OpenGRASP: A Toolkit for Robot Grasping Simulation

Beatriz León[1], Stefan Ulbrich[2], Rosen Diankov[5], Gustavo Puche[1], Markus Przybylski[2], Antonio Morales[1], Tamim Asfour[2], Sami Moisio[3], Jeannette Bohg[4], James Kuffner[5], and Rüdiger Dillmann[2,⋆]

[1] Universitat Jaume I, Robotic Intelligence Laboratory, Castellón, Spain
{len,puche,morales}@icc.uji.es
[2] Karlsruhe Institute of Technology, Institute for Anthropomatics, Karlsruhe, Germany
{ulbrich,przybyls,asfour,dillmann}@ira.uka.de
[3] Lappeenranta University of Technology, Centre of Computational Engineering and Integrated Design, Finland
smoisio@lut.fi
[4] Royal Institute of Technology, Computer Vision and Active Vision Laboratory, Stockholm, Sweden
bohg@csc.kth.se
[5] Carnegie Mellon University, Institute for Robotics, USA
{rdiankov,kuffner}@cs.cmu.edu

Abstract. Simulation is essential for different robotic research fields such as mobile robotics, motion planning and grasp planning. For grasping in particular, there are no software simulation packages, which provide a holistic environment that can deal with the variety of aspects associated with this problem. These aspects include development and testing of new algorithms, modeling of the environments and robots, including the modeling of actuators, sensors and contacts. In this paper, we present a new simulation toolkit for grasping and dexterous manipulation called *OpenGRASP* addressing those aspects in addition to extensibility, interoperability and public availability. OpenGRASP is based on a modular architecture, that supports the creation and addition of new functionality and the integration of existing and widely-used technologies and standards. In addition, a designated editor has been created for the generation and migration of such models. We demonstrate the current state of OpenGRASP's development and its application in a grasp evaluation environment.

Keywords: software toolkit, grasping simulation, robot modeling.

1 Introduction and Related Work

Robot simulators have accompanied robotics for a long time and have been an essential tool for the design and programming of industrial robots. Almost all

⋆ The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement n° 215821.

industrial manipulator manufacturers offer simulation packages accompanying their robotic products. These tools allow the users to program and test their applications without using the real hardware or even building it since such tools allow the analysis of behaviours and performance beforehand. In robotics research, simulators have an important role in the development and demonstration of algorithms and techniques in areas such as path planning, grasp planning, mobile robot navigation, and others. There are several reasons for the use of robot simulations. First, they allow exhaustive testing and tuning of mechanisms and algorithms under varying environmental conditions. Second, they avoid the use, or misuse, and wearing of complex and expensive robot systems. Finally, simulation software is cheaper than real hardware.

Often, simulation tools used to support research are specifically developed for particular experiments. However, there have been some successful attempts to develop general robot simulators specifically for mobile robotics.

Stage and Gazebo are respectively 2D and 3D simulator back-ends for Player [1,4], which is a widely used free software robot interface. Gazebo [2] in particular, implements a 3D multi-robot simulator which includes dynamics for outdoor environments. It implements several robot models, actuators and sensors. USARSim [5] has a similar functionality. It is a free mobile robot simulator based on a gaming engine. Microsoft provides its Robotics Studio [3], a framework for robot programming that includes a visual simulation environment. OpenHRP [7] is an open software platform with various modules for humanoid robot systems such as a dynamics simulator, a view simulator, motion controllers and motion planners. OpenHRP is integrated with CORBA, with each module, including the dynamics simulator implemented as a CORBA server. Commercial options include Webots [6], a product which has been widely successful in educational settings.

The variety of simulation tools for robotic grasping is rather limited. The most renowned and prominent one is *GraspIt!*, a grasping simulation environment [9]. *GraspIt!* includes models of several popular robot hands, implements the dynamics of contacting bodies, includes a grasp planner for a Barrett hand and has recently included a simple model of the human hand. However, *GraspIt!* has several limitations. Its rather monolithic and less modular architecture makes it difficult to improve, add functionality and integrate with other tools and frameworks. In addition, it does not provide a convenient Application Programming Interface (API), which allows script programming. Furthermore, it does not include sensor simulation.

