The work described in this paper was partially conducted within the EU Cognitive Systems project GRASP (FP7-215821) funded by the European Commission.
The biggest remaining problem is that grasping and grasp related manipulation is investigated within very heterogeneous environments, that is, on different robots using different middleware and programming languages. Therefore, an interface for the creation of well-defined experiments that can be performed under equal and stable conditions is presented in this paper. The vision driving this work is to encourage scientists world-wide working on the field of robotic grasping and grasp-related manipulation to participate in the process of creating a benchmark suite and feeding the results into a database. This database helps to evaluate many of the algorithmic components developed in their research field, to compete with them, and certainly will also facilitate the choice of the right components when designing new systems. This interface is designed to be extendable and open comes as a part of the open-source OpenGRASP toolkit [37] that builds on top of the OpenRAVE simulator [35]. It consists of four parts:

1) a Python-based interface that glues the simulator to evaluated algorithmic component,
2) a web service collecting and providing results of various benchmarks run by various groups,
3) a set of unified kitchen centered real-life objects
4) and a simulated environment that features a humanoid robot acting in a kitchen.

This architecture will be presented in the remainder of the document. The next section will present the components of the OpenGRASP toolkit including the new benchmark architecture and the GRASP model data-base. Afterwards, the organization of the benchmark architecture will be explained in detail and then, in section IV, two exemplary implementations (one for grasp and one for motion planning) and their results will be presented.

II. THE OPENGRASP TOOLKIT

In robotics, simulation of robotic systems is an essential component in design and planning, and many industrial robot manufactures provide simulators for their robots. The OpenGRASP toolkit [37] is a new simulation environment that is dedicated to grasping. It allows the development and testing of new grasp-related algorithms as well as the modeling of new robots. The simulations are carried out within an improved version of the OpenRAVE simulator [35], which has been enhanced with extended sensor models, interchangeable physics engines and a tool for the creation of new robot models. With OpenGRASP, many of the building blocks required for the benchmarking environment are already available. The following sections highlight some of the most important features.

A. OpenGRASP Robot Editor

Based on the open source 3D modeling tool Blender.org [39], the OpenGRASP Robot Editor is not directly integrated into the simulator itself (see Fig. 1). It allows the convenient creation of new robot models and the conversion from other file formats, and offers a scientific user interface that gives easy access to the many features of the underlying comprehensive modeling software. The key aspects of this software are:

- **geometric modeling:** The creation of new robots models requires a tool that excels in modeling of the geometric components (i.e., meshes),
- **semantic modeling:** The ability to allow the description of semantic properties, such as definitions of kinematic chains, sensors and actuators, or even specify algorithms,
- **dynamics modeling:** Definition of 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 and have to be converted first before they can be loaded into the editor.

For the storage of the models, an open, extensible and already widely accepted file format, which supports the definition of at least kinematics and dynamics, has been chosen. 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. Due to its acceptance as an industry standard, the wide distribution, the now native support for kinematics, and a clear and extensible design, COLLADA in version 1.5 [40] has been selected as the preferred file format in OpenGRASP. At the time of writing, the OpenGRASP Robot Editor produces valid COLLADA documents and experimental support for the import has been added recently. Special annotations to the file that are processed by the OpenRAVE simulator have also been supported.

Having employed this editor, a great number of robot and hand models have already been created so far. Among them are the humanoid ARMAR-III [14], the PA-10 robot, the shadow hand, the Schunk SAH and SDH hands, just to name a few. All of these models can be applied within the simulation and are available for selection within the configuration of the benchmarks.
B. Physics Simulation

The Physics Abstraction Layer (PAL) [41] is a software package created by Adrian Boing that renders the most common physics engines interchangeable. It is an abstraction layer that provides an interface to a number of different physics engines which allows to dynamically switch between them. This functionality adds even more flexibility to the OpenGRASP simulator, offering the possibility to choose the engine with the best performance [42], depending on the specific environment and task. Using this interface, it becomes also possible benchmark the different engines.

The OpenRAVE Physics Engine interface allows the simulator to apply different engines and different collision checkers. OpenGRASP replaces the basic physics engine in OpenRAVE, which is limited to offer an ODE\(^1\) interface, by a new plugin that encapsulates PAL. It is capable of initializing PAL with a specific engine and, thus, eliminates the need to create different plugins.

