NAO robot simulation for service robotics purposes

Stefano Michieletto
Email: michieletto@dei.unipd.it

Davide Zanin
Intelligent Autonomous Systems Lab (IAS-Lab)
Department of Information Engineering (DEI)
University of Padova
Email: zanindav@dei.unipd.it

Emanuele Menegatti
Email:emg@dei.unipd.it

***Abstract*—Humanoids playing soccer are required to solve a great variety of tasks: from perception to body motion, from decision making to team coordination. On the other hand, results from this community are sometimes underestimated or unexploited because of the dedicated software developed. In particular simulators are often designed for a specific robotics platform or in some other cases the integration with existing software and frameworks is hard to implement and time consuming.**

In this paper we introduce a novel virtual model to simulate the humanoid robot Aldebaran NAO. The URDF (Unified Robot Description Format) standard has been followed in order to maintain the model as general purpose as possible. Related plug-ins to make it works in Gazebo and V-REP simulation environments were also developed in order to test the model under ROS (Robot Operating System), a very common robotics framework.

I. Introduction

The importance of simulation environments is increasing in both technical and research areas. Simulating mobile robot platforms provides several advantages compared to the real world: no limitations in the number of robots due to lack of money, no sensors or actuators inaccuracies, flexibility in the environment complexity, and debugging capabilities. These features are even more important working with humanoids. Moving around with humanoid robots means take into account of at least 10 DoF, maintaining meantime the robot stability. Falls might become unavoidable and a real scenario probably involve damages to robots or sensors.

One of the most diffuse humanoids is Aldebaran NAO. A lot of research laboratories bought one of them and it is the Standard Platform at Robot Soccer World Cup (RoboCup). Due its use in RoboCup, in the last decade a large amount of humanoid soccer simulators has been developed using NAO as test platform. Nevertheless, only a small part of the tested algorithms are actually available in the robotics field. The most of these algorithms come from projects based on popular frameworks o middleware[5], where other communities than soccer roboticists are interested in.

The same is for the simulators themselves: they are mainly focused on soccer competitions, only few specific platforms are available and no attention is paid on common situations in manifacturing or home environments. Thus, it is very difficult to reuse in a different environment the features introduced, even if they are well implemented. And the viceversa is also true: innovation in service or industrial robotics takes long to be integrated in soccer simulators, with a consequent technological delay. Several attempts have been done to close the gap. In [7] Kalyanakrishnan and Stone proposed simulators such as Microsoft Robotics Studio [6] and Cyberbotics Webots [10] to provide to RoboCup teams useful tools for better programming their robots. A comparison with 3 RoboCup platforms is exposed starting from the experiences author's team made in the competitions. They put attention on physics engines testing skills such as walk, kick, turn, and get up. Data from all simulators are applied to the physical robot and no one work consistently, however they are able to improve their performances merging information from the different simulations. In [15] authors tested a 3D RoboCup simulator (SimSpark [11]) and a novel robotics simulator developed by their Faculty (SimTwo [12]). The comparison approach was based on a benchmark composed of two different motors movements, the most significantly one is walking. A similar approach was used in [9] and [4] comparing the distance walked by a real robot and the one simulated in USARSim [2] and Microsoft Robotics Studio [6]. An additional test was performed increasing the number of robots in the scene and counting the frame per seconds (FPS) generated by each simulator.

In our work, we have followed this trend moving the line even more closer to home robotics simulators. Our aim is to let soccer roboticists simulate their algorithms using a very common framework in service robotics community, in order to take advantage of the good practices and help to solve open problems. We created a novel virtual model of Alberan NAO suitable for different Open Source 3D simulators. In particular, we tested it in Gazebo [8] and V-REP [3]. As highlighted in [17] one of most popular and well performing simulators is Gazebo, while V-REP is an emerging platform which became Open Source at the beginning of 2013. Two different tests are planned to verify the robot model is near to real one. The former test involved straight walk, the latter is focused on turn around.

The remainder of the paper is organized as follows: in Section II an overview and a comparison of the two simulators chosen to try the model are presented. The NAO characteristics and a detailed description of our virtual model are given in Section III. The architectures used, the tests performed and results collected are described in Section IV. Conclusions and future works are contained in Section V.