Another existing and publicly available software framework is OpenRAVE, the Open Robotics and Animation Virtual Environment [10]. It has been designed as an open architecture targeting a simple integration of simulation, visualization, planning, scripting and control of robot systems. It has a modular design, which allows its extension and further development by other users. Regarding robot grasping simulation, it provides similar functionality to GraspIt! and various path planning components. It provides the models of several robot arms and hands and allows the integration of new ones. It also enables the development of virtual controllers for such models.

In this paper we introduce a simulation toolkit specifically focused on robot grasping and manipulation. The principal design issues have been extensibility, interoperability and public availability (see Sec. 2). The core of the toolkit is an improved version of OpenRAVE, which has been enhanced with a Robot Editor and the adoption of the COLLADA file format [12] and Physics Abstraction Layer (PAL) [11] to enable standardization and flexibility (see Sec. 3). Finally, several applications which demonstrate the utility of the toolkit are presented (see Sec. 4).

This work is the result of a team funded by the European Commission within the project GRASP [21]. Within the project the toolkit is used as the main tool for reasoning and prediction of object properties as well as task constraints of manipulation and grasping activities executed by robots. Thus, it is a key tool for constructing a world model and understanding the robot environment.

2 Requirements for a Grasp Simulator

From a scientific point of view, a novel simulator for robot grasping should provide primarily a realistic simulation of dynamic properties of, at least, rigid objects and advanced contact models including soft contacts and deformable surfaces. From a practical and user point of view, it should include the models of the most popular robot hands, and provide the possibility of easily creating and adding new ones. Furthermore, it should provide realistic simulations of real actuators and sensors, which would enable the use of the same API as in their real counterparts. Regarding sensors, a grasping simulator has to provide simulations of specific grasping sensors, such as force/torque, contact and tactile. Finally, it should provide a rich and detailed visualization of simulations.

With respect to software engineering, a novel robot grasping simulator must be implemented in a modular way that enables on the one hand, an easy extension by both developers and users and on the other hand, the integration with commonly-used robotics software frameworks. In order to increase its opportunities of being adopted and used in the scientific community, the simulator should be open source and make use of open standards for file formats and other representations. In addition, the simulator should have appropriate tools to import/export robot and object models to/from standard representations.

To our best knowledge, none of the existing simulation tools and software packages fulfill all of these requirements. Therefore, we present a software toolkit for grasping simulation *OpenGRASP* [8], which is built on top of OpenRAVE to meet the requirements discussed above.

3 Toolkit Description

In order to develop a tool that meets the requirements listed in the previous section, we adopted a basic practical principle: *Do not reinvent the wheel.* This means, first to review the existing software paying special attention to those that already meet part of the requirements and second to make use of existing open and widely-available software packages and standards.

After a wide review of existing simulators, physics engines, 3D render engines, and CAD 3D modelers, we concluded that OpenRAVE is the tool that most closely meets our requirements. So our efforts have focus on improving and extending OpenRAVE capabilities and features towards the realization of an advanced grasping simulator. In addition, we developed a robot editor allowing to create and modify new robot models. In the following sections we describe these extensions in more detail.

3.1 OpenRAVE Architecture

OpenRAVE is a planning architecture developed at the Carnegie Mellon University Robotics Institute. It is designed for autonomous robot applications and consists of three layers: a core, a plugins layer for interfacing to other libraries, and scripting interfaces for easier access to functions (see Fig.1).

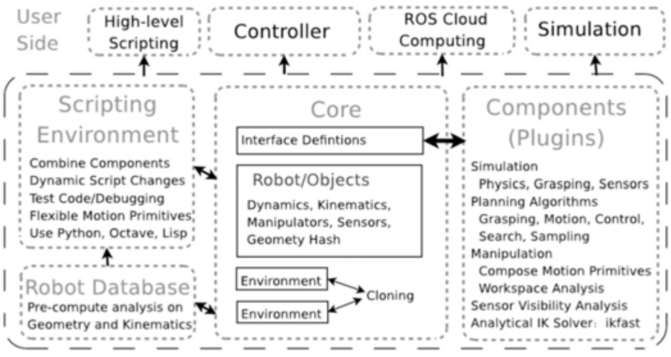

Fig. 1. OpenRAVE Architecture (Picture reproduced from [22])

The Scripting Layer enables network scripting environments like Octave, Matlab and Python to communicate with the core layer in order to control the robot and the environment. It is possible to send commands to change any aspect of the environment, read any of its information, move real robots, or change physics/collision libraries. The scripts also enable the control of multiple OpenRAVE instances across the network, thus allowing different users to independently see and interact with the environment.