C. Tactile Sensors

A new tactile sensor plugin provides simulation models of the Weiss Robotics DSA 9330 and DSA 9335 tactile matrix sensor modules [43]. The model simulates the mapping of contact locations to the sensor matrix cells of the sensing surface. Further, it allows specification of a linear characteristic between contact force and compression ratio of the deformable sensing surface to emulate pressure readings similar to the real sensor.

D. KIT ObjectModels Web Database

The OpenGRASP benchmark features every-day objects that can be found in a human-centered scenario (i.e., a kitchen) [44]. Among them are objects of different shapes and sizes, colors and textures, and different topology and difficulty to grasp (see Fig. 2). All database entries are available as multiple-view stereo images, mesh data and point clouds. These objects can be selected from the benchmark GUI, are then automatically downloaded from the database, and integrated into the benchmark.

E. The Kitchen Scenario

A scenario is required for the grasp benchmarks which reflects a real-life situation which involves a real robot. For this reason, OpenGRASP proposes (and provides) a kitchen environment complete with furniture, realistic objects (see Sec. II-D) and a model of a real humanoid robot [14] with an anthropomorphic full-finger hand (see Fig. 3 for an actual screenshot of the scene). It is suggested that benchmarks use this standardized environment, optionally replacing the robot by their own hardware. However, it will also be possible for the community to define completely new scenarios reflecting different situations and places for new benchmarks.

III. THE OPENGRASP BENCHMARK

The OpenGRASP Benchmark consists of four components:

1) real-life scenarios (for instance, the presented kitchen environment together with the graspable objects and the humanoid robot, see Sec. II-E),
2) a web-service that provides test cases, scenarios, robot models, as well as records the results, if requested. A “high-score” that displays the results of all participants of a benchmark is available on the web server,
3) a control software that communicates with the web-service and controls and monitors the simulation,
4) a software interface that serves as a connection/compatibility layer between the participant’s observed algorithm, the benchmark, and the simulator.

That way, the architecture fulfills an adaption of the model-view-controller design pattern, see Fig. 4.

A. The Controller Software

The preferred way of extending and controlling the OpenGRASP simulation is through the access of OpenRAVE’s Python interface. The controller software is consequently completely written in Python and starts OpenRAVE transparently in the background. On startup, it connects to the web service provided with the OpenGRASP Benchmark and fetches information about all available registered test

---

\(^1\)Open Dynamics Engine
environments, benchmarks, models and user high-scores. A graphical user interface is created by the controller, which allows to select, setup and launch the benchmark (see Fig. 5). All benchmarks are organized in categories according to their domains. For instance, there are categories for grasp manipulations, such as motion planning (see Sec. IV-B) and grasp planning itself (see Sec. IV-A). Each of them can be configured to use one of the standardized and predefined benchmark environments, a robot manipulator, and a set of graspable objects. After a successful test run, the results can be chosen to be uploaded to the web-service for future reference or simply be displayed for personal information.

IV. BENCHMARKS

A. Grasp Planners

The grasp planner benchmark allows planning algorithms to compete against each other. Grasp planning aims at finding poses of the hand relative to the object and vectors of hand joint angles that, together, represent force closure grasps. That is, grasps where the object cannot move inside the hand if external forces and torques are applied. In this benchmark, grasp planning for objects of known shape is presented. The grasp planner presented by Berenson et al. [24] is integrated in OpenRAVE and serves as the reference in the presented example. Its competitor is the grasp planner based on the “medial axis” presented by Przybylski et al. [45], [46].

The two planners use different approaches to generate candidate grasps which are then tested for force closure. The grasp planning method by Berenson [24] uses surface normals of the object as approach directions for the hand toward the object, and a user-defined number of roll angles...
of the hand around the approach direction are tested. On the other hand, the grasp planner presented in [45], [46] generates candidate grasps by analyzing symmetry information of the objects contained in their medial axes.

Both planners test candidate grasps by placing the hand at an initial pose where it collides with the object. Then they retract it along the approach direction until it is no longer in collision with the object. Now the fingers of the hand close around the object. Finally, the contact points between the object and the robot hand are determined, and the force closure score is computed.

In the benchmark, a number of candidate grasps is generated with both methods for the set of objects described in Sec. II-D and two different robot hands: The Barrett hand and the ARMAR-III hand; the complete kitchen scenario is not necessary for this benchmark. The generation of candidate grasps was restricted to grasps where the palm is in direct contact with the object. Results are presented in Tab. I and II. The numbers of generated candidate grasps per object and the percentage of force closure grasps is displayed. The benchmark contains the element which is responsible of the actual evaluation of the results created by the two grasp planners, in terms of the force closure grasp. That way, it is ensured that the same evaluation criterion is applied to both planners.

