Simultaneous Grasp and Motion Planning

Nikolaus Vahrenkamp, *Member, IEEE,* Tamim Asfour, *Member, IEEE,* and Rüdiger Dillmann, *Fellow, IEEE*

***Abstract*—In this work, we present an integrated approach for planning collision-free grasping motions. The proposed *Grasp–RRT* planner combines the three main tasks needed for grasping an object: building a feasible grasp, solving the inverse kinematics problem and searching a collision-free trajectory that brings the hand to the grasping pose. Therefore, RRT-based algorithms are used to build a tree of reachable and collision-free configurations. During RRT-generation, grasp hypotheses are generated and approach movements toward them are computed. The quality of reachable grasping poses is evaluated via grasp wrench space analysis. We also present an extension to a dual arm planner which generates bimanual grasps together with corresponding dual arm grasping motions. The algorithms are evaluated with different setups in simulation and on the humanoid robot ARMAR-III.**

***Index Terms*—Grasp Planning, Motion Planning, Humanoid Robots.**

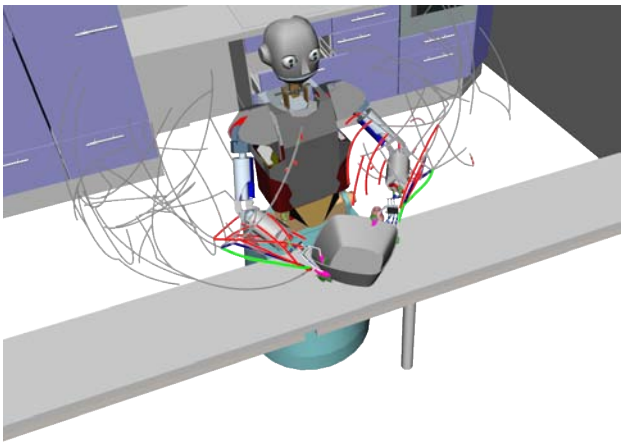

Fig. 1. A bimanual grasping trajectory.

I. INTRODUCTION

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 non-artificial surroundings. One essential ability for working autonomously is to grasp a completely known object for which an internal representation is stored in a database (e.g. information about shape, weight, associated actions or feasible grasps). Furthermore, the robot should be able to grasp objects for which the internal representation is incomplete due to inaccurate perception or uncertainties resulting in an incomplete knowledge base.

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

The *Grasp–RRT* planning algorithm combines the creation of feasible and reachable grasps with the search for collision-free motions and thus no pre-calculated grasping data is needed. This online search for feasible grasping configurations has the advantage that the search is not limited to a potentially incomplete set of offline generated grasps. Furthermore, online generated requirements or constraints, that have to be met by the the grasping configuration, can be implicitly considered. Since such constraints (e.g. don't grasp at a specific part of the object, consider post-grasping stability for transportation, etc) usually limit the number of feasible grasps that can be applied, the object's grasp database must be huge in case offline generated data is used. By generating the grasp hypotheses online, the *Grasp–RRT* approach avoids a filter step on offline generated discretized grasping data. Further, the search for a feasible grasp is focused on reachable configurations and thus the computation of grasping poses is only performed for positions that can be reached by the robot. This focus on the reachable part of the object allows an efficient implementation, as demonstrated by the experiments.

The algorithms can be applied for single and dual arm planning problems and even when just a rough estimation of an unknown object is given, an approximated 3D 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 impacting forces resulting from grasping contacts. In this work we utilize the more common approach of analyzing the Grasp-Wrench space for determining the quality of a grasping hypotheses. Additionally 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.

In the next section, related work dealing with planning motions for grasping is presented. The three parts of the *Grasp–RRT* algorithm (computing grasping poses, generating approach movements and the online grasp quality measurement) are discussed in Section III. In Section IV the *Bimanual Grasp–RRT* algorithm, an approach for generating dual arm grasping motions, is presented. Several experiments for planning single arm and bimanual grasping motions in simulation and on the humanoid robot ARMAR-III (see Fig. 1) are discussed in Section V.