OpenRAVE is designed as a plugin-based architecture, which allows the creation of new components to continuously improve its original specifications. Each plugin is an implementation of a standard interface that can be loaded dynamically without the need to recompile the core. Following this design, different plugins can be created for components such as sensors, planners, controllers or physics engines. The core layer communicates with the hardware through the plugins using more appropriate robotics packages such as Player and the Robot Operating System (ROS) [30].

A GUI can be optionally attached to provide a 3D visualization of the environment. It periodically queries the core to update the world's view and allows the user to change the position of the objects in the scene. As viewers are provided through plugins, a single OpenRAVE instance can allow multiple viewers to communicate with multiple copies of the environment. A more detailed description of the OpenRAVE architecture can be found in [10,22].

Although many plugins are already implemented to provide basic functionality, the current grasp simulation functionality has several shortcomings. In order to make OpenRAVE suitable for our purposes, we require:

– Implementation of plugins for specific sensors used to improve the grasping capabilities of the robot.
– Implementation of more physics engines and collision checkers that help to compare and improve simulation performance.
– Implementation of a standard plugin interface for a basic actuator and implementations of different motors. This would allow us to accurately simulate the robot's arm and hand articulations.

We have taken these considerations into account in our toolkit design. We have developed two new sensor plugins to be used mainly for anthropomorphic robot hands. One is a tactile sensor, commonly used in finger tips such as in the Barrett hand, which detects and calculates the forces on the predefined sensory area and returns them as an array. The other is a force sensor, placed for example in the wrist, to measure the forces applied while grasping.

Additionally, as models for actuators were not included in OpenRAVE, we have developed a new plugin interface called ActuatorBase. Using this interface, we implemented a new plugin to simulate the motor of the arm and hands joints which can be controlled using angles, velocities or voltages.

We have created a model of the Schunk PG70 parallel jaw gripper using the tactile sensor plugin to simulate the Weiss Robotics sensor (DSA 9205) attached to each finger. We have also provided the Gripper Sensor plugin which returns the distance between fingers, their current and velocity. Finally, a model of the gripper actuator was also included which can control the velocity of the fingers and the maximum current applied.

In order to use different physics engines, we have implemented a plugin which makes use of the physics abstraction layer (PAL), which is addressed in more detail in the following section.

Visualisation is an important part of the simulation. At the moment OpenRAVE uses Coin3D/Qt to render the environment, but we are extending it to communicate with Blender given that our RobotEditor (see Section 3.4) is developed on top of it. Because both Blender and OpenRAVE provide a Python API, it is possible to use Blender as a front-end for not just visualization, but also for calling planners, controlling robots, and editing robot geometry.

3.2 Physics Simulation

Nowadays, there exist many available physics engines, both commercial and open-source. Some of them are high-precision engines that require higher computational power while others sacrifice this accuracy to work in real time. The methods they use to simulate physics are also different. Some of them use penalty methods, some rely on physical laws using constraint equations, and others use methods based on impulses.

None of these engines are perfect, they all have advantages and disadvantages which makes it very difficult to decide which one to use for a simulator. It basically depends on what we want to simulate in addition to what the application of the simulator will be.

The Physics Abstraction Layer (PAL) [11] is a software package created by Adrian Boing that saves us from having to decide at the start what physics engine to use. This layer provides an interface to a number of different physics engines allowing us to dynamically switch between them. This functionality adds incredible flexibility to our simulator offering us the possibility to, depending on our specific environment and use, decide which engine would provide us the best performance [14]. Using their interface, it is also possible test and compare our own engines with existing ones.

The OpenRAVE Physics Engine interface allows the simulator to run using different engines. It also has an interface to implement different collision checkers. Each one of them has to be created as a plugin, extending either the PhysicsEngineBase or the CollisionCheckerBase class. The basic version of OpenRAVE only offers the ODE (Open Dynamics Engine) implementation within the oderave plugin. We have created a new plugin to use PAL, called palrave. This plugin is able to initialize PAL with the specific engine we want to use, without the need of creating different plugins for each one of them.