### B. Motion Planning

This benchmark offers a standarized interface for comparing motion planning algorithms in the context of grasping and manipulation. Here, the focus lies on sampling-based approaches (e.g., RRT [47] or PRM [48]) but an extension to other algorithms, such as potential field or grid-based approaches, is possible. The design of the benchmark allows to include any motion planning library, as long as python calls can be processed. Since most libraries are based on C++, python wrappers (e.g., boost.python [49]) can be used to enable C++ calls from within the benchmarking framework. In the following, this is demonstrated by extending the C++ library Simox [38] so that it can be accessed by python code. To guarantee, that the same data is used for all evaluations, the kinematic definitions as well as the 3D models of the robot, the environment, and the objects are extracted from the openRAVE framework and passed to a Simox import filter.

The results of the benchmarking scene are verified with the underlying OpenRAVE framework, whereas the following metrics are used for evaluation:

1. **Setup:** The time needed to setup the planner, including all necessary steps for loading and data preperation. Usually this step could be performed on robot startup and thus it is not considered as part of the motion planning process.
2. **Planning:** The time needed to perform the motion planning.
3. **Post-Processing:** This measure represents the duration of potential post-processing steps (e.g., path smoothing).
4. **Correctness:** When using sampling-based planners, usually discrete collision detection (DCD) is applied for validating path segments. The required parameter, defining the step size between two samples, directly affects the planning time as well as the correctness of the results. Even with small step sizes there still remains a probability that a collision was missed by the DCD methods. Hence, the correctness of the resulting motion is evaluated by applying DCD methods with a tiny step size parameter. In future releases, continuous collision detection (CCD) methods will be integrated in order to guarantee the correctness of the results [50].
5. **Length:** The length of the resulting path in configuration space. In case grasping or manipulation motions are benchmarked, the length of the end-effector’s motion in workspace is additionally measured.
6. **Clearance:** The average distance of the robot to obstacles during path execution.

In order to offer well-defined scenes for differing grasping and manipulation tasks, several setups are initially provided by the framework which can be extended in further releases. All scenes are related to the humanoid robot ARMAR-III operating in a kitchen environment as described earlier in Sec. II-E:

1. **Standard motion planning:** In this scene, the start and goal configuration of the robot are predefined, so that standard motion planning algorithms (e.g. RRT or PRM) can be evaluated. The joints used for planning

<table>
<thead>
<tr>
<th>Objects</th>
<th>MA-based planner Candidates</th>
<th>PRM</th>
<th>Surface normals planner Candidates</th>
<th>PRM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread box</td>
<td>296</td>
<td>99.0%</td>
<td>842</td>
<td>16.5%</td>
</tr>
<tr>
<td>Prismatic box</td>
<td>516</td>
<td>99.8%</td>
<td>520</td>
<td>43.8%</td>
</tr>
<tr>
<td>Salt box</td>
<td>408</td>
<td>85.8%</td>
<td>408</td>
<td>41.2%</td>
</tr>
<tr>
<td>Salt can</td>
<td>1072</td>
<td>80.0%</td>
<td>616</td>
<td>48.4%</td>
</tr>
<tr>
<td>Detergent</td>
<td>1022</td>
<td>87.6%</td>
<td>928</td>
<td>38.5%</td>
</tr>
<tr>
<td>Spray</td>
<td>590</td>
<td>77.5%</td>
<td>762</td>
<td>29.5%</td>
</tr>
<tr>
<td>White cup</td>
<td>732</td>
<td>81.1%</td>
<td>456</td>
<td>48.2%</td>
</tr>
<tr>
<td>Green cup</td>
<td>514</td>
<td>51.8%</td>
<td>496</td>
<td>52.0%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Objects</th>
<th>MA-based planner Candidates</th>
<th>PRM</th>
<th>Surface normals planner Candidates</th>
<th>PRM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread box</td>
<td>296</td>
<td>66.9%</td>
<td>842</td>
<td>11.8%</td>
</tr>
<tr>
<td>Prismatic box</td>
<td>516</td>
<td>20.2%</td>
<td>520</td>
<td>19.2%</td>
</tr>
<tr>
<td>Salt box</td>
<td>408</td>
<td>94.3%</td>
<td>408</td>
<td>47.3%</td>
</tr>
<tr>
<td>Salt can</td>
<td>1072</td>
<td>91.4%</td>
<td>616</td>
<td>44.2%</td>
</tr>
<tr>
<td>Detergent</td>
<td>1022</td>
<td>49.5%</td>
<td>928</td>
<td>25.8%</td>
</tr>
<tr>
<td>Spray</td>
<td>590</td>
<td>44.1%</td>
<td>762</td>
<td>13.9%</td>
</tr>
<tr>
<td>White cup</td>
<td>732</td>
<td>63.8%</td>
<td>456</td>
<td>37.5%</td>
</tr>
<tr>
<td>Green cup</td>
<td>514</td>
<td>34.8%</td>
<td>496</td>
<td>36.1%</td>
</tr>
</tbody>
</table>
cover the hip yaw joint and all seven DoF of the right arm of ARMAR-III. The setup together with exemplary solution paths can be seen in Fig. 6(a).