N. Vahrenkamp, T. Asfour and R. Dillmann are with the Institute of Anthropomatics, Karlsruhe Institute of Technology (KIT), Germany, e-mail: (vahrenkamp,asfour,dillmann)@kit.edu.

Manuscript received October 03, 2011; revised February 12, 2012.

II. 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 in general [3]. Hence, complete algorithms will suffer from low performance, mainly caused by the complex task of determining a representation of C_{free}, the part of the configuration space (C-Space) whose configurations do not cause work space collisions. Instead of building a representation of C_{free}, probabilistic algorithms may be used without having an explicit representation of the free space and thus a time consuming computation of C_{free} can be avoided. 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 pre-defined 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 IK-solutions as constraints that limit the operational space, e.g. while opening a door or transporting objects [9]. When using grasp planners (e.g. see [6], [10]), the grasping information is computed in an offline step and the data is stored in a database for use during online search. Such grasp planners strongly rely on determining the quality of a grasping hypothesis. Evaluation of an object grasp by a multi-fingered 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 evaluation the grasp quality is based on the computation of the wrench space, formed by the contact points between hand and object, also called Grasp Wrench Space (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. [13] proposes the concept of the Object Wrench Space (OWS) which represents 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 the scaling factor. In [14], which proposes a task-dependent wrench space, 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 in order 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 at the *KIT Object Models Database* which can be accessed online [20].

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

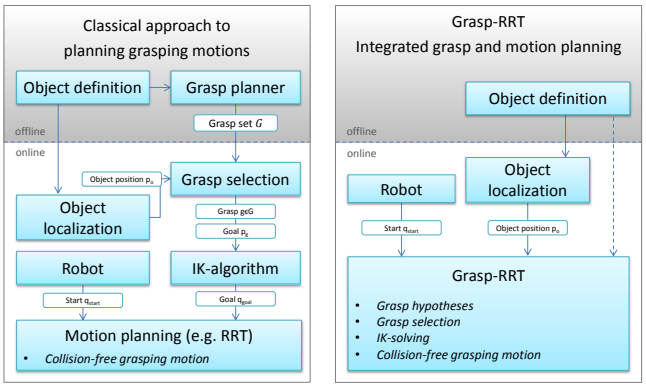

Fig. 2. The classical approach to planning grasping motions compared with the *Grasp–RRT* algorithm.

Planning dual arm motions is addressed in [8] where collision-free motions for two end effectors are planned with RRT-based algorithms for bimanual grasping or re-grasping actions. In the work presented 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 the choice of the grasp on the manifold of valid grasps.

III. INTEGRATED GRASP AND MOTION PLANNING

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

A. Motivation

The process of planning collision-free grasping motions can usually divided in an offline and an online phase. During offline processing, a representation of the object is built (e.g. a 3D-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 offline phase (see e.g. [6]). As shown on the left of Fig. 2 the grasp set G and the current object location are then used during online processing to select a reachable grasp $g \in G$.

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 motion can be found in later processing steps, different grasps must be tried. By applying the grasp specific object-hand relation the workspace target pose $p_g \in SE(3)$ can be determined and passed to an IK-solver. The IK-solver is used to compute a goal configuration q_{goal} that is passed