3.3 COLLADA File Format for Robot Models

For the storage of models of robot manipulators and hands, we were looking for an open, extensible and already widely accepted file format that supports the definition of at least kinematics and dynamics. This is necessary in order to enable the exchange of robot models between supporting applications, leading to greater flexibility in the selection of appropriate tools. Another important aspect was the ability to convert to and from other formats. Among the large variety of file formats for 3D models, there are only a few that are both public domain and not limited to storing only geometric information. Typically, a simulator environment does not only rely on geometric structures but also, for instance, on information about dynamics, kinematics, sensors and actuators of the robot.

Its acceptance as an industry standard and its wide distribution (3D Studio, Blender, OSG, OGRE, Sony, etc.), in addition to a clear and extensible design, led to the choice of COLLADA [12] as the preferred file format for the simulator. In addition, there are open source frameworks available that facilitate integration into new applications.

Since version 1.5, the standard contains many useful constructs dedicated to describing kinematic chains and dynamics that can be used directly for the description of robot models. COLLADA is an XML-based file format that enables and encourages developers to extend the specification to their needs without having to violate the underlying schema definition.

In order to support specific robot features like sensors and actuators, we have used this mechanism to extend COLLADA partially using the original OpenRAVE file definition. These additions are specific to the simulator and are hidden to all other applications so that compatibility remains guaranteed. So far, basic support for COLLADA import and export has been included in the simulator.

3.4 Robot Editor

With the creation of a simulator for grasping the need also arises for a large data base of geometrical, kinematic and dynamic models of robot arms and manipulators. To fill this gap, the development of a modeling tool, the Robot Editor, has been started. Its main goal is to facilitate modeling and integration of many popular robots. The development is driven by the following key aspects:

- **Geometric modeling:** The creation of new robots models requires a tool that excels in modeling of the geometric components (i.e., meshes).
- **Semantic modeling:** Even more important is the ability to allow the description of semantic properties, such as definitions of kinematic chains, sensors and actuators, or even specify algorithms.
- **Dynamics modeling:** Another necessary aspect is the ability to define physical attributes of the robot's elements. At the moment, the focus lies on the dynamics of rigid bodies.
- **Conversion:** Robot models usually come in a variety of different file formats. The modeling tool needs to be capable of processing these formats and converting them into the COLLADA standard. GraspIt! files in particular, being an already widely-used standard with many conform models available, should be readily usable by the simulator.

To our knowledge, there is no existing solution openly available that could meet all these requirements. Therefore, we decided to develop a new modeling tool based on available open source software. The conceptual design of the Robot Editor hence relies on two techniques: on the one hand the open data format COLLADA and on the other hand on the open source project Blender [15]. Blender is a very versatile, powerful and extensible 3D editor that has been chosen because of its convenient 3D modeling capabilities and the built-in support for many CAD file formats and rigid body kinematics. Furthermore, it can be easily extended via a Python scripting interface and offers high-quality ray tracing.

Blender itself, however, lacks the functionality and the appropriate interface for the convenient definition of robot components. In addition, conversions between certain file formats need to be improved or implemented, namely the import of the COLLADA format and GraspIt! robot models.

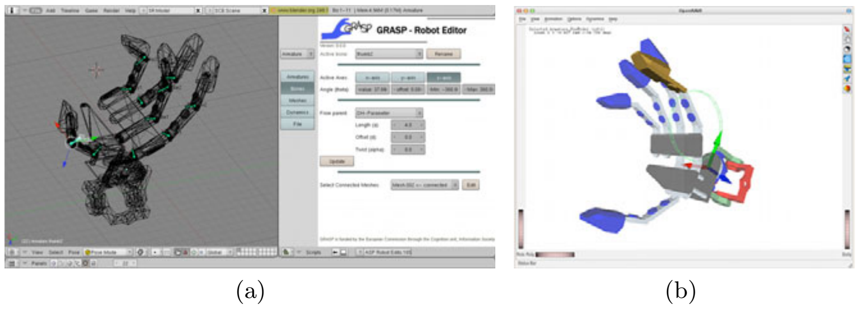

Fig. 2. Modeling the Karlsruhe anthropomorphic robot hand. a) The user interface of the Robot Editor and b) Screenshot of the complete model in the simulator.

