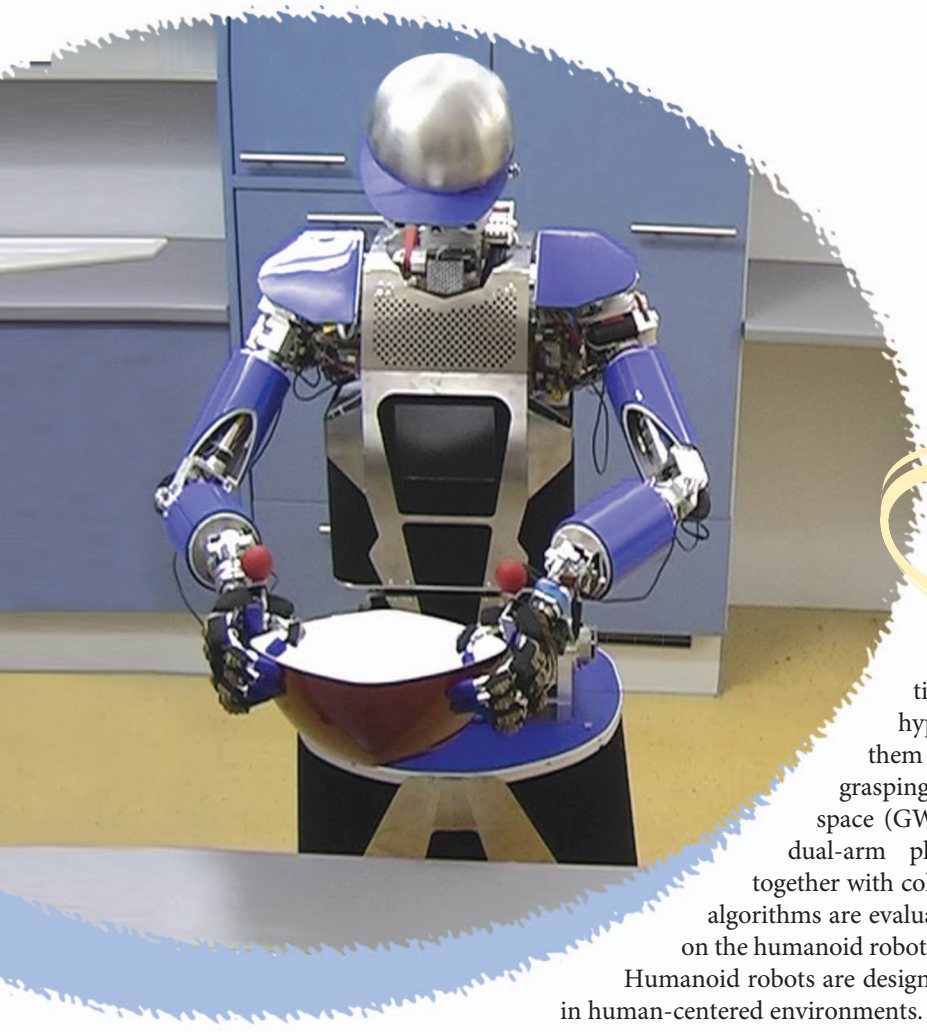
By Nikolaus Vahrenkamp, Tamim Asfour, and Rüdiger Dillmann



Humanoid Robot ARMAR-III

In this work, we present an integrated approach for planning collision-free grasping motions. Therefore, rapidly exploring random tree (RRT)-based algorithms are used to build a tree of reachable and collision-free configurations. During tree generation, both grasp hypotheses and approach movements toward them are computed. The quality of reachable grasping poses is evaluated using grasp wrench space (GWS) analysis. We present an extension to a dual-arm planner that generates bimanual grasps together with collision-free dual-arm grasping motions. The algorithms are evaluated with different setups in simulation and on the humanoid robot ARMAR-III (Figure 1).

Humanoid robots are designed to assist people in daily life and to work in human-centered environments. In contrast to industrial applications, where the environment is structured to the needs of the robot, humanoids must be able to operate autonomously in nonartificial surroundings. Thereby, one essential ability is to grasp a completely known object for which an internal representation is stored in a database (e.g., information about the shape, weight, associated actions, or feasible grasps). Furthermore, the robot should be able to grasp objects for which the internal representation is incomplete as a result of inaccurate perception or uncertainties resulting in an incomplete knowledge base.

The task of grasping an object induces several subtasks that have to be solved, such as searching for a feasible grasping pose, solving the inverse kinematics (IK) problem, and finding a collision-free grasping trajectory. In this article, all these problems with the algorithms proposed are solved with a probabilistic planning approach based on RRT [1].

Simultaneous Grasp and Motion Planning

Digital Object Identifier 10.1109/MRA.2012.2192171
Date of publication: 17 May 2012

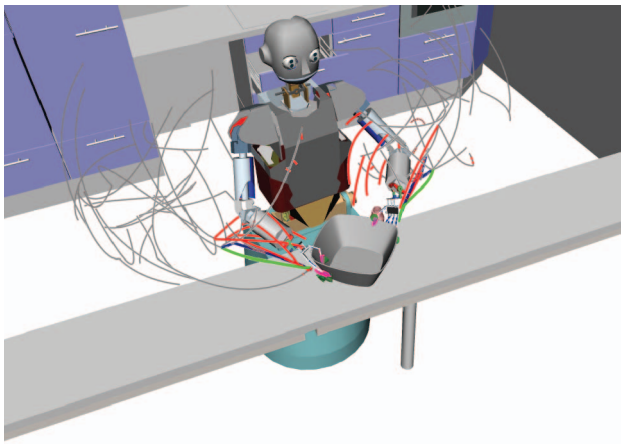


Figure 1. An example of a bimanual grasping trajectory.

The Grasp–RRT planning algorithm combines the computation of feasible and reachable grasps with the search for collision-free motions. This online search for feasible grasping configurations has several advantages compared with classical approaches:

- No precalculated grasping data are needed since grasping hypotheses are generated and evaluated online.
- Hence, the search for feasible grasps is not limited to a potentially incomplete set of offline-generated grasps.
- Online-generated requirements or constraints, which have to be met by the grasping configuration, can be implicitly considered.
- The online search for feasible grasps is focused on reachable configurations; thus, the computation of grasping poses is only performed for positions that can be reached by the robot.
- Complete object knowledge is not needed. An approximated three-dimensional (3-D) object model can be used to search grasping poses online.

The proposed Grasp–RRT planner was originally presented in [2], where we used an efficient grasp measurement based on a low-dimensional space related to the impact forces resulting from grasping contacts. In this work, we use a more common approach of analyzing the GWS for determining the quality of a grasping hypothesis. In addition, we present a more comprehensive evaluation, including the growth in grasp quality over time, and a comparative study that compares the algorithm to classical approaches.

Related Work

Planning collision-free motions for robots with a high number of degrees of freedom (DoF) is known to be a P-space hard problem [3]. Hence, complete algorithms will suffer from low performance, mainly caused by the complex task of computing the free space, C_{free}, the part of the configuration space (C-space) whose configurations do not lead to collisions in the workspace. Probabilistic algorithms may be used to avoid the time-consuming computation of the boundary of C_{free}. RRT-based approaches are widely used in the context of planning reaching and grasping motions for humanoid robots. The general theory for planning collision-free motions with RRT methods can be found in [1] or [4].

Planning grasping motions with predefined sets of grasping poses is discussed in [5]–[8]. These approaches use offline calculated grasping poses for which the IK solutions are searched during the planning process. Stilman considers the IK solutions as constraints that limit the operational space, e.g., while opening doors or transporting objects [9]. When using grasp planners (e.g., see [6] and [10]), the grasping information is computed in an offline step, and the data are stored in a database for use during online search. These grasp planners strongly rely on quality metrics, which are intended to approximate the quality of a grasping hypothesis. Evaluation of an object grasp by a multifingered robot hand has been a major topic in robotics for years. A basic quality criterion for grasp evaluation is the force closure test, first proposed by Lakshminarayana in 1978 [11]. Another approach for evaluating grasp quality is based on the computation of the wrench space, formed by the contact points between the hand and object, also called GWS. Based on the GWS, a score is introduced in [12], which approximates the GWS by a convex hull and tries to fit in the largest wrench space sphere. In [13], the concept of object wrench space (OWS) is proposed to represent the optimal grasp in wrench space by applying forces on numerous points distributed along the object's surface. The OWS is scaled to fit within the GWS, leading to a score in the form of a scaling factor. The authors in [14] propose a task-dependent wrench space where the complexity of calculating the OWS is reduced by approximating it by an ellipsoid.

Several approaches use heuristics to generate grasp candidates based on the object's known geometry. Miller et al. [15] manually decomposes objects into boxes, spheres, cylinders, and cones to plan grasps on the individual primitives. Huebner et al. [16] performed shape approximation using only minimum-volume bounding boxes. The object's medial axis is used by Przybylski et al. [17] to improve the shape-approximation accuracy.

Goldfeder et al. [18] present algorithms to automatically build a database of stable grasps for numerous objects and their application, resulting in "The Columbia Grasp Database." A related approach was used by Xue et al. [19] for automatic grasp planning. The results have been published in the Karlsruhe Institute of Technology object models database, which can be accessed online [20].

Rosell et al. [21] consider observations of humans to capture the human hand workspace that is reduced by determining the most relevant principal motion directions. This allows a mapping to the workspace of a robotic hand, which can be efficiently used to plan hand-arm grasping movements.

Planning dual-arm motions is addressed in [8], where collision-free motions for two end effectors are planned using RRT-based algorithms for bimanual grasping or regrasping

actions. In [22], object-specific task maps are used to simultaneously plan collision-free reaching and grasping motions. The proposed motion optimization scheme uses analytic gradients to jointly optimize the motion costs and choice of the grasp on the manifold of valid grasps.