II. Simulation Environments

A. Gazebo

Gazebo [8] is an Open Source 3D simulator which is capable of simulating a wide population of robots, sensors and objects. The project started in 2002 at the University of Southern California. It was designed to help researchers working on robotic vehicles in outdoor environments, but it manages also indoor situations. Gazebo has historically been used as a research and development tool to rapid prototyping, locomotion, robot competition, person simulation, and regression testing. It nominally provides multiple physics engines including ODE [19] and Bullet [18] (not completely supported also in the last software release) and general parameters such as accuracy and performance are exposed to better suite to user needs. Gazebo relies on OGRE [20] to render 3D graphics and improve realism generating correct lighting and shadows using state of the art GPU shaders.

Several objects can be loaded ranging from simple shapes like cubes or spheres to complex models like buildings or animals. Each object has his own attributes: mass, velocity, friction and several other properties to push physic and aspect as realistic as possible. Many robot models are provided in a community-supported database, and everyone can create his own model defining a physical entity with dynamic, kinematic, and visual properties. Gazebo can also generate information from different kind of sensors: laser range finder, 2D and RGB-D cameras, contact sensors, inertial measurement units (IMUs) and radio-frequency identifications (RFIDs). The use of sensors is very important: in this way robots can to act in different ways depending on data read in the simulated world. Custom plug-ins can be developed to make the robot model interact with the world. Plug-ins provide direct control over all robot aspects and manage data collected by sensors.

Many simulation parameters can be directly controlled also by a QT-based graphical interface. Gazebo is compatible with several Linux distributions and a native interface to ROS [13] and Player [14] is provided in order to integrate different kind of robots. In this way, it is not necessary to use the API to develop a specific interface for each robot or sensor, but any device working with ROS or Player can be simulated. The supported programming language are C, C++, Java and Python.

B. V-REP

The 3D robot simulator V-REP [3] has been developed to perform simulation of factory automation systems. V-REP is available in 4 licenses: Player (free), Pro Edu (free for educational), Pro Eval (not for commercial use) and Pro (commercial use) and it is Open Source for not commercial use. The first public release was in March 2010, in August 2012 started the ROS integration and at the beginning of 2013 it became Open Source. V-REP offers fast prototyping and verification, fast algorithm development, robotics related education, remote monitoring, hardware control, safety monitoring, and product presentation. V-REP can rather be seen as a hybrid simulator that combines kinematics and dynamics in order to obtain the best performance for various simulation scenarios. It is based on two physics engine: Bullet [18] and ODE [19] (the same as Gazebo) and user is free to switch from one to the other at any time.

The integrated development environment is based on a distributed control architecture: each object/model can be individually controlled via an embedded script, a plugin, a ROS node, a remote API client, or a custom solution. Controllers can be written in C/C++, Python, Java, Lua, Matlab or Urbi.

V-REP is compatible with Windows, Linux and Mac OS X. Documentation is very good and cover all aspect of the simulator. A lot of tutorials and examples allow users to learn quickly how use it.

C. Comparison

In Table I, a selection of simulators parameters are summarized. The following features are compared:

1) Available licenses(License);
2) Operating system (OS); it describes which OS is supported by the robotic software.
3) Simulator type; it describes if the robotic software provides either a 2-D or 3-D simulation environment.
4) Programming language; it describes the language supported by the robotic software.
5) Year of origin; it specifics the year of commencement of the simulator;
6) Collision Detection;
7) Sensors; it describes the supported sensors. Only the most requested sensors are reported in the table due to space limitations.
8) Graphical User Interface; it describes if it is possible to modify objects and the environment during run-time and/or program functions in an development environment. The graphical user interface does not include windows that open to display the simulation.
9) Portability; “Yes” means that the code written for a simulation is portable to a real robotic platform.
10) Scalability “High” means that the simulator does not require a lot a resource for running complex simulation and that it can simulate more robot simultaneously, “Medium” means that the simulator in some case required a lot resource and “Low” specifics that the simulator can’t simulate multiple robot at the same time;.
11) Real Time; it specifics the number of allowed operation during the simulation. “High” means that you can do a very large number of operation in real time, “Low” means that only a few operation can do in real time.
12) Interfaces; it describes the facility of integration the simulator with another system.
13) Documentation level provided with the simulator (Documentation); it can be “High” or “Low”. “High” means the documentation only provides descriptions of the functions in the robotic software libraries; “Low” means the documentation provides the code for the functions in the robotic software libraries.

