Carnegie Mellon UberSim Project (2004) http://www-2.cs.cmu.edu/ robosoccer/ubersim/.

5. Koenig, N., Howard, A.: Gazebo - 3D multiple robot simulator with dynamics (2004) http://playerstage.sourceforge.net/gazebo/gazebo.html.
6. Gerkey, B., Vaughan, R.T., Howard, A.: The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In: Proceedings of the 11th International Conference on Advanced Robotics, Coimbra, Portugal (2003) 317 – 323
7. Michel, O.: Cyberbotics Ltd. - WebotsTM: Professional Mobile Robot Simulation. **1** (2004) 39–42
8. Cyberbotics Ltd.: Cyberbotics Webots (2005) `http://www.cyberbotics.com/products/webots/`.
9. Röfer, T.: Strategies for using a simulation in the development of the Bremen Autonomous Wheelchair. In Zobel, R., Moeller, D., eds.: Simulation-Past, Present and Future, Society for Computer Simulation International (1998) 460–464
10. Röfer, T., Laue, T., Burkhard, H.D., Hoffmann, J., Jüngel, M., Göhring, D., Lötzsch, M., Düffert, U., Spranger, M., Altmeyer, B., Goetzke, V., v. Stryk, O., Brunn, R., Dassler, M., Kunz, M., Risler, M., Stelzer, M., Thomas, D., Uhrig, S., Schwiegelshohn, U., Dahm, I., Hebbel, M., Nisticó, W., Schumann, C., Wachter, M.: GermanTeam RoboCup 2004 (2004) Only available online: http://www.robocup.de/germanteam/GT2004.pdf.
11. Ghazi-Zahedi, K., Laue, T., Röfer, T., Schöll, P., Spiess, K., Twickel, A., Wischmann, S.: Rosiml - robot simulation markup language (2005) http://www.tzi.de/spprobocup/RoSiML.html.
12. Kurlbaum, J., Laue, T., Lück, B., Mohrmann, B., Poloczek, M., Reinecke, D., Riemenschneider, T., Röfer, T., Simon, H., Visser, U.: Bremen Small Multi-Agent Robot Team (B-Smart) Team Description for RoboCup 2004. In: RoboCup 2004: Robot Soccer World Cup VIII. Lecture Notes in Artificial Intelligence, Springer (2005)
13. Mandel, C., Hübner, K., Vierhuff, T.: A Demonstrator for Cognitive Aspects in Service Robotics. In: XXVII Annual Meeting of the Cognitive Science Society (submitted). (2005)
14. German Team: German Team web site. (2005) http://www.germanteam.org.
15. Röfer, T.: SimRobot Website (2005) http://www.tzi.de/simrobot.