Integrated Grasp and Motion Planning

In this section, the Grasp–RRT planner and the required components, such as the definition of an end effector, generation of approach movements, and algorithms for measuring the grasp quality, are presented.

Fundamentals

The process of planning collision-free grasping motions can be divided into offline and online phases. During offline processing, a representation of the object is built (e.g., a 3-D model), and further preprocessing steps can be performed. When using a classical approach for planning collision-free grasping motions, a set G of potential grasping configurations is built and stored in a database during an offline phase (e.g., see [6]). As shown in Figure 2(a), the grasp set G and the current object location are used during online processing to select a reachable grasp $g \in G$.

By applying the grasp-specific object-hand relation, the workspace target pose $p_g \in \mathrm{SE}(3)$ can be determined and passed to an IK solver. Note that there is no guarantee at this point that a collision-free motion exists, which brings the end effector from the current position to the selected grasping pose, p_g. Usually, heuristics are used to select a feasible grasp, and, in case no collision-free motion is found in the later processing steps, different grasps must be tested.

The IK solver is used to compute a goal configuration, q_{goal}, which is passed to the motion-planning algorithm together with the current robot configuration, q_{start}. As the IK problem may have an infinite number of solutions, the selection of one specific IK solution q_{goal} could lead to a situation for which no collision-free path from q_{start} to q_{goal} exists. This cannot be detected directly at this step since solving the problem of the existence of a collision-free path in C-space is as hard as solving the motion-planning problem itself [1]. Again, heuristics may be used to select an IK solution for which it can be assumed that a solution exists [6]. Finally, motion-planning algorithms, such as RRT-based approaches, can be used to search a collision-free motion.

In contrast to these stepwise approaches, the proposed Grasp–RRT algorithm integrates the generation of grasping hypotheses, selection of feasible grasps, IK solving, and searching for collision-free motions. An overview of the proposed planner is shown in Figure 2(b). The only information that is needed by the planner is the current robot configuration q_{start}, the object's workspace location $p_o \in \mathrm{SE}(3)$, and access to a spatial representation of the target object. There is no explicit definition for a target configuration since the goal is derived from a feasible grasp that is calculated during the planning process. In Figure 3, a simplified two-dimensional (2-D) example is shown to compare the behavior of three approaches for planning

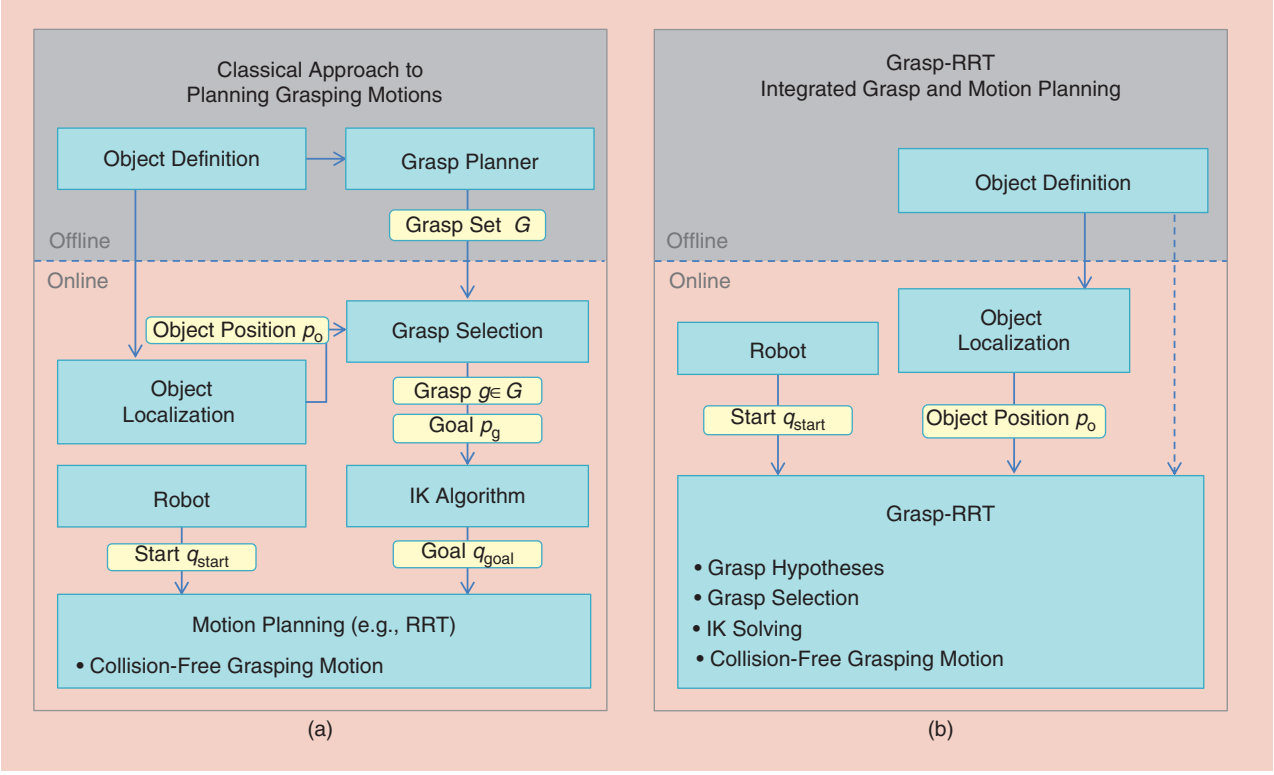


Figure 2. A comparison of (a) the classical approach for planning grasping motions and (b) the Grasp–RRT algorithm.

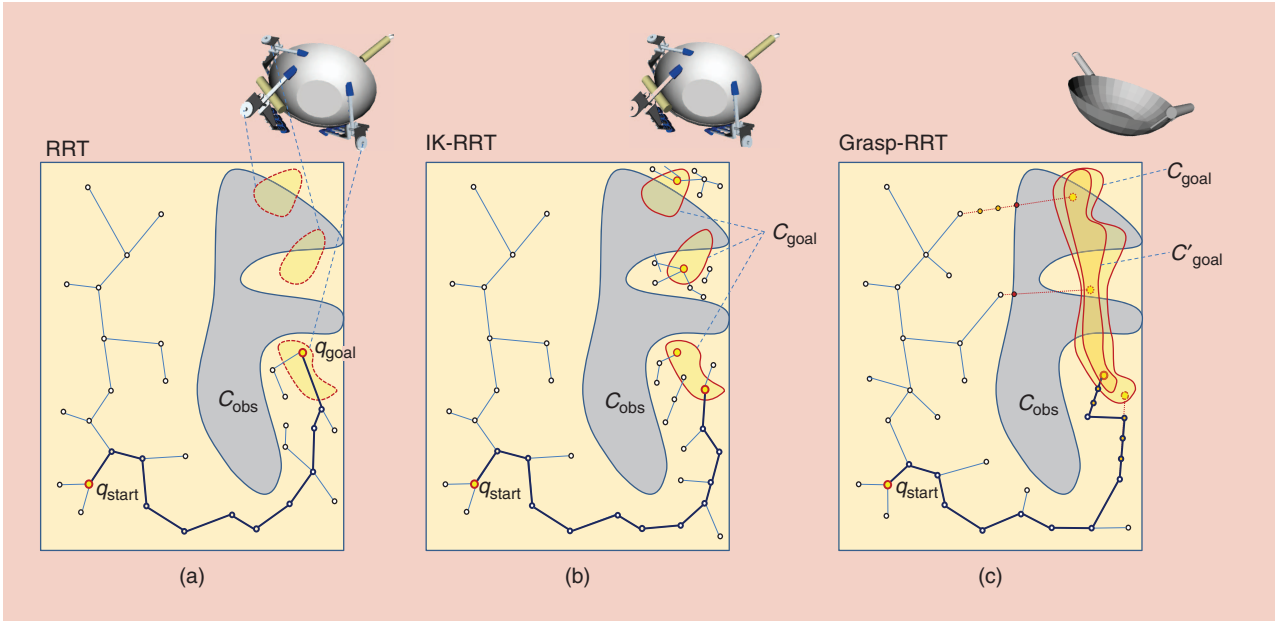


Figure 3. The behavior of three approaches for planning grasping motions illustrated by a simplified 2-D example. (a) A stepwise planning approach is used where a grasp is selected, the IK problem is solved, and a collision-free motion is planned with the RRT algorithm. (b) The IK-RRT planner is depicted [8]. (c) The Grasp–RRT planner does not rely on any predefined grasping configurations. Instead, approach movements are generated during tree expansion to guide the growth in the direction of C_{goal}, the region of all configurations that result in force-closure grasps. The goal region C'_{goal} may be smaller when task-specific constraints or additional grasp quality evaluations are required.

grasping motions. In Figure 3(a), a stepwise RRT-based approach that relies on an offline-generated grasp set is shown. As explained earlier, the selected grasp and computed IK solution specify the target configuration q_{goal} that is used for motion planning. The IK-RRT approach does not use one target configuration q_{goal}, but a goal region C_{goal}, induced by a set of offline-generated grasps, defines all possible targets. While the RRT is extended, C_{goal} is sampled by efficient IK solvers to build new seeds for multiple backward trees. If one of the backward trees can be connected to the forward tree, a collision-free solution is found, which implicitly defines a grasp and an IK solution. Figure 3(b) illustrates the IK-RRT approach [8] that is able to handle the complete set of IK solutions, which are induced by a set of precomputed grasps. Since this set is sampled during planning to generate new seeds for backward search trees, no explicit representation of C_{goal} is needed. If one of the backward trees can be connected to q_{start}, a collision-free solution, together with the corresponding grasp and IK solution, is found. Figure 3(c) depicts the Grasp–RRT planner. Here, the goal region C_{goal} is implicitly defined by the set of all force closure grasps that can be applied to the object with the given end effector. Again, the goal region must not be known explicitly since the Grasp–RRT planner generates approach movements to guide the tree expansion in the direction of C_{goal}. Furthermore, higher requirements on grasp quality and task-specific constraints can be taken into account, resulting in a smaller goal region C'_{goal}.