14) Tutorial; it describes if examples and a step-by-step guide are provided. “Yes” means that a well defined guide with examples is available; “limited” means a guide exists, but not enough details and examples are provided. Finally, “No” means there is not a useful tutorial and/or examples.
15) Debugging/Logging; describes if the simulator debugging, fault tolerances, play-back, and logging features are provided by the robotic software.

III. Aldebaran NAO

NAO is a humanoid robot produced by Aldebaran Robotics. With 3000 pieces sold to 550 different universities, NAO is the most widely used humanoid robot for academic purposes worldwide.

The robot we used is a H25 v 4.0 version of NAO, it is 58 cm tall with 25 DoF (12 DoFs are embedded in its legs) and an integrated Intel Atom CPU @ 1.6GHz. It can communicate with external systems using a Wi-Fi connection. In addition, the NAO has feedback from of all joints, pressure sensors on its feet, two gyroscopes, and an accelerometer. NAO came with a complete software suite to fully program the humanoid platform, an SDK package makes the developer interact with the NAOqi application programming interface (API).

A ROS [13] package was also published by Humanoid Robots Lab at the Albert-Ludwigs-Unversitaet in Freiburg [22] in order to provide basic functionality to integrate the NAO humanoid robot into ROS. The ROS integration allows us to move simulated robots in the same way we control the real one.

A. Robot model and plugins

The Unified Robot Description Format (URDF) has been used to create an accurate NAO H25 v4.0 model to simulate the physical robot in Gazebo and V-REP. The URDF is the ROS standard way to represent a robot model. This format can be read from the ROS Visualizer (RViz) and any simulator integrated in ROS should be able to import this kind of models.

We based our work on the documentation[1] by Aldebaran Robotics. The virtual model is not manually composed, but we auto-generate it starting from some `xacro` scripts in order to better structure our model. The `xacro` language is based on XML and it allows us to define basic macros to create more complex URDF models.

Four basic macros has been created to build the complete model:

- Joint: defines the model joints;
- Link: defines the model links;
- Visual: defines the model visual meshes;
- Collision: defines the model collision meshes.

this division is very important in order to easily update the model to new robot versions or attach NAO accessories to the main body.

[1] *NAO documentation* http://www.aldebaran-robotics.com/documentation/

All joints are defined using a single macro setting an effort limit of $100N$, a velocity limit of $5rad/s$, a damping factor of $0\frac{N \cdot m \cdot s}{rad}$ and a friction factor of $25N \cdot m$.

For each link composing NAO, we defined the inertial matrices starting from the ones declared from Aldebaran and referring them to the CoM (Center of Mass) using the *Huygens-Steiner* or *Parallel axis* Theorem. In Equation 1 the used formula is explained

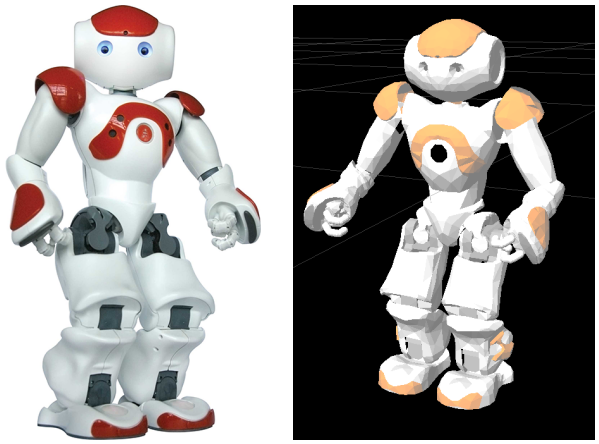
$$\Gamma_{CoM} = \Gamma_{NAO} + m \left\{ \|t_{CoM,NAO}\|^2 I - t_{CoM,NAO} t_{CoM,NAO}^T \right\} \quad (1)$$

were:

- Γ_{CoM} is the calculated inertia matrix referred to the Center of Mass;
- Γ_{NAO} is the original inertia matrix given by Aldebaran Robotics;
- m is the mass of the considered NAO part;
- $t_{CoM,NAO}$ is relative translation between the Center of Mass and the original reference system.

An URDF model already exists in the NAO ROS package from Freiburg University, but inertial matrices are not correctly defined, no realistic meshes has been provided and the xacro scripting only take account of visual and structure separation. The resulting model is very hard to read and modify, consequently we start to building our new model from zero. It is worth to notice that, with no proper inertial matrices, the model coming with the actual ROS package is not suitable for simulation and so no comparison with our model is possible.