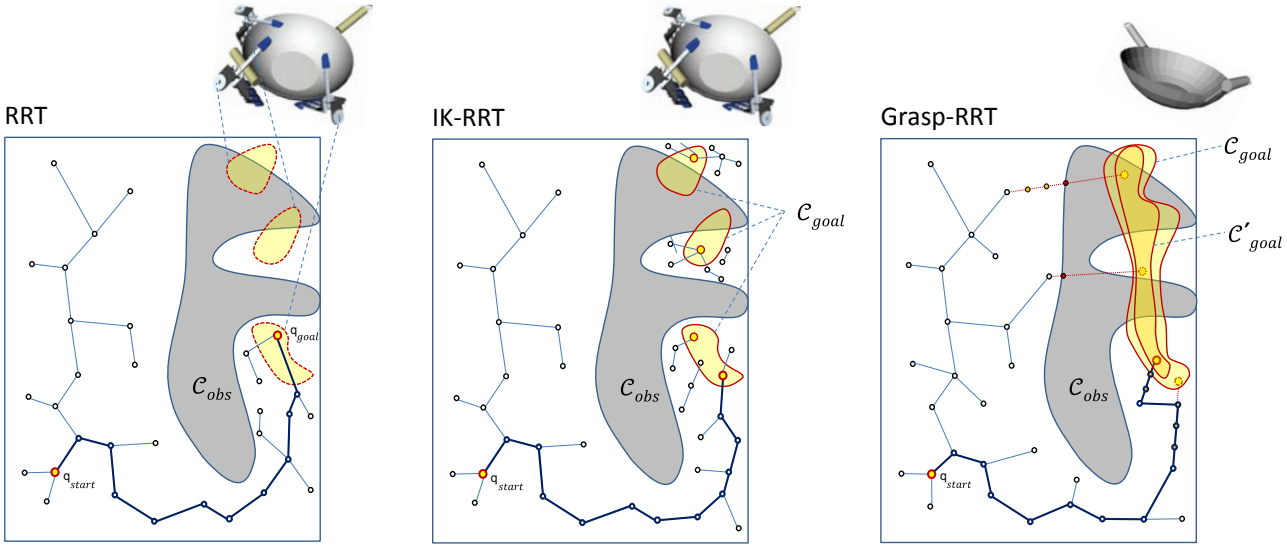

Fig. 3. The behavior of three approaches for planning grasping motions are compared with a simplified 2D example. On the left, the stepwise planning approach is used, which selects one grasp at first, solves the IK problem and finally uses the RRT planner to find a collision-free motion from q_{start} to q_{goal}. In the middle the IK-RRT planner is depicted [8]. 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 in order 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. The *Grasp–RRT* planner, shown on the right, does not rely on any predefined grasping configurations. Instead, approach movements are generated during tree expansion in order to guide the growth in 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 higher grasp qualities are required.

to the motion planning algorithm together with the current robot configuration q_{start}. Since 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-related approaches, can be used to search a collision-free motion.

In contrast to such stepwise approaches, the proposed *Grasp–RRT* algorithm integrates the generation of grasping hypothesis, the selection of feasible grasps, IK-solving and the search for collision-free motions. In Fig. 2, an overview of the proposed planner is shown on the right. The only information that is needed by the planner is the current robot configuration q_{start}, the object's workspace location $p_o \in SE(3)$ and access to a spatial representation of the target object. There is no explicit definition of a target configuration, since the goal is derived from a feasible grasp which is calculated during the planning process. In Fig. 3 a simplified 2D example is shown to compare the behavior of three approaches for planning grasping motions. On the left side a stepwise RRT-based approach, that relies on an offline generated grasp set, is shown. As explained before, the selected grasp and the computed IK-solution specify the target configuration q_{goal} that is used for motion planning. The figure in the middle 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. The right figure 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 direction of C_{goal}. Further, higher requirements in grasp quality and task specific constraints can be taken into account, resulting in a smaller goal region C'_{goal}.

B. The Grasp–RRT Planner

The main loop of the proposed *Grasp–RRT* apporach is presented in Alg. 1. The planner is initialized with the root configuration q_{start} and p_o, the 6D pose of the object that should be grasped. Starting from q_{start} RRT-based extension methods are used to build up a tree of collision-free and reachable configurations. This part of the *Grasp–RRT* planner is similar to uni-directional RRT-approaches where sampling-based extensions are performed to cover the free configuration space C_{free}. Additionally, approach motions together with grasping hypothesis are generated from time to time (see Alg. 2) until a collision-free grasping motion can be determined.