The Grasp–RRT Planner

The main loop of the proposed Grasp–RRT approach is presented in Algorithm 1. The planner is initialized with the root configuration q_{start} and p_o, the six-dimensional (6-D) pose of the object to be grasped. Starting from q_{start}, RRT-based extension methods are used to build a tree of collision-free and reachable configurations. This part of the Grasp–RRT planner is similar to unidirectional RRT approaches, where sampling-based extensions are performed to cover the free space, C_{free}. In addition, approach motions along with grasping hypothesis are generated occasionally (see Algorithm 2) until a collision-free grasping motion is determined.

To produce appealing solution trajectories, the result is finally smoothed with path-pruning techniques.

Algorithm 1. ***GraspRRT*** (q_{start}, p_o)

```
1  RRT.AddConfiguration(q_start)
2  while (!TimeOut()) do
3  |  ExtendRandomly(RRT)
4  |  if (rand()) then
5  |  |  n_grasp ← TryGraspObject(RRT, p_o)
6  |  |  if (n_grasp) then
7  |  |  |  return BuildSolution(n_grasp)
8  |  |  end
9  |  end
10 end
```

Approach Movements and Grasping Hypothesis

As shown in Algorithm 1, approach movements are generated occasionally to test whether a collision-free grasping motion can be determined. Therefore, the following steps are performed:

- select an RRT node $n_{\texttt{Approach}}$ as an initial configuration for performing the approach movement (see Algorithm 2, line 1)
- based on $n_{\texttt{Approach}}$, a virtual target pose p_{grasp} is computed in the workspace (see Algorithm 2, line 2)
- by using the pseudoinverse Jacobian, the end effector is moved toward p_{grasp} as far as possible while adding intermediate configurations to the RRT (see Algorithm 2, line 3 and Algorithm 3)
- the resulting grasping pose is evaluated by closing the fingers, determining the contacts and performing GWS analysis (see Algorithm 2, lines 4 and 5).

Algorithm 2. ***TryGraspObject*** (RRT, p_o)

```
1 n_Approach ← SelectGraspExtensionNode(RRT)
2 p_grasp ← ComputeGraspingPose(n_Approach, p_o)
3 n_grasp ← Approach(RRT, n_Approach, p_grasp)
4 C_g ← ContactPoints(CloseHand(n_grasp))
5 if (GraspEvaluation(C_g) ≥ ϱ_min) then
6 |   return n
7 else
8 |   return NULL
9 end
```

The steps are explained in more detail in the following subsections.

Selecting an RRT Node for Extension

The approach direction, defining the direction of approach movements toward an object, is essential for finding a feasible grasp since, in general, a stable grasp may only be found for a small amount of all possible approach directions. In our case, where an RRT node $n_{\texttt{Approach}}$ has to be selected as a starting point for generating an approach movement, a random-node selection does not respect this fact since the distribution of configurations of the RRT is independent from the 3-D relation between an end effector and object. In contrast, if the distribution of the node selection uniformly covers the approach directions, the search for good grasps benefits from varying relations between an object and end effector.

To encode different approach directions, we propose the use of a data structure, which we call `ApproachSphere` throughout this article. These data structure represents a triangulated surface of a sphere that is located at the object's center of mass. Whenever a new RRT node n is added during the planning loop, the corresponding surface triangle t_n of the `ApproachSphere` is determined by projecting the tool center point (TCP) position onto the sphere [see Figure 4(a)]. Then, n is added to a list of associated RRT nodes of t_n.

When a random RRT node $n_{\texttt{Approach}}$ for grasp testing is selected, 1) one of the available approach directions represented by the triangles of the `ApproachSphere` is randomly chosen and 2) one of the associated nodes is randomly selected. Hence, the distribution of the node selection uniformly covers the possible approach directions (within the limits resulting from the approximation of the sphere). The advantage of selecting RRT nodes in this way can be seen in Figure 4(b), where the state of the `ApproachSphere` after building an RRT is shown. The color intensity of a triangle is proportional to the number of RRT nodes in the direction represented by the triangle. It can be clearly seen that randomly choosing $n_{\texttt{Approach}}$

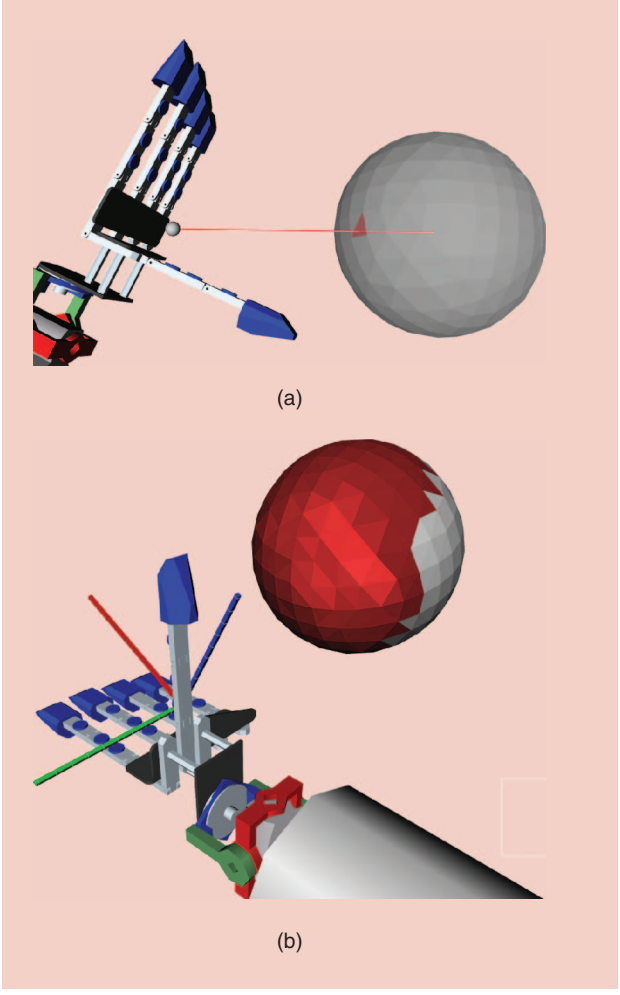


Figure 4. (a) For each node of the RRT, the corresponding surface triangle of the `ApproachSphere` is determined by projecting the TCP position on the sphere's surface. (b) The distribution of approach directions is visualized for a planning task by setting the color intensity so that it is proportional to the number of RRT nodes in the direction of the triangle.

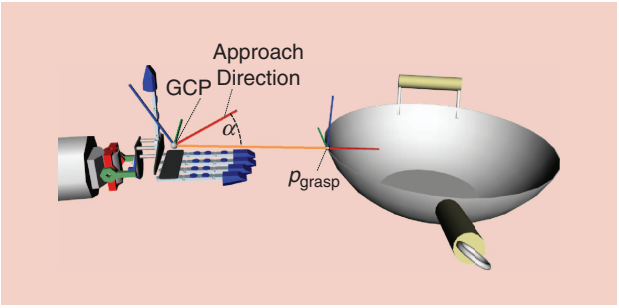


Figure 5. The computation of the grasping pose p_{grasp}.

from all RRT nodes will result in a nonuniform distribution of approach directions.

Computing the Target Pose

For computing the target grasping pose p_{grasp}, a virtual representation of the hand, including a preshape, a grasping point, and an approach direction, is used. Based on the work of [23], the grasp center point (GCP) and the approach direction are defined for the hand that should be used for grasping. The preshape, the definition of GCP, and the approach direction of the anthropomorphic hand that is used in our experiments can be seen in Figure 5. Based on this GCP definition, the target grasping pose $p_{grasp} \in \mathrm{SE}(3)$ is determined by searching for the point $\mathrm{pt}_o \in R^3$ on the object's surface that has the shortest distance to the GCP. pt_o defines the translational part of p_{grasp}, and the rotational component is derived by rotating the coordinate system of the GCP by α so that the approach direction points toward pt_{obj} (see Figure 5).

Approaching the Object

In Algorithm 3, the generation of an approach movement can be seen. Beginning with the selected RRT node $n_{Approach}$, intermediate RRT nodes are created by iteratively moving the end effector toward the grasping pose p_{grasp}. This is done by computing the workspace difference Δ_p between the current TCP pose (stored in the RRT node $n.p$) and the target p_{grasp}. To ensure small steps in the workspace, the method `LimitCartesianStepSize` is applied to limit the resulting displacement. Later, the corresponding movement Δ_q in C-space is computed by using the pseudoinverse Jacobian. The pseudoinverse J^+ is derived via singular value decomposition, although other approaches, which may be more efficient, can be chosen as well. If the resulting configuration $n'.q$, which is computed by applying the movement to the current configuration $n.q$, is in collision or joint limits are violated, the last valid RRT node n is returned. Otherwise, the corresponding TCP position of $n'.q$ is computed, and the RRT is extended by n'. This is performed until the distance to the target pose p_{grasp} falls below $\mathtt{Threshold}_{\mathtt{Cartesean}}$.