Both Gazebo and V-REP are able to import URDF models, we only have to develop the specific plugin. Model and plugins will be released as a Open Source software[2]. In Figure 1 the physical and the simulated[3] robot are shown.



(a) NAO physical robot. (b) NAO URDF model.

Fig. 1. Aldebaran NAO: physical robot (a) and his URDF model (b).

[2] A link to the model and plugins webpages will be inserted in the final version of the paper.

[3] NAO users can download official meshes from the Aldebaran Robotics site. A script will be available to properly split in parts the provided 3D mesh file.

TABLE I
SUMMARY OF THE MAIN FEATURES OF GAZEBO AND V-REP SIMULATORS.

	Gazebo	**V-Rep**
License	Open Source	Open Source for not commercial use
OS	Linux	Linux, Win, MacOSX
Programming Language	C++/Python/Java	C/C++/Python/Java/Matlab/ Urbi
Year of origin	2002	2010
Collision detection	Yes	Yes
Sensors	Laser Range Finders, 2D and RGB-D Cameras, Contact Sensors, IMU, RFID	Proximity, Vision, Force
GUI	Sufficient	Good
Portability	Yes	Yes
Scalability	Medium	High
Real time	Low	High
Interfaces	ROS (Excellent), Player (Excellent)	ROS (Good)
Documentation	Low	High
Tutorial	Yes	High
Debugging/ Logging	Yes	Yes

IV. A CASE STUDY

In order to evaluate our model characteristics on the selected simulators, we tested two concrete, challenging robot tasks, namely walking straight and turning around. The system developed is able to generate suitable movements based on standard motion provided by Aldebaran. The same joint movements could be used as input for both real and simulated robot. Thus, the tasks were also performed using a real NAO platform in order to better compare the results coming from the simulators.

A. Test 1: Straight walk

The first test was based on the standard walking straight command included in Aldebaran NAO Drivers. The robot walked 3 different distances (0.5, 1, and 3 meters) at 3 different velocities (40%, 80%, and 100% of the maximum effort given by NAO motors). Every single trail was repeated 7 times. The real robot, V-REP simulator with both ODE and Bullet engines, and Gazebo simulator were tested. The robot were stopped when the distance theoretically walked reach the goal, the joint positions feedback coming from robot motors were considered to better approximate the measure. On the other hand, the algorithm did not take advantage of any feedback to refine the movements.

The space walked by simulated and real robots was analyzed with respect the expected distance in the xy ground plan. Figure 2 shows the distances measured along the x axis, where x is the walking direction; while Figure 3 describes y as the lateral deviation with respect the straight trajectory.

The typical trajectory followed by the robot is a curve progressively deviating from the straight direction for both simulation and reality. All the tested modalities are able to correctly perform the walk with no falls. Analyzing the collected data, it is easy to see that the real robot exceeded the goals, while simulators usually underestimated the distance.

Looking at the requested task, ODE engine, in particular Gazebo, has performed a great work. The distance walked is about 70-80% of the goal, the deviation is minimal, and the variability between the attempts and the tested velocities is low. On the other hand, the performances are quite poor compared with the real robot trajectory. The difference along the x axis starts from 20-25% to walk 0.5 m at 40% of the maximum effort, and reach 35-40% to walk 3.0 m at 80% of the maximum effort. The gap is even higher along the y axis: the deviation performed by the real robot is up 2 m, while the ODE simulations return substantially zero drift.

The results from Bullet engine are deeply different. Along x axes, the walked distance is 60-65% less then expected at low velocities, while it consistently grows augmenting the speed. The lateral deviation follows the same trend, and it is aligned to the to the real robot drift, at high velocities.

It is also worth to notice that Gazebo presents a great repeatability, while V-REP shows a great variability with both ODE and Bullet engines.

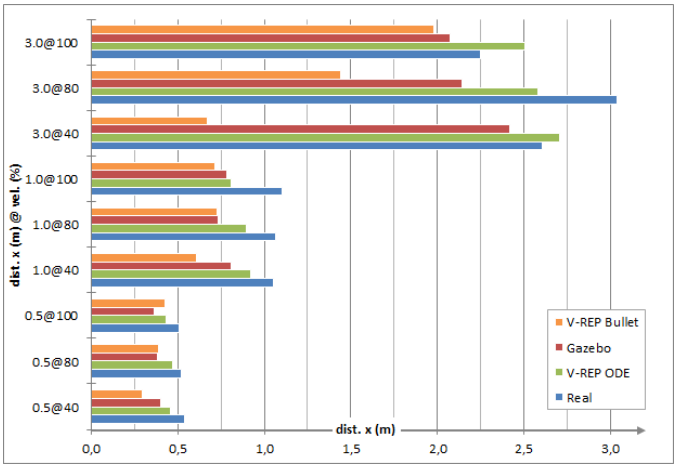


Fig. 2. Results of the task straight walk along x direction. This is not the real distance walked by the robot, but the projection along x.