The scripting interface mechanism mentioned above allowed us to build the modeling tool on top of Blender. On the scripting level, one gains access to all relevant data structures. The robot model can be augmented by the required data structures and conserved within the native file format. The scripting mechanism also allows the creation of user-interface that is highly specialized for use in robotics (see Fig. 2(a)). For instance, you can define kinematics via Denavit-Hartenberg parameters. In the long run, the Robot Editor will provide interfaces for essential robotics algorithms, such as the computation of dynamics characteristics from the geometric meshes and conversions between kinematics representations. Adjacency information of joints and the impact of joint movements to the robot are additional computational information which, in the future, will be useful for developers' planning algorithms.

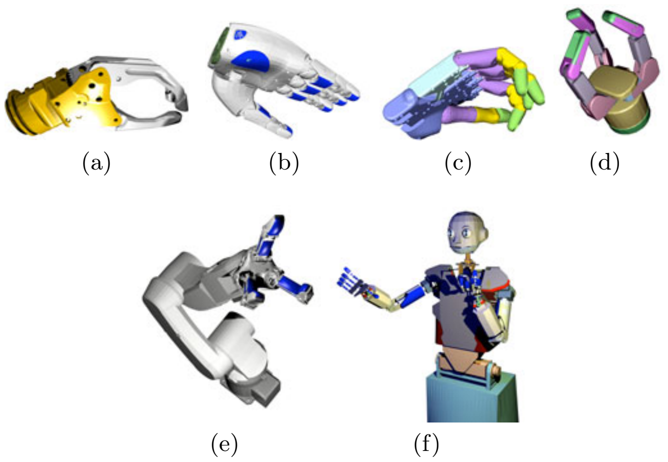

Fig. 3. Different robot models generated with the Robot Editor (ray-traced images). (a) A myoelectric upper extremity prostheses of Otto Bock, the Schunk (b)SAH and (c)SDH hands, (d) the Shadow hand, (e) the Barrett hand and the Kuka KR5 sixx R850 arm, and (f) the humanoid robot ARMAR-III.

In the current Blender version (2.49), COLLADA support [17] is limited to documents in the older specification 1.4 which excludes the newly introduced kinematics and dynamics. The additional data required by the simulator also needs to be included in the resulting document. This led to the further development of COLLADA compatibility which now enables the Robot Editor to create valid documents suitable for simulation. Fig.2(a) shows a functional model of the Karlsruhe anthropomorphic hand [16] modified using the Robot Editor and Fig.2(b) the resulting COLLADA file loaded into the simulator.

Robot Models. As stated in Section 2, it is of great importance to have models of the most popular robot hands included in the toolkit. The modeling capabilities of the Robot Editor already enable quick and easy creation of new models. So far, a selection of robot hands has been transformed into COLLADA 1.5 for use within the simulator (see Fig. 3). In addition to these new models, there are various models available in the older XML file format which is still supported.

4 Applications

4.1 Planning and Grasping

Using the functions provided by OpenRAVE, we can easily build a set of stable grasps and quickly get our robots to manipulate various objects in their environment. Figure 4 shows the grasp simulation process by analyzing the contact points between the robot and the target object. Recently Przybylski et al. [26] used OpenGRASP to develop a new grasp planning method based on the Medial Axis of objects to reduce the search space of candidate grasps. The algorithm exploits structural and symmetry information contained in the Medial Axis in order to reduce the search space for promising candidate grasps.

In order to get the robot to autonomously manipulate objects in the environment, we would need an inverse kinematics solver that can quickly map grasp locations into robot configuration joints. Recently, OpenRAVE started providing an analytical inverse kinematics equation solver called *ikfast*. With it we can generate C++ code that can return all possible IK solutions while simultaneously handling degenerate cases. By combining the automatically generated grasp sets, inverse kinematics solvers and planners, robots developed in our RobotEditor can manipulate everyday objects.

Grasping Known Objects. Using the planning and grasping capabilities of the simulator, we have successfully grasped known objects in the real world with the robotic platform consisting of the Karlsruhe Humanoid Head (see [27,28]) and a Kuka KR5 sixx R850 arm equipped with a Schunk Dextrous hand 2.0.