Algorithm 3. ***Approach*(*RRT*, $n_{Approach}$, p_{grasp})**

```
n ← n_Approach
repeat
    Δ_p ← p_grasp · (n.p)^-1
    Δ_q ← J+(n.q) · LimitCartesianStepSize(Δ_p)
    n'.q ← n.q + Δ_q
    if (Collision(n'.q) || !InJointLimits(n'.q)) then
        return n
    end
    n'.p ← ForwardKinematics(n'.q)
    RRT.AddNode(n')
    n ← n'
until (Length(Δ_p) < Threshold_Cartesean)
return n
```

Determining the Grasp Quality

To evaluate the quality of the resulting pose n_{grasp}, the fingers are closed and the set C_g of resulting contacts are determined. Closing the hand is performed by iteratively moving the finger joints with small steps until (self-)collisions are detected. Based on this contact information (consisting of positions and normals on the object's surface), a

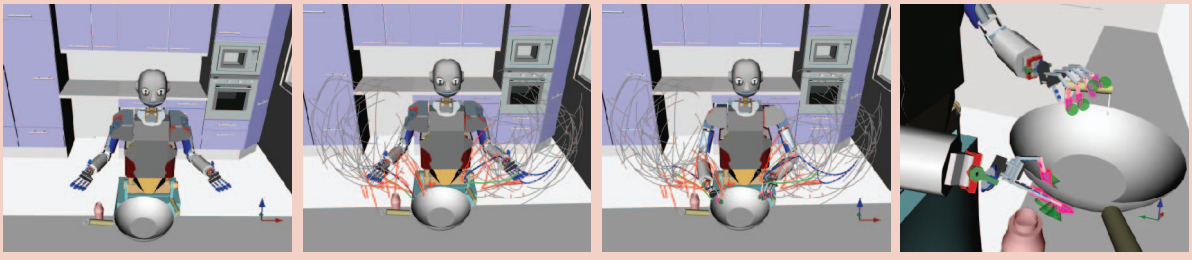


Figure 6. The Bimanual Grasp–RRT planner is used to search a collision-free grasping trajectory for 14 DoF of both arms of ARMAR-III. (a) The initial setup. (b) A workspace visualization of the RRT. (c) The generated grasping configuration. (d) The friction cones are visualized at the contacts.

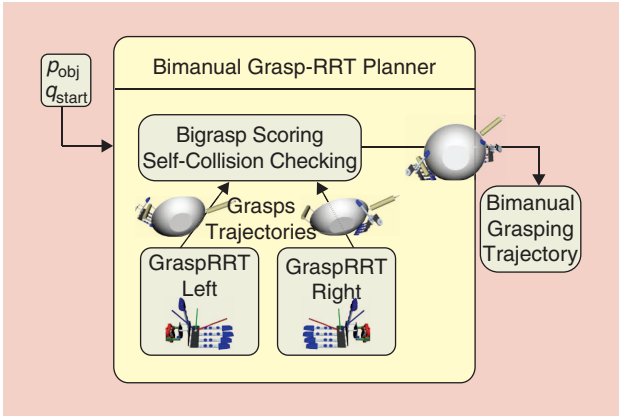


Figure 7. An overview of the Bimanual Grasp–RRT planner.

GWS analysis is performed as described in "Quality of a Grasp."

Planning Bimanual Grasping Motions

When large objects, like the wok in Figure 6, should be grasped by a humanoid robot, both hands are needed for applying a stable grasp. On the basis of the Grasp–RRT planner, introduced in the last section, we propose the Bimanual Grasp–RRT planner, which combines the search for a bimanual feasible grasp along with the search for a collision-free grasping motion for both arms. Since the bimanual planner decouples the search for left and right arm, the approach cannot be used for planning common parts of the robot, as torso or platform. In case such joints should be considered for planning a bimanual motion, a different approach that does not rely on decoupling must be chosen.

Bimanual Grasp–RRT

Figure 7 depicts an overview of the Bimanual Grasp–RRT planner. This planner instantiates two Grasp–RRT planners, one for each end effector. These instances are started in parallel so that the search for feasible grasps is done simultaneously for the left and right hand. Furthermore, they are configured to search and store grasps until the main planner terminates.

The Bimanual Grasp–RRT planner collects the grasps and the corresponding grasping trajectories for the left and right end effector and tries to find a feasible bimanual solution by performing quality evaluations of bimanual grasping combinations. Every time a planner for one end effector reports that a new grasping trajectory was found, all possible bimanual combinations of this grasp together with the already stored grasps of the other hand are built and evaluated as described in the "Evaluation of Bimanual Grasps" section. If the resulting bimanual score is above the threshold $\varrho_{\min}$, the self-collision status of the two pruned grasping trajectories is checked. If no collision was determined, the combined solution for both arms together with the resulting grasping information is returned (see Algorithm 4).

Algorithm 4. ***BimanualGraspRRT*** $(q^l_{start}, q^r_{start}, p_o)$

```
1  GraspRRT_l ← GraspRRTInstance(q^l_start, p_o)
2  GraspRRT_r ← GraspRRTInstance(q^r_start, p_o)
3  GraspRRT_l.start()
4  GraspRRT_r.start()
5  while (!TimeOut()) do
6  |  /* process new results of GraspRRT_l */
7  |  s_l ← GraspRRT_l.GetNewSolution()
8  |  if (s_l) then
9  |  |  Results_l.add(s_l)
10 |  |  foreach (s_r ∈ Results_r) do
11 |  |  |  if (BiGraspEvaluation(s_l, s_r) > ϱ_min &&
   |  |  |  !SelfCollision(s_l, s_r)) then
12 |  |  |  |  GraspRRT_l.stop()
13 |  |  |  |  GraspRRT_r.stop()
14 |  |  |  |  return BuildSolution(s_l, s_r)
15 |  |  |  end
16 |  |  end
17 |  end
18 |  /* process new results of GraspRRT_r */
19 |  ...
20 end
```

Evaluation of Bimanual Grasps

The GWS analysis can be easily applied on bimanual grasping. Considering a robot with two hands, one obtains the two contact point sets C_g^l and C_g^r, for the left and right hand. By analyzing GWS of the united set $C_g' = C_g^l \cup C_g^r$, a quality score of a bimanual grasp can be computed analogously to the single-handed case.

As shown in the "Experiments" section, the performance of a bimanual planner can be increased when the force-closure test is only performed for the resulting bimanual grasp and not for the single-handed grasps since force closure is only needed for the resulting grasp. This leads to a significant improvement in performance, but the appearance of some of the resulting grasping configurations was not appealing. Because of the missing force closure condition for one hand, sometimes degenerated grasps (e.g., just two fingers in contact with the object) were generated, when the corresponding grasp of the other hand was already force closure. These bimanual grasps are evaluated as satisfactory by the GWS algorithm, but the result does not look natural. Hence, the GWS metric is not sufficient for generating anthropomorphic-looking bimanual grasping configurations, and we extended the evaluation of bimanual grasps. An enhancement can be achieved by claiming force closure for both single-handed grasps. Beside the aforementioned drawback of an increased planning time, the number of valid grasps that can be found by such an approach is limited. Usually, there exists bimanual force closure grasps that are composed of two

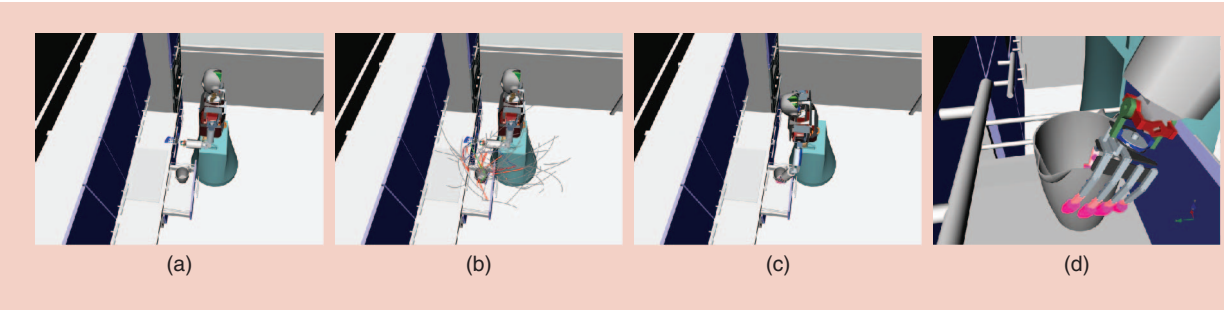


Figure 8. The Grasp–RRT planner is used to search a feasible grasp and a collision-free grasping trajectory for 10 DoF (hip and arm) of ARMAR-III. (a) The planning scene. (b) A visualization of the RRT. (c) The final configuration that was generated during planning. (d) The planned grasp.

single-handed grasps that do not necessarily have to be force closure (one can think of a large vase that is held with both hands from the left and right, where one grasp alone is not force closure, but both grasps together form a force-closure bimanual grasp). Hence, we introduce a simple, but efficient constraint that the grasp of one hand must have at least n_{contact} contacts with the object. By setting n_{contact} to the number of fingers that are considered for planning (five in all our experiments), the results were improved significantly. The resulting algorithm for evaluating a bimanual grasp is shown in Algorithm 5.

Experiments

The following experiments demonstrate that the Grasp–RRT planner is suitable for generating collision-free motions for a wide range of single-handed and bimanual grasping tasks. Therefore, several grasping setups for the humanoid robot ARMAR-III in complex environments are investigated.