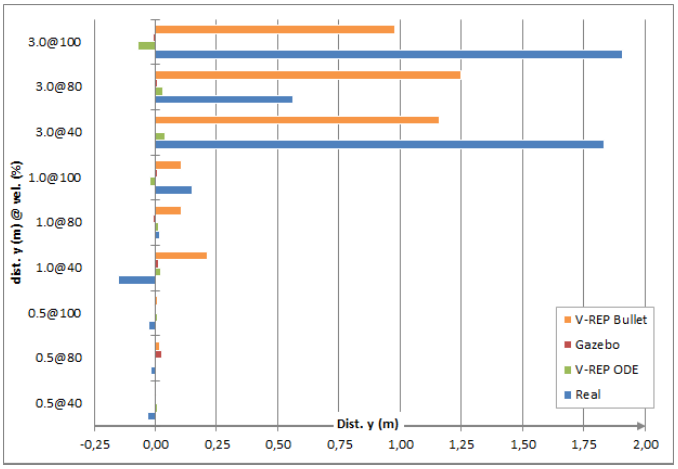


Fig. 3. Results of the task straight walk along y direction. A positive value corresponds to a right deviation of the robot, a negative value corresponds to a left deviation of the robot.

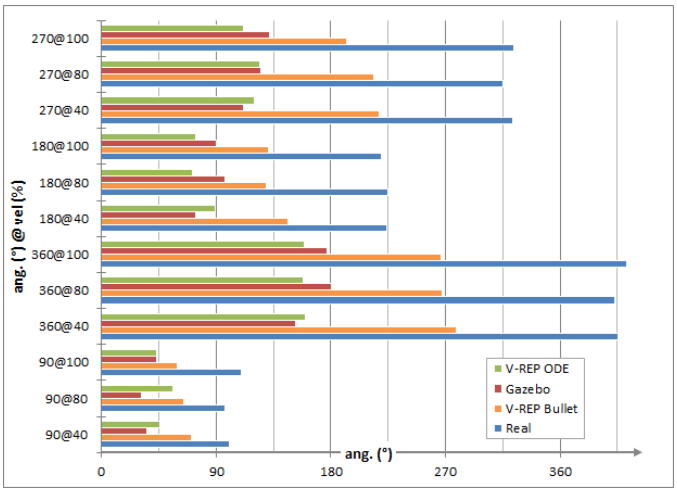


Fig. 4. Result of the task turn around. The absolute value of rotation angle is showed. At 90° and 360° the robot has turned counterclockwise, while at 180° and 270° the robot has turned clockwise.

B. Test 2: Turn around

The second test was based on a standard motion implemented in Aldebaran NAO Drivers to let the robot perform the turning around task. NAO turned of 4 distinct angles (90° and 360° counterclockwise, 180° and 270° clockwise) at 3 different velocities (again 40%, 80%, and 100% of the maximum effort given by NAO motors). As before, each trail were performed 7 times using the four tested modalities: real robot, V-REP (ODE and Bullet) and Gazebo. Again, the robot were stopped when the desired turning angle has been reached. The feedback coming from robot joint positions helped us in the measure approximation, but the motion did not depend from this feedback.

The angle turned by simulated and real robots was analyzed with respect the expected rotation. Figure 4 shows the absolute value of the robot rotation around itself.

The robot turned with no falls and a minimal deviation from its theoretical rotation center in all the considered modalities, so we did not further investigate this parameter. Again, the real robot usually overcame the goals, while simulators highly underestimated the desired rotation.

ODE engine has not performed a great work as in the first task. The angle turned is usually less than 50% with respect to the goal, on the other hand the variability between the attempts and the tested velocities is still low for both Gazebo and V-REP.

The results from Bullet engine are quite similar, the rotation performed is more effective than ODE engine and the angle is 60-65% less then expected. at low velocities, while it consistently grows augmenting the speed. The variability decreases with respect the walking task reaching the ODE engine accuracy.