In order 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
```

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
```

C. Approach Movements and Grasping Hypothesis

As shown in Alg. 1, approach movements are generated from time to time in order to test, if a collision-free grasping motion can be determined. Therefore, the following steps are performed:

- Select a node $n_{Approach}$ of the RRT as an initial configuration for performing the approach movement (see Alg. 2, line 1).
- Based on $n_{Approach}$, a virtual target pose p_{grasp} is computed in workspace (see Alg. 2, line 2).
- By utilizing the pseudoinverse Jacobian, the end-effector is moved toward p_{grasp} as far as possible, while adding intermediate configurations to the RRT (see Alg. 2, line 3 and Alg. 3).
- The resulting grasping pose is evaluated by closing the fingers, determining the contacts and performing grasp wrench space analysis (see Alg. 2, line 4 and 5).

In the following, these steps are explained in more detail:

1) 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 $n_{Approach}$ 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 3D relation between 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 object and end effector.

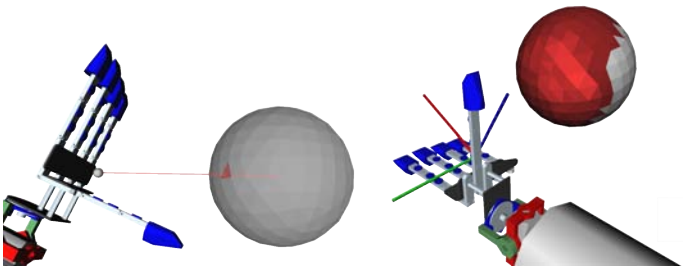

Fig. 4. (a) For each node of the RRT the corresponding surface triangle of the *ApproachSphere* is determined by projection the TCP position on the sphere's surface. (b) The distribution of approach directions is visualized for a planning task by setting the color intensity proportional to the number of RRT-nodes in the direction of the triangle.

In order to encode different approach directions, we propose the use of a data structure, that we call *ApproachSphere* throughout this paper. This 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 TCP position onto the sphere (see Fig. 4(a)). Then n is added to a list of associated RRT-nodes of t_n.

When a random RRT-node $n_{Approach}$ for grasp testing is selected, at first one of the available approach directions represented by the triangles of the *ApproachSphere* is randomly chosen and then 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 this way can be seen in Fig. 4(b). Here, the state of the *ApproachSphere* after building up an RRT is shown. The color intensity of a triangle is proportional to the number of RRT-nodes in direction represented by the triangle. It can be seen clearly that choosing $n_{Approach}$ randomly from all RRT-nodes will result in a non-uniform distribution of approach directions.

2) 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 the GCP and the approach direction of the anthropomorphic hand that is used in our experiments can be seen in Fig. 5. Based on this GCP definition, the target grasping pose $p_{grasp} \in SE(3)$ is determined by searching the point $pt_o \in R^3$ on the object's surface which 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 Fig. 5).

3) Approaching the object: In Alg. 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

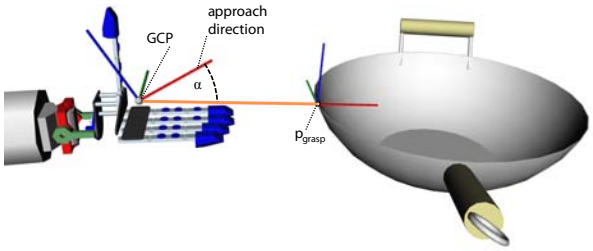

Fig. 5. The computation of the grasping pose p_{grasp}.