Algorithm 5. ***BiGraspEvaluation*** (s_l, s_r)

$C_g^l \leftarrow ContactPoints(s_l)$
$C_g^r \leftarrow ContactPoints(s_r)$
if $(|C_g^l| < n_{contact} n || |C_g^r| < n_{1contact})$ **then**
  return 0
end
$C'_g = C_g^l \cup C_g^r$
return $GraspEvaluation(C'_g)$

Measuring Cup in a Drawer

In this experiment, the humanoid robot ARMAR-III is supposed to grasp a measuring cup, which is located in the drawer of a kitchen. The robot should use three hip and seven arm joints; thus, planning is performed in a ten-dimensional C-space. The setup, depicted in Figure 8, limits the possibility of applying a feasible grasp in a collision-free way since the measuring cup is located near the side walls of the drawer. Nevertheless, the Grasp–RRT algorithm is able to find a suitable grasping pose together with a collision-free trajectory in 7.2 s on an average (measured more than 50 test runs). When higher grasp qualities are requested, the planning time increases as shown in Table 1. The first row shows the results when only force-closure grasps are generated and the grasp quality ϱ is ignored. The second row shows the results that have been measured when setting ϱ_{min} to 0.08, resulting in grasping configurations with higher quality.

In Figure 8(a), the starting configuration is shown, whereas in Figure 8(b), an RRT that was generated during planning is visualized (the visualization of the C-space search tree was built by illustrating the corresponding

Table 1. Performance evaluation.

	Planning Time (s)					
	Total	RRT	Approach Movement	Grasp Evaluation	Number of Approach Trajectories	Number of Grasp Evaluations
Measuring cup (force closure)	7.2	2.5	2.9	1.8	147.7	86.6
Measuring cup ($\varrho_{min} = 0.08$)	13.3	4.6	5.4	3.3	281.3	165.0
Wok (bimanual force closure)	0.6	0.2	0.2	0.2	20.0	10.2
Wok (single arm force closure)	7.6	2.2	2.5	2.8	332.6	142.7
Bowl (bimanual force closure)	0.6	0.2	0.2	0.2	18.7	9.9
Bike (bimanual force closure)	4.0	1.4	0.8	1.7	195.2	81.0

end-effector movements in workspace). The approach movements that were generated during planning are highlighted in red, and the final end effector trajectory is shown in green. Figure 8(b) and (c) shows the final configuration and a narrow view of the computed grasping configuration.

Evaluation of the Bimanual Grasp–RRT Planner

In this simulation experiment, the Bimanual Grasp–RRT planner is queried to find a grasping trajectory for a wok, which is located at the sideboard of the kitchen. The use of both arms of ARMAR-III results in a 14-DoF planning problem, which is solved in 0.6 s on an average. As mentioned in the "Planning Bimanual Grasping Motions" section, the bimanual approach cannot be applied when common joints, such as hip or platform, should be considered for planning. Because of the parallelized search for a left and right trajectory, the planner performs well in this experiment (see Table 1, Row 3). For comparison, we present the average performance when all (single-handed) grasps are tested for force closure in Row 4 of Table 1. As described in the "Evaluation of Bimanual Grasps" section, the force-closure test for single-handed grasps limits the set of bimanual grasps that can be found by the planner. Furthermore, the planning time increased significantly to 7.6 s.

RRT-based approaches are widely used in the context of planning reaching and grasping motions for humanoid robots.

The resulting grasping configuration along with the collision-free trajectories for the left and right arm are shown in Figure 6. Figure 6(a) shows the initial configuration, and an RRT, which was generated during a planning process, is visualized in Figure 6(b). The final configuration and a narrow view of the planned grasp are shown in Figure 6(c) and (d). The contact points of the bimanual grasping configuration are visualized by the corresponding friction cones.

Experiment on the Humanoid Robot ARMAR-III

This experiment is performed online on the humanoid robot ARMAR-III. The Bimanual Grasp–RRT planner is used to search a collision-free trajectory for grasping a bowl on the sideboard with both hands. The ketchup bottle, located near the target object, is limiting the number of feasible grasps for the left hand. Figure 9 shows the results of the planner and the execution of the planned trajectories on the humanoid robot ARMAR-III. The average planning

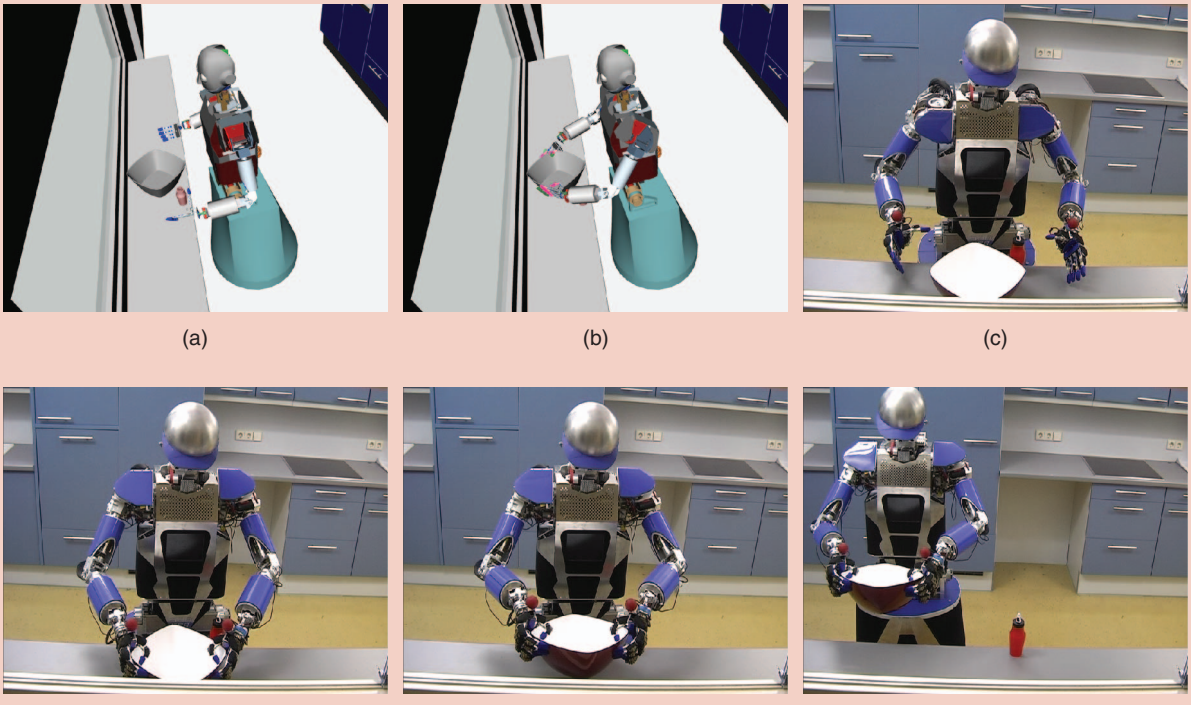


Figure 9. The Bimanual Grasp–RRT planner enables the humanoid robot ARMAR-III to grasp a bowl in the kitchen. (a) and (b) The setup and the generated grasping configuration. (c)–(f) Execution on the robot.

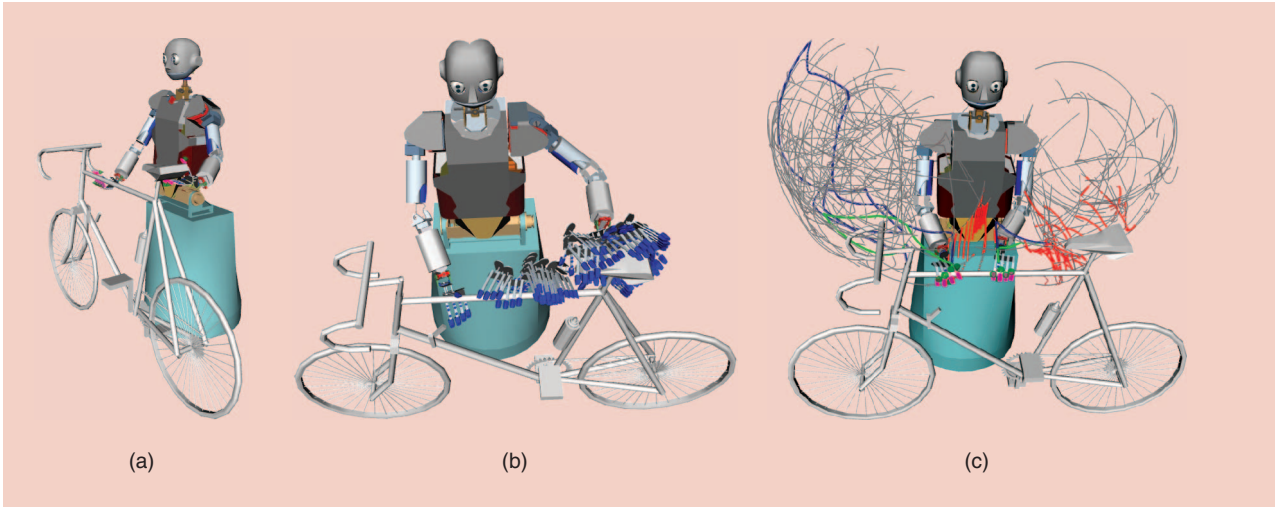


Figure 10. (a) The final grasp ($\varrho = 0.032$) that was generated with the Bimanual Grasp–RRT planner. The friction cones are visualized at the contact points. (b) The visualization shows all grasps that have been generated by the Grasp–RRT planners for the left and right hand. The resulting configuration was evaluated with $\varrho = 0.027$. (c) The workspace visualization of the RRTs generated by the subplanners for the left and right arm. The red parts depict the approach movements that have been built by the Grasp–RRT planners.

time of this experiment was measured with 0.6 s (see Table 1, Row 5).