Gazebo has maintain the great repeatability showed in the first task, V-REP has performed better than walking task, but a certain variability still persists.

V. Conclusion

In this paper a novel virtual model of Aldebaran NAO was presented. We met the purpose of creating a flexible model suitable for simulation of the most diffuse humanoid robot used in scientific research and in soccer competitions.

Our model is able to properly work with different simulators using exactly the same code developed for the real robot. We tested it through two popular service robotics simulators, namely Gazebo and V-REP. The results showed that both simulators are able to correctly perform a standard walk and to make the robot turn around itself without falling.

In this first model version all joints and links were correctly set, but no sensor has been included. As future work we will integrate different kind of sensors to better simulate real situations.

References

[1] B. Browning, and E. Tryzelaar. Übersim: A multi-robot simulator for robot soccer. In *In Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS*, pages 948–949, 2003.

[2] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper. Usarsim: a robot simulator for research and education. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1400–1405, April.

[3] M. Freese, S. Singh, F. Ozaki, and N. Matsuhira. Virtual robot experimentation platform v-rep: a versatile 3d robot simulator. In *Proceedings of the Second international conference on Simulation, modeling, and programming for autonomous robots*, SIMPAR'10, pages 51–62, Berlin, Heidelberg, 2010. Springer-Verlag.

[4] N. Greggio, E. Menegatti, G. Silvestri, and E. Pagello. Simulation of Small Humanoid Robots for Soccer Domain. *Journal of the Franklin Institute*, 346(5):500–519, June 2009.

[5] A. Hornung, A. Dornbush, M. Likhachev, and M. Bennewitz. Anytime Search-Based Footstep Planning with Suboptimality Bounds. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS)*, Osaka, Japan, November 2012.

[6] J. Jackson. Microsoft robotics studio: A technical introduction. *Robotics Automation Magazine, IEEE*, 14(4):82–87, Dec.

[7] S. Kalyanakrishnan, T. Hester, M. Quinlan, Y. Bentor, and P. Stone. Robocup 2009. chapter Three humanoid soccer platforms: comparison and synthesis, pages 140–152. Springer-Verlag, Berlin, Heidelberg, 2010.

[8] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154 vol.3, Sept.-2 Oct.

[9] E. Menegatti, G. Silvestri, E. Pagello, N. Greggio, A. Cisternino, F. Mazzanti, R. Sorbello, and A. Chella. 3d models of humanoid soccer robot in usarsim and robotics studio simulators. *I. J. Humanoid Robotics*, 5(3):523–546, 2008.

[10] O. Michel. Webots: Symbiosis between virtual and real mobile robots. In J.-C. Heudin, editor, *Virtual Worlds*, volume 1434 of *Lecture Notes in Computer Science*, pages 254–263. Springer Berlin Heidelberg, 1998.

[11] L. Mora, K. C. Mendonca, E. Wurtz, and C. Inard. Simspark: An object-oriented environment to predict coupled heat and mass transfers in buildings. In *Proc. 8th Int.l IBPSA Conference*, 2003.

[12] C. Paulo, G. Jos, L. Jos, and M. Paulo. Simtwo realistic simulator: A tool for the development and validation of robot software. *Theory and Applications of Mathematics & Computer Science*, 1(1), 2011.

[13] M. Quigley and K. Conley. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[14] B. P. Gerkey, R. T. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *In Proceedings of the 11th International Conference on Advanced Robotics*, pages 317–323, 2003.

[15] N. Shafii, L. Reis, and R. Rossetti. Two humanoid simulators: Comparison and synthesis. In *Information Systems and Technologies (CISTI), 2011 6th Iberian Conference on*, pages 1–6, June.

[16] A. Hornung, k. M. Wurm, and M. Bennewitz. Humanoid Robot Localization in Complex Indoor Environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2010.

[17] A. Staranowicz and G. L. Mariottini. A survey and comparison of commercial and open-source robotic simulator software. In *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '11, pages 56:1–56:8, New York, NY, USA, 2011. ACM.New York, NY, USA, 2011. ACM.

[18] Bullet Physics Library [online]. Available: http://bulletphysics.org/.

[19] Open Dynamics Engine [online]. Available: http://www.ode.org/.

[20] Object-oriented Graphics Rendering Engine [online]. Available: http://www.ogre3d.org/.

[21] Robot Operating System. [online] Available: http://www.ros.org/.

[22] NAO_robot ROS stack. [online] Available: http://www.ros.org/wiki/nao_robot/.