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

$n \leftarrow n_{Approach}$
repeat
  $\Delta_p \leftarrow p_{grasp} \cdot (n.p)^{-1}$
  $\Delta_q \leftarrow J^{+}(n.q) \cdot LimitCartesianStepSize(\Delta_p)$
  $n'.q \leftarrow n.q + \Delta_q$
  if $(Collision(n'.q)\,||\,!InJointLimits(n'.q))$ **then**
    return n
  end
  $n'.p \leftarrow ForwardKinematics(n'.q)$
  $RRT.AddNode(n')$
  $n \leftarrow n'$
until $(Length(\Delta_p) < Threshold_{Cartesean})$;
return n

Δ_p between the current TCP pose (stored in the RRT node $n.p$) and the target p_{grasp}. To ensure small steps in workspace the method $LimitCartesianStepSize$ is applied to limit the resulting displacement. Afterwards the corresponding movement Δ_q in C-space is computed by utilizing the pseudoinverse Jacobian. The pseudoinverse J^{+} is derived via singular value decomposition (SVD), although other approaches, which may be more efficient, can be chosen as well. If the resulting configuration $n'.q$, that 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 $Threshold_{Cartesean}$.

4) 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 grasp wrench space analysis is performed as described in the appendix.

IV. Planning Bimanual Grasping Motions

When large objects like the wok in Fig. 8 should be grasped by a humanoid robot, both hands are needed for applying a stable grasp. On 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 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 decoupling must be chosen.

A. Bimanual Grasp–RRT

Fig. 6 depicts an overview of the *Bimanual Grasp–RRT* planner. The 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 the right hand. Furthermore they are configured to search and store grasps until the main planner terminates.

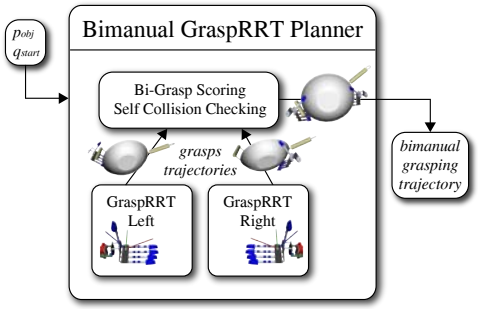

Fig. 6. Overview of the *Bimanual Grasp–RRT* planner.

The *Bimanual Grasp–RRT* planner collects the grasps and the corresponding grasping trajectories for the left and the 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 Section IV-B. If the resulting bimanual score is above the threshold ϱ_{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 Alg. 4).

B. Evaluation of Bimanual Grasps

The grasp wrench space 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 the right hand. The united set $C_g' = C_g^l \cup C_g^r$ is used to build the GWS analogously to the single-handed case. The increase of the number of contact points leads to wider wrench space which results in an higher grasp evaluation, whereas the position of the contact points, respectively the pose of the hands, plays a more crucial role. Another aspect is the performance to quality ratio. As shown in the experiments of Section V, 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,

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

```
GraspRRT_l ← GraspRRTInstance(q^l_start, p_o)
GraspRRT_r ← GraspRRTInstance(q^r_start, p_o)
GraspRRT_l.start()
GraspRRT_r.start()
while (!TimeOut()) do
    /* process new results of GraspRRT_l */
    s_l ← GraspRRT_l.GetNewSolution()
    if (s_l) then
        Results_l.add(s_l)
        foreach (s_r ∈ Results_r) do
            if (BiGraspEvaluation(s_l, s_r) > ϱ_min &&
            !SelfCollision(s_l, s_r)) then
                GraspRRT_l.stop()
                GraspRRT_r.stop()
                return BuildSolution(s_l, s_r)
            end
        end
    end
    /* process new results of GraspRRT_r */
    ...
end
```

since force closure is only needed for the resulting grasp. This leads to an 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 for one hand (e.g. just two fingers in contact with the object) were generated when the corresponding grasp of the other hand was already force closure. Such bimanual grasps are evaluated as satisfactory by the grasp wrench space 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 already mentioned drawback of an increased planning time, the number of valid grasps that can be found by such an approach is limited. Usually there exist bimanual force closure grasps which are composed by two 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 the 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 Alg. 5.