Bicycle

In this setup, ARMAR-III is supposed to grasp a bicycle with both hands to lift it afterward. As shown in Table 1, the Bimanual Grasp–RRT planner is able to find a force-closure grasp in 4.0 s on an average. An exemplary result can be seen in Figure 10(a). Figure 10(b) shows a visualization of all generated grasping hypothesis that were build by the subplanners of the Bimanual Grasp–RRT algorithm. A workspace visualization of the resulting RRTs generated by both subplanners is shown in Figure 10(c). The approach movements that were generated during planning are highlighted in red, and the solution trajectory is shown in blue (original) and green (pruned).

Grasp Quality

When run time is crucial, the Grasp–RRT planner can be used to quickly find a solution that is not optimal. By increasing the run time, better grasping trajectories can be optionally generated. This allows adapting the quality of the grasp for the demands of online processing, whereas in some cases, a suboptimal solution is preferred over long

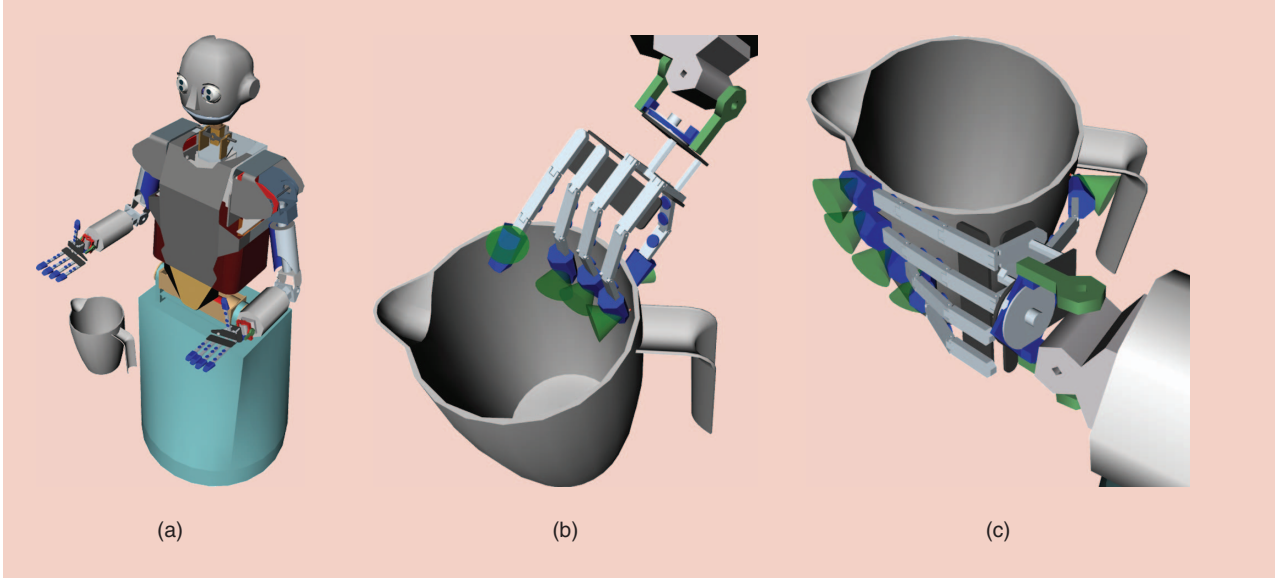


Figure 11. To evaluate the grasp quality over time, a grasping motion for the left and right end effector is searched. Once a solution is found, the Grasp–RRT planner is not stopped. Instead, it tries to find a solution that results in a better grasp evaluation. (a) The setup is depicted and the two resulting grasps are shown in (b) ($\varrho = 0.066$) and (c) ($\varrho = 0.119$).

computation times. In the experiment, depicted in Figure 11, the quality of the best solution is determined over time (see Figure 12).

Performance

The performance of the proposed Grasp–RRT planner in single- and dual-arm planning setups is presented in Figure 13 and Table 1. The run-time analysis has been carried out on a dual-core CPU with 2.7 GHz by averaging 50 test runs. The time spent for the three main parts of the algorithm is distinguished, pointing out that the parameter setup was well balanced since approximately the same amount of time is spent for building the RRT, computing the approach directions, and evaluating the grasping poses. The last two columns of Table 1 show the number of approach trajectories that have been generated and the number of grasp measurements that were calculated during the planning process. These values differ since not all approach trajectories result in a suitable grasping configuration.

Comparative Study

In the following experiment, we simulate the application of the Grasp–RRT approach for grasping an unknown object to compare the proposed approach with classical stepwise algorithms for planning grasping motions [19]. Therefore, we use an imperfect 3-D model of an object that was created with the approaches of [24]. The object is located in front of the robot, and the task is to create a collision-free grasping motion without using any precomputed sets of grasps. Hence, RRT- and IK-RRT-related planning approaches must initially create such grasping information by building a set of potential grasps. This set of grasps is used before (RRT) or during (IK-RRT) planning to solve IK queries for determining target configurations. In this experiment, grasp planning is performed by computing a set of 50 feasible grasps with the wrench-space approach, as it is used in GraspIt! [10]. To allow comparison with the Grasp–RRT approach, the grasp planner that is included in Simox is used so that the performance measurement as well as the resulting grasp quality is based on the same source code. Furthermore, an efficient IK solver must be present for the RRT and IK-RRT planner to determine collision-free IK solutions to one of the planned grasps. In general, the majority of the planned grasps are not reachable by the manipulator, we perform a filter step to generate a subset of reachable grasps. This is done by using precomputed reachability information similar to the approaches presented in [8].

One of the advantages of the Grasp–RRT approach is that neither a dedicated grasp planner nor a robot-specific IK solver is needed. Furthermore, no precomputed reachability information is needed to speed up any IK queries. In the 3-D mesh (Figure 14), the start and final configurations that were generated during planning are depicted.

In Table 2, the results of the RRT, IK-RRT, and Grasp–RRT approach are shown. Initially, the three approaches are compared in a scene where the target object is reachable without difficulty. In this situation, it was sufficient to plan 50 grasps to achieve a collision-free and reachable IK solution for the RRT and IK-RRT algorithms. The average time for this step was measured with 2.7 s. Before planning a collision-free motion with the RRT approach, an IK

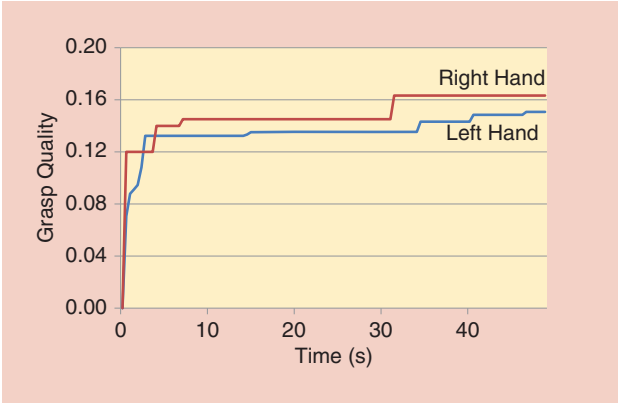


Figure 12. The quality of the resulting grasp increases over time. In this example, the first force-closure grasp was found after 70 ms. When high-quality grasps are needed, several seconds have to be spent until the quality of a resulting grasp cannot be increased significantly.

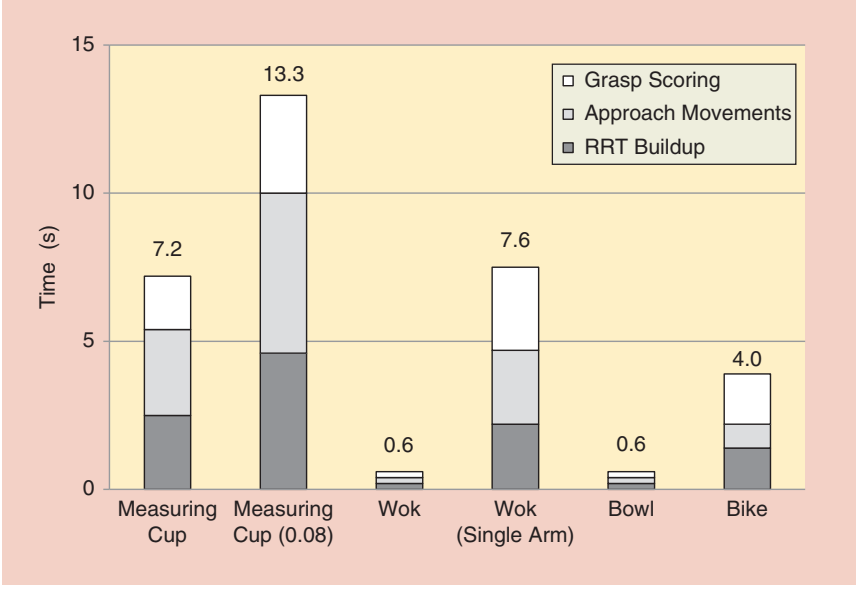


Figure 13. An overview of the average performance measurements.

The approach direction is essential for finding a feasible grasp since a stable grasp may only be found for a small amount of all possible approach directions.

solver for 10 DoF is queried to generate a feasible target configuration. This step, together with filtering the grasps according to their reachability, took 32 ms on average. Because of the simple scene, the motion planning itself could be performed within 75 ms on average, resulting in an overall planning time of 2,834 ms. The IK-solving step can be omitted with the IK-RRT approach, since, here, IK solutions are sampled during motion planning. Nevertheless, a set of grasps has to be generated in advance similar to the RRT approach. Overall, planning with the IK-RRT algorithm took 2,907 ms on average. The Grasp–RRT planner is able to compute a collision-free grasping motion, without the need of precomputing a set of feasible grasps, in 339 ms on average. Rows 3–6 of Table 2 show the results when an additional obstacle is considered [see Figure 14(d)]. As a result of the limited workspace, 200 grasps have to be planned for the RRT and IK-RRT approaches to serve a dense set of grasps that can be used for reliable IK solving. Hence, the overall planning time increased to more than 10 s for the RRT and IK-RRT algorithms. The Grasp–RRT planner was able to solve the problem in 4.2 s on average.