A complete description of known objects is provided by the KIT object model database [29]. The object description consists of geometry described by a mesh and physical properties like mass. Using the grasp simulation process, different grasp hypotheses are tried out off-line. Only those that result on force closure are then saved to a data base. Using an active vision system [23] in combination

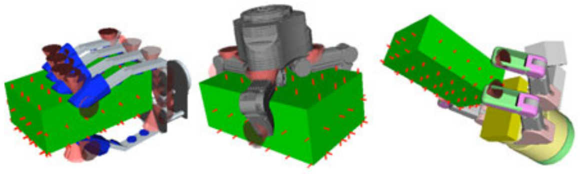

Fig. 4. Grasping simulated for several robot hands

with rigid point set registration [24], the robot can explore the current scene and recognize where and in which pose these known objects are (see an example in Fig. 5). This information is sent to OpenGRASP which reproduces the state of the environment. Using the planning plugins, the selected force closure grasps are executed. The ones that collide with the environment or that are not reachable given the current configuration of the robot are discarded.

Grasping Unknown Objects. We have also used OpenGRASP to manipulate unknown objects with the above mentioned robotic platform. This task is more complex since no information about the objects is available. Thus, the grasp simulation process cannot be executed off-line. The robot uses the above mentioned active vision system [23] to explore the scene and segment unknown objects. From the reconstructed stereo point cloud, a mesh is created as an approximation of the object's geometry. This mesh and its location are sent to the simulator which uses an available library for decomposing it into a reduced number of convex hulls to simplify its collision detection. Given this mesh, there is an unlimited number of grasp configurations that can be applied to the object and tested for force closure. For reducing the search space and thereby online processing time, we apply a grasp point detection method [25] to the segmented object. The simulator then tries only grasp configurations with the fingers of the robotic hand being aligned with the grasp points. Those that are in force closure are saved in a database before being executed with the planner to check for collisions and reachability. An example of the successful grasping of an unknown object is shown in Fig. 6.

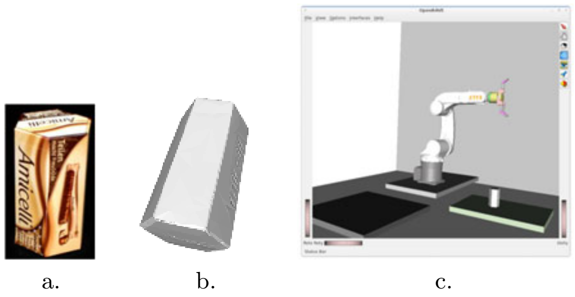

Fig. 5. Grasping a known object: a) real object, b) mesh geometry and c) example of a scene visualization after its recognition

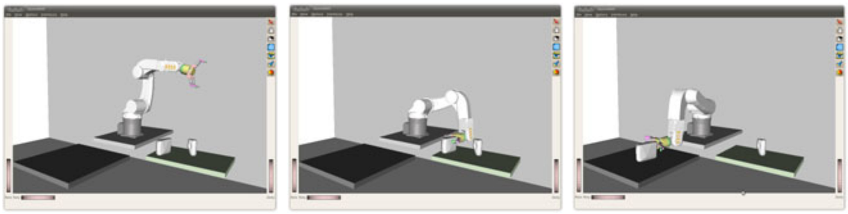

Fig. 6. A successful grasp of an unknown object with the Kuka KR5 R850 arm and a Barret hand

5 Conclusion and Future Work

In this paper we have presented a fully operational simulation toolkit for robot grasping and manipulation. Its main design principles are extensibility, interoperability and public availability. In its development we have used existing and widely-available components to ensure its standardization and easy adoption. We have also emphasised providing additional tools and features that provide users with a fast start to enhance utility through features such as the robot editor based on Blender, COLLADA file format, a Physics Abstraction Library, and models of existing robot hands. The utility of OpenGRASP is demonstrated through a series of applications cases.

In future, we are looking to improve the toolkit in three directions. First a novel contact modelling for soft contacts and body deformation is being implemented and validated. It will be included as a library to allow the modelling of complex contact interactions. Second, OpenGRASP is being extended to offer full support for ROS thus offering a common control interface for simulated and real robots. Finally, several interfaces are being created in order to offer the user several options to visualize and record the simulated systems.