2) **IK-based motion planning:** The goal is not defined as a single point in configuration space, but as a grasping pose in workspace related to an object. Hence, two tasks have to be considered: solving the inverse kinematics (IK) problem and finding a collision-free motion towards the IK-solution. The position of the object is randomly sampled in front of the robot, so that no pre-calculated IK-solutions can be used. To solve this scene, either a two step algorithm (at first solve the IK, then plan the motion) or an integrated approach as the IK-RRT [51] can be used.

3) **Clean the Table:** This is the most challenging setup, where multiple objects are located on a table and the goal is to clean the table by transporting the objects to another area on the sideboard (see Fig. 6(b)). To plan the sequence of actions, multiple sub-tasks have to be solved, such as deciding which object should be grasped, finding IK-solutions, and planning collision-free motions for grasping and placing the objects. The considered configuration space covers an arm, the hip and the position and orientation of the robot’s base.

![Fig. 6](http://wwwiaim.ira.uka.de/GraspBenchmark)

(a) The setup of the first planning scene. An planned and a post-processed solution are depicted in blue and green. (b) Exemplary showcase of the Clean the Table scene, that will be offered by the motion planning benchmark. The goal of this benchmark is to transport all objects that are randomly located on the red table to the green sideboard without any collisions.

Currently, the setups are specified as described in this section and interface methods are provided within the benchmarking framework. A reference implementation of the first scene has been evaluated according to the proposed metrics (see Tab. III). It can be seen, that the sampling step size of 0.05 produces several incorrect results after post-processing. This is caused by the shortcut algorithm, which tends to generate motions that come close to obstacles and hence the probability of an undetected collision increases due to the applied discrete collision detection (DCD) methods. Further information about the parameter setup can be found online\(^2\).

In the near future we will upload more evaluations based on Simox, and thus serving reference benchmarks as a basis for comparison of different approaches for motion planning.

<table>
<thead>
<tr>
<th>Average time</th>
<th>Solution</th>
<th>Planned</th>
<th>Postprocessed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup</td>
<td>1.44s</td>
<td>98.1%</td>
<td>76.92%</td>
</tr>
<tr>
<td>Planning</td>
<td>0.41s</td>
<td>Length</td>
<td>6.73</td>
</tr>
<tr>
<td>Postprocessing</td>
<td>1.17s</td>
<td>Clearance</td>
<td>111.12 mm</td>
</tr>
</tbody>
</table>

### V. Conclusion and Future Work

In this work, the development of a new environment for benchmarking was presented which is integrated in the OpenGRASP toolkit. It will provide a complete tool chain for the integration of various grasp and grasp-related manipulation algorithms, including many robot models and a tool to design new ones, a database containing standardized graspable items, an real-life kitchen environment featuring a complete humanoid robot with anthropomorphic hands. Its individual components were described in detail. The benchmarks are configured and launched using a control program with a graphical user interface that connects to a web service that administers the available benchmarks, scenarios, models and a benchmarking “high-score” that records the participants’ results.

OpenGRASP Benchmark is intended as a first step in the direction of an open platform for comparative analysis of algorithmic technologies related to grasping and grasp manipulation. The project will be heavily depending on the robotics community, and further development of test cases and scenarios will rely on suggestions and input of fellow robot scientist. This is why, at the time of writing, there still exist only the two presented benchmarks: grasp planning and motion planning. Further advances will probably occur in the form of the design and definition of additional benchmarks and scenarios. Areas likely to be covered are hereby grasping and re-grasping, tactile exploration, pre-/post-grasp manipulations, pick and place actions and of course the design of many more daily-life scenarios.

### References


---

\(^2\)Address: http://wwwiaim.ira.uka.de/GraspBenchmark