As shown in Table 2, the Grasp–RRT planner performs well when no grasping information is present. In case offline-generated grasping data is stored in a database, the proposed approach introduces some overhead, which is caused by the online generation of grasping hypotheses. Furthermore, the Grasp–RRT approach is a single-directional planner, which is known to be slower compared with bidirectional approaches (such as RRT and IK-RRT). Nevertheless, the Grasp–RRT algorithm is able to find suitable solutions quickly while having the advantage that no IK solver, no reachability information, and no dedicated grasp planner must be present for the robot.

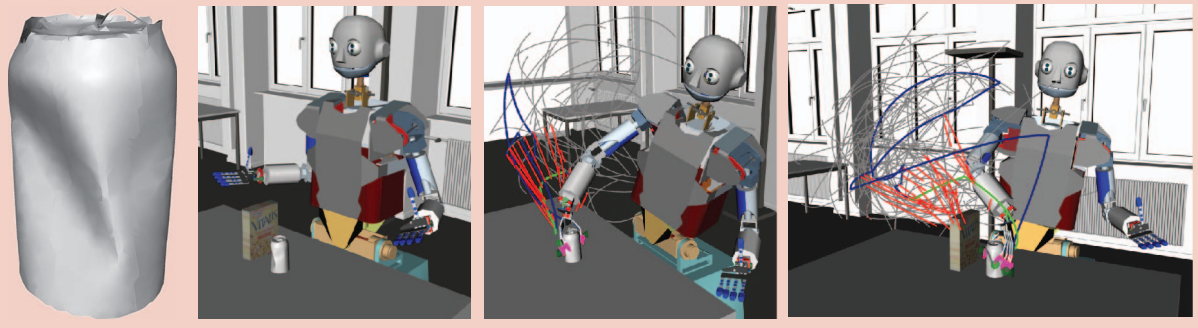


Figure 14. (a), (b) A 3-D model that was generated with shape-retrieval algorithms [24]. The start configuration was used to compare the results of different approaches. (c), (d) Two visualizations of the results of the Grasp–RRT approach. The RRT is visualized in a workspace, whereas the generated approach movements are shown in red and the collision-free grasping motion and its pruned version are depicted in blue and green.

Table 2. A comparison of the different approaches.

	Total	Grasp Planning	IK-Solving	Motion Planning
No obstacle, 50 grasps				
RRT	2,834 ms	2,727 ms	32 ms	75 ms
IK-RRT	2,907 ms	2,727 ms	–	180 ms
Grasp–RRT	339 ms	–	–	339 ms
With obstacles, 200 grasps				
RRT	10,222 ms	9,601 ms	193 ms	428 ms
IK-RRT	10,200 ms	9,601 ms	–	599 ms
Grasp–RRT	4,181 ms	–	–	4,181 ms

Quality of a Grasp

The quality of a grasp is an important aspect for the selection of the optimal candidate from a set of grasps. A common approach to evaluate the quality of grasps is the construction of GWS, which describes a set of all wrenches that can be applied on the grasp contact points. A single wrench is defined as the concatenation of the force and torque vector exerted on a grasp contact point. A frictional point contact $c_i = (p_i, n_i)$ is defined by the position $p_i \in R^3$ on the surface of the object and the corresponding contact normal $n_i \in R^3$.

To evaluate the quality of a grasp, the contacts between end effector and object are used to build friction cones. A friction cone (defined by contact point, contact normal, and material dependent friction coefficient μ) covers all stable contacts for the given material property [14], [26].

Grasp Wrench Space

By approximating the friction cones with m-sided pyramids, force vectors $f_{i,1}, \ldots, f_{i,m}$ are defined, describing the border of the pyramid. For such force vectors, a 6-D force wrench $w_{i,j}$ can be constructed as shown in (S1), whereas $w_{i,j}$ reflects the impacting forces and torques. The torque depends on the position of the contact point with respect to the object's center of mass, p_{com}. Dividing the distance to the center of mass by the object length λ guarantees scale invariance, which can be useful in comparing grasps on different objects [13].

$$w_{i,j} = \begin{pmatrix} f_{i,j} \\ \frac{1}{\lambda}(c_i - p_{\text{com}}) \times f_{i,j} \end{pmatrix}. \tag{S1}$$

Based on (1), the GWS can be built by determining the convex hull over the union of all wrenches described in [27].

A grasp together with the resulting contacts and corresponding friction cones is shown in Figure S1(a). The force subspace of the GWS that was generated by setting the torque components to zero is visualized in Figure S1(b). The torque subspace that was built by disabling the force components is shown in Figure S1(c).

Object Wrench Space

In [13], the OWS was introduced as a representation of all potential grasps that can be applied to an object. The OWS can be computed by applying the GWS computation to a set of virtual contacts which are randomly chosen on the object's surface.

As shown in Figure S2, the resulting force space is represented by a unit sphere, but due to the object dependent definition of λ, the torque components do not form a unit sphere. Hence, the 6-D wrench representation differs between objects.

To avoid the manual definition of a scaling factor for every object, which will limit the significance of the quality score, the

(*continued*)

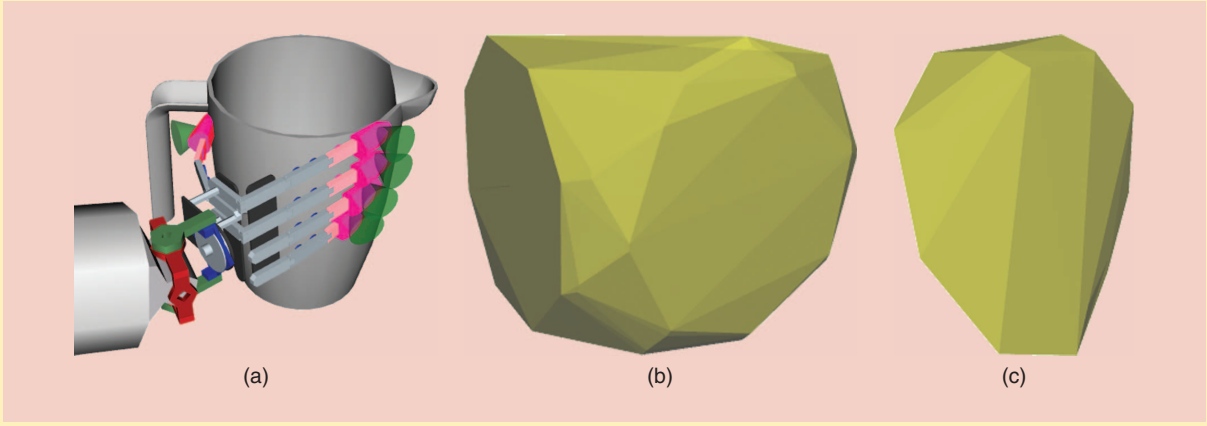


Figure S1. The grasp and the two visualizations of the 6-D GWS that was generated for it. (a) The friction cones are visualized at the corresponding contact points. (b) The subspace representing the forces. (c) The covered torques are visualized.

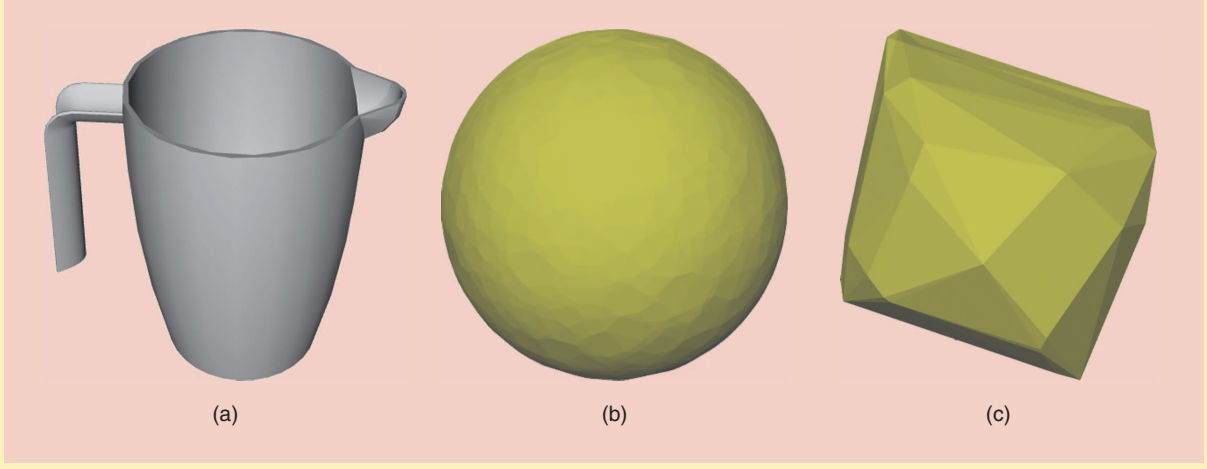


Figure S2. (a) A measuring cup, (b) the projected force, and (c) torque space of the 6-D OWS. The shown OWS subspaces were built by setting the torque and, respectively, the force to zero.

Quality of a Grasp (cont'd)

OWS is analyzed to determine a reference quality value. This ensures that an invariant grasp quality score is computed by the evaluation algorithm.