References

1. Gerkey, B., Vaughan, R., Howard, A.: The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In: 11th International Conference on Advanced Robotics (ICAR 2003), Coimbra, Portugal, pp. 317–323 (2003)
2. Gazebo, `http://playerstage.sourceforge.net/index.php?src=gazebo`
3. Microsoft Robotics Studio, `http://msdn.microsoft.com/robotics`
4. The Player Project, `http://playerstage.sourceforge.net/wiki/Main/_Page`
5. USARSim, `http://sourceforge.net/projects/usarsim/`
6. Webots, `http://www.cyberbotics.com/`
7. OpenHRP, `http://www.is.aist.go.jp/humanoid/openhrp/`
8. OpenGRASP, `http://opengrasp.sourceforge.net/`
9. Miller, A., Allen, P.: GraspIt!: A versatile simulator for robotic grasping. IEEE Robotics & Automation Magazine 11, 110–122 (2004)
10. Diankov, R., Kuffner, J.: OpenRAVE: A Planning Architecture for Autonomous Robotics. Technical report, Robotics Institute, Pittsburgh, PA (2008)
11. PAL (Physics Abstraction Layer), `http://www.adrianboeing.com/pal`

12. COLLADA, http://collada.org
13. Khronos Group, http://www.khronos.org/
14. Boeing, A., Bräunl, T.: Evaluation of real-time physics simulation systems. In: 5th international Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia (GRAPHITE 2007), Perth, Australia, pp. 281–288 (2007)
15. Blender, http://Blender.org
16. Gaiser, I., Schulz, S., Kargov, A., Klosek, H., Bierbaum, A., Pylatiuk, C., Oberleand, R., Werner, T., Asfour, T., Bretthauer, G., Dillmann, R.: A new anthropomorphic robotic hand. In: IEEE/RAS International Conference on Humanoid Robots (Humanoids), pp. 418–422 (2008)
17. Illusoft: Blender Collada Plugin, http://colladablender.illusoft.com/cms/
18. Blender - Google Summer of Code (2009), http://www.blendernation.com/blender-google-summer-of-code-2009/
19. OpenCOLLADA, http://www.opencollada.org/
20. Otto Bock, http://www.ottobock.de/cps/rde/xchg/ob_de_de/hs.xsl/384.html
21. GRASP Project, http://www.csc.kth.se/grasp/
22. OpenRAVE website, http://openrave.programmingvision.com/
23. Björkmann, M., Kragic, D.: Active 3D scene segmentation and Detection of Unknown Objects. In: International Conference on Robotics and Automation (ICRA 2010), Anchorage, Alaska, USA (2010)
24. Papazov, C., Burschka, D.: Stochastic Optimization for Rigid Point Set Registration. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Kuno, Y., Wang, J., Pajarola, R., Lindstrom, P., Hinkenjann, A., Encarnação, M.L., Silva, C.T., Coming, D. (eds.) ISVC 2009. LNCS, vol. 5876, pp. 1043–1054. Springer, Heidelberg (2009)
25. Richtsfeld, M., Vincze, M.: Grasping of Unknown Objects from a Table Top. In: ECCV Workshop on 'Vision in Action: Efficient strategies for cognitive agents in complex environments', Marseille, France (2008)
26. Przybylski, M., Asfour, T., Dillmann, R.: Unions of Balls for Shape Approximation in Robot Grasping. To appear in IROS (2010)
27. Asfour, T., Regenstein, K., Azad, P., Schröder, J., Vahrenkamp, N., Dillmann, R.: ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control. In: IEEE/RAS International Conference on Humanoid Robots (Humanoids), pp. 169–175 (2006)
28. Asfour, T., Welke, K., Azad, P., Ude, A., Dillmann, R.: The Karlsruhe Humanoid Head. In: IEEE/RAS International Conference on Humanoid Robots (Humanoids), pp. 447–453 (2008)
29. Kasper, A., Becher, R., Steinhaus, P., Dillmann, R.: Developing and Analyzing Intuitive Modes for Interactive Object Modeling. In: International Conference on Multimodal Interfaces (2007), http://i61www.ira.uka.de/ObjectModels
30. ROS, http://www.ros.org/wiki/