V. EXPERIMENTS

The following experiments state 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^l_g ← ContactPoints(s_l)
C^r_g ← ContactPoints(s_r)
C'_g = C^l_g ∪ C^r_g
return GraspEvaluation(C'_g)
```

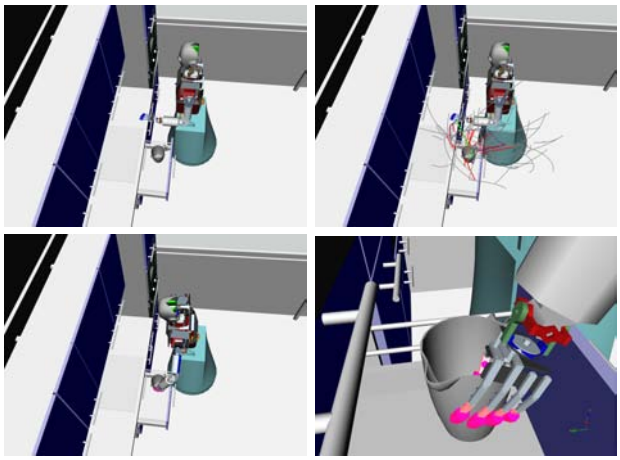

Fig. 7. The *Grasp–RRT* planner is used to search a feasible grasp and a collision-free grasping trajectory for 10 DoF (hip and arm) of ARMAR-III.

A. Measuring Cup in a Drawer

In this experiment the humanoid robot ARMAR-III is supposed to grasp a measuring cup located in a drawer of a kitchen. The robot should use three hip and seven arm joints and thus the C-Space used for planning is 10-dimensional. The setup, depicted in Fig. 7, 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 seconds on average (measured over 50 test runs). When higher grasp qualities are requested, the planning time increases as shown in Table I. The first row shows the results when only force closure grasps are generated and the grasp quality ϱ is ignored. The second row shows results that have been measured when setting ϱ_{min} to 0.08, resulting in grasping configurations with higher quality.

In the top row of Fig. 7 the starting configuration is shown on the left and on the right, the RRT, that was generated during planning, is visualized (the visualization of the C-Space search tree was build by illustrating the corresponding 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. The bottom row of Fig. 7 shows the final configuration on the left and on the right, a narrow view of the computed grasping configuration is shown.

B. A Wok in the Kitchen: Evaluating the Bimanual Grasp–RRT Planner

In this simulation experiment, the *Bimanual Grasp–RRT* planner is queried to find a grasping trajectory for a wok 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 seconds on average. As mentioned in Section. IV, the bimanual approach cannot be applied when common joints, such as hip or platform, should be considered for planning. Due to the parallelized search for a left and a right trajectory, the planner performs well in this experiment (see Table I, row 3). For comparison we present the average performance when all (single handed) grasps are tested for force closure in row 4 of Table I. As described in Section IV-B, the force closure test for single handed grasps limits the set of bimanual grasps that can be found by the planner. Further, the planning time increased significantly to 7.6 seconds.