Grasp Quality

Several approaches to determine the quality of a GWS can be found in the literature, such as the volume of the GWS, the radius of the largest inscribing ball, or the minimum distance ϵ from the origin to the border of GWS. Furthermore, the grasp is stable (or force closure) when the GWS contains the wrench space origin [11], [26].

When doing online grasp planning, computation time is crucial. In an earlier work, we presented a grasp quality measure based on forces, which are adapted to the torques exerted on the object [2]. The performance of the proposed approach was very convincing since the 3-D force space was investigated instead of analyzing 6-D wrench spaces. One drawback was that the magnitude of the forces has to be determined by steepest descent methods, which tend to get stuck in local minima. To increase the reliability of grasp evaluation, we implemented a highly efficient grasp quality measurement component within the GraspStudio library of Simox [28], which offers fast determination of object and GWSs.

This implementation allows efficient computation of online grasp quality evaluation. The grasp quality is determined by performing a test for force closure, followed by computing the minimum distance ϵ from the wrench space origin to the surface of the GWS.

In addition, the OWS ϵ value (denoted with ϵ') is determined in a precomputing step. With ϵ', a reference value for a perfect grasp is computed, which can be used to correlate the quality score of the GWS during online processing. The computation of ϵ' has to be done once for every object and can be stored in a database together with other object properties. Finally, the quality $\varrho \in [0, 1]$ of a grasp is computed as follows:

$$\varrho = \frac{\epsilon}{\epsilon'}. \tag{S2}$$

Note that the grasp quality evaluation component of the Grasp–RRT approach is exchangeable, allowing to incorporate other algorithms for determining grasp quality. Furthermore, we want to explicitly mention that other algorithms for evaluating grasps are known in literature, which may result in a more realistic computation of grasp qualities. An in-depth discussion of GWS analysis can be found in [14].

Conclusions

In this work, a planning approach for computing grasping trajectories was presented. Compared with existing state-of-the-art planners, the proposed Grasp–RRT approach does not rely on any precomputed grasping positions since suitable grasping poses are determined during the planning process. The algorithm integrates the search for solutions of the three main tasks needed for grasping an object: finding a feasible grasp, solving the IK problem, and computing a collision-free trajectory. Since the Grasp–RRT approach does not rely on any precomputed set of grasps, the results are not limited to such a discretization of potential goal configurations. Compared with stepwise approaches, where a grasp is selected from a set of precomputed grasps to compute a specific IK solution that is finally used as a goal configuration for planning a grasping motion, the whole set of potential goal configurations is considered by the Grasp–RRT planner. Hence, the approach can be used without any heuristics for grasp selection or IK solving, and a more general way of generating grasping motions can be achieved.

The Grasp–RRT algorithm is able to find suitable solutions quickly while having the advantage that no IK solver, no reachability information, and no dedicated grasp planner must be present for the robot.

Furthermore, it was shown that the bimanual grasping trajectories can be efficiently planned with the Bimanual Grasp–RRT planner. The approach relies on decomposing the high-dimensional planning problem that arises when considering two arms and two multifingered hands. This is achieved by independently considering the generation of grasping trajectory hypotheses for each end effector. These single-arm motions are investigated to find a feasible combination, resulting in a bimanual configuration that respects the requested grasp quality. Since the decomposed subtasks do not directly depend on each other, parallelized concepts can be used to improve the efficiency.

As shown in the "Experiments" section, collision-free motions for a large variety of single arm and bimanual grasping tasks can be efficiently planned with the Grasp–RRT approach. Depending on the environment and task, the planning time that was measured during our experiments varied from less than a second to 7.6 s. If run time is not crucial, grasps that respect higher-quality demands can be generated by extending the time that is spent for planning.

Further improvements may be achieved by considering constraints within the grasp quality evaluations. These constraints may result from demands of postgrasping actions, such as lifting or object-specific manipulations, which should be performed after grasping. In addition, task-specific constraints can be taken into account to adjust the generation of grasping candidates with respect to a task dependent goal. Furthermore, a local optimization of the calculated grasping trajectory could be applied to locally maximize the quality of the grasping pose. In case of grasping nonconvex objects, a better grasp quality evaluation could be achieved by a hierarchical decomposition in multiple superquadrics, which

can be used to generate a more comprehensive set of approach directions, as introduced in [25].

Acknowledgments

The work described in this article was partially conducted within the German Humanoid Research project SFB588 funded by the German Research Foundation [Deutsche Forschungsgemeinschaft (DFG)] and the European Union (EU) Cognitive Systems project Xperience funded by the EU under grant agreement 270273.

References

[1] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2000, pp. 995–1001.

[2] N. Vahrenkamp, M. Do, T. Asfour, and R. Dillmann, "Integrated grasp and motion planning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Anchorage, May 2010, pp. 2883–2888.

[3] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Proc. IEEE 20th Annual Symp. Foundations of Computer Science (SFCS 1979)*, Washington, DC, 1979, pp. 421–427.

[4] S. M. LaValle, 2006. *Planning Algorithms*, Cambridge, U.K., Cambridge Univ. Press [Online]. Available: http://planning.cs.uiuc.edu/

[5] E. Drumwright and V. Ng-Thow-Hing, "Toward interactive reaching in static environments for humanoid robots," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems (IROS)*, Oct. 2006, pp. 846–851.

[6] D. Berenson and S. S. Srinivasa, "Grasp synthesis in cluttered environments for dexterous hands," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots (Humanoids)*, Dec. 2008, pp. 189–196.

[7] M. V. Weghe, D. Ferguson, and S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots (Humanoids)*, Nov. 2007, pp. 477–482.

[8] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner, and R. Dillmann, "Humanoid motion planning for dual-arm manipulation and re-grasping tasks," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems (IROS)*, St. Louis, Oct. 2009, pp. 2464–2470.

[9] M. Stilman, "Task constrained motion planning in robot joint space," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems (IROS)*, 2007, pp. 3074–3081.

[10] A. T. Miller, "Graspit!: A versatile simulator for robotic grasping," Ph.D. dissertation, Dept. Comput. Sci., Columbia Univ., 2001.

[11] K. Lakshminarayana, "Mechanics of form closure," ASME, NY, Tech. Rep. 78-DET-32, 1978.

[12] D. Kirkpatrick, B. Mishra, and C. Yap, "Quantitative steinitz's theorems with applications to multifingered grasping," in *Proc. 20th ACM Symp. Theory of Computing*, 1990, pp. 341–351.

[13] N. Pollard, "Parallel methods for synthesizing whole-hand grasps from generalized prototypes," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., MIT, 1994.

[14] C. Borst, M. Fischer, and G. Hirzinger, "Grasp planning: How to choose a suitable task wrench space," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2004, pp. 319–325.

[15] A. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Sept. 2003, vol. 2, pp. 1824–1829.

[16] K. Huebner, S. Ruthotto, and D. Kragic, "Minimum volume bounding box decomposition for shape approximation in robot grasping," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2008, pp. 1628–1633.

[17] M. Przybylski, T. Asfour, and R. Dillmann, "Planning grasps for robotic hands using a novel object representation based on the medial axis transform," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems (IROS)*, 2011, pp. 1781–1788.

[18] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, "The columbia grasp database," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2009, pp. 1710–1716.

[19] Z. Xue, A. Kasper, J. Zllner, and R. Dillmann, "An automatic grasp planning system for service robots," in *Proc. 14th Int. Conf. Advanced Robotics (ICAR)*, 2009, pp. 1–6.

[20] A. Kasper, R. Becher, P. Steinhaus, and R. Dillmann, "Developing and analyzing intuitive modes for interactive object modeling," in *Proc. 9th Int. Conf. Multimodal Interfaces,* ser. *(ICMI '07)*, ACM, New York, 2007, pp. 74–81.

[21] J. Rosell, R. Suárez, C. Rosales, and A. Pérez, "Autonomous motion planning of a hand-arm robotic system based on captured human-like hand postures," *Auton. Robots*, vol. 31, pp. 87–102, May 2011.

[22] M. Gienger, M. Toussaint, and C. Goerick, "Task maps in humanoid robot manipulation," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems (IROS)*, 2008, pp. 2758–2764.

[23] T. Asfour, P. Azad, N. Vahrenkamp, K. Regenstein, A. Bierbaum, K. Welke, J. Schroeder, and R. Dillmann, "Toward humanoid manipulation in human-centred environments," *Robot. Auton. Syst.*, vol. 56, no. 1, pp. 54–65, 2008.

[24] D. Gonzalez-Aguirre, J. Hoch, S. Rohl, T. Asfour, E. Bayro-Corrochano, and R. Dillmann, "Towards shape-based visual object categorization for humanoid robots," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2011, pp. 5226–5232.

[25] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof, "Grasp planning via decomposition trees," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2007, pp. 4679–4684.

[26] V.-D. Nguyen, "Constructing force-closure grasps," *Int. J. Robot. Res.*, vol. 7, pp. 3–16, June 1988.

[27] C. Ferrari and J. Canny, "Planning optimal grasps," in *Proc. IEEE Int. Conf. Robotics and Automation*, May 1992, vol. 3, pp. 2290–2295.

[28] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Simox: A simulation and motion planning toolbox for C++," Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, Tech. Rep., 2010.

Nikolaus Vahrenkamp, Institute for Anthropomatics, Karlsruhe Institute of Technology (KIT), Germany. E-mail: vahrenkamp@kit.edu.

Tamim Asfour, Institute for Anthropomatics, KIT, Germany. E-mail: asfour@kit.edu.

Rüdiger Dillmann, Institute for Anthropomatics, KIT, Germany. E-mail: dillmann@kit.edu.