A resulting grasping configuration together with the collision-free trajectories for the left and the right arm are shown in Fig. 8. The top row shows the initial configuration on the left and an RRT, that was generated during a planning process, is visualized on the right. The final configuration and a narrow view of the planned grasp is shown in the bottom row. The contact points of the bimanual grasping configuration are visualized by the corresponding friction cones.

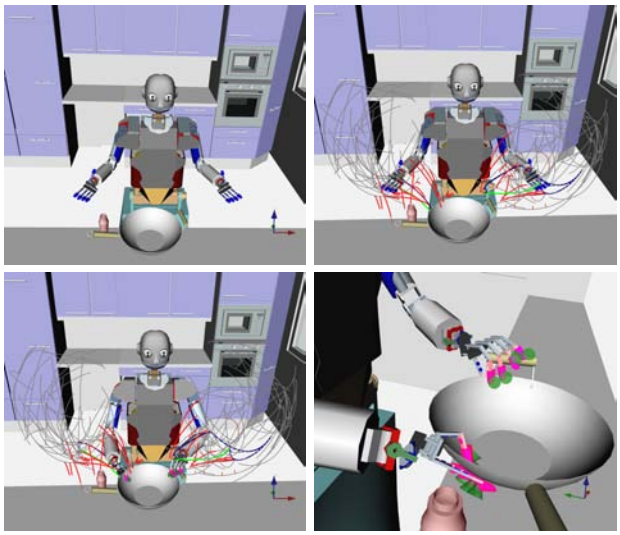

Fig. 8. The *Bimanual Grasp–RRT* planner is used to search a collision-free grasping trajectory for 14 DoF of both arms of ARMAR-III.

C. 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. Fig. 9 shows the results of the planner and the execution of the planned trajectories on the humanoid robot ARMAR-III. The average planning time of this experiment was measured with 0.6 seconds (see Table I, row 5).

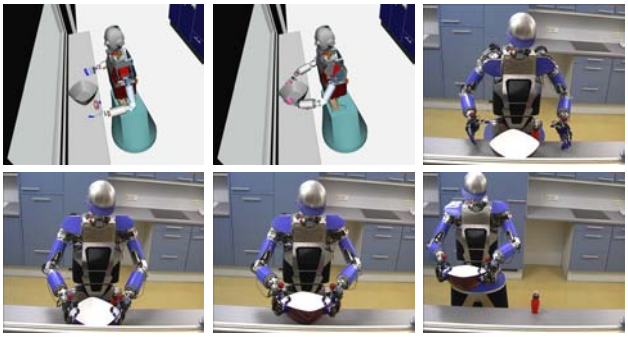

Fig. 9. The *Bimanual Grasp–RRT* planner enables the humanoid robot ARMAR-III to grasp a bowl in the kitchen.

D. Bicycle

In this setup, ARMAR-III is supposed to grasp a bicycle with both hands in order to lift it afterwards. As shown in Table I, the *Bimanual Grasp–RRT* planner is able to find a force closure grasp in 4.0 seconds on average. An exemplary result can be seen in Fig. 10(a). Fig. 10(b) shows a visualization of all generated grasping hypothesis that were build by the sub planners of the *Bimanual Grasp–RRT* algorithm. A workspace visualization of the resulting RRTs generated by both subplanners is given in Fig. 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).

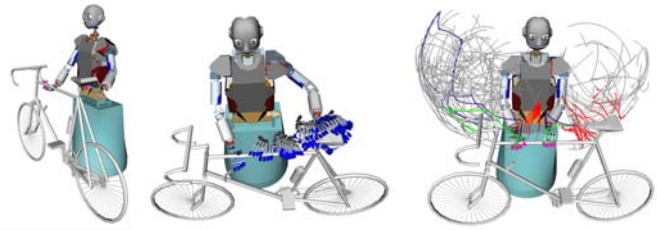

Fig. 10. (a) The final grasp ($\varrho = 0.032$) that was generated with the *Bimanual Grasp–RRT* planner. The friction cones are visualized at the contact points. (b) The visualization shows all grasps that have been generated by the *Grasp–RRT* planners for the left and right hand. The resulting configuration 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.

E. Grasp Quality

When runtime is crucial, the *Grasp–RRT* planner can be used to quickly find a solution that is not optimal. By increasing the runtime better grasping trajectories can be optionally generated. This allows to adapt the quality of the grasp to the demands of online processing, where in some cases a suboptimal solution is preferred over long computation time. In the experiment, depicted in Fig. 11, the quality of the best solution is determined over time (see Fig. 12).

F. Performance

The performance of the proposed *Grasp–RRT* planner in single and dual arm planning setups is presented in Fig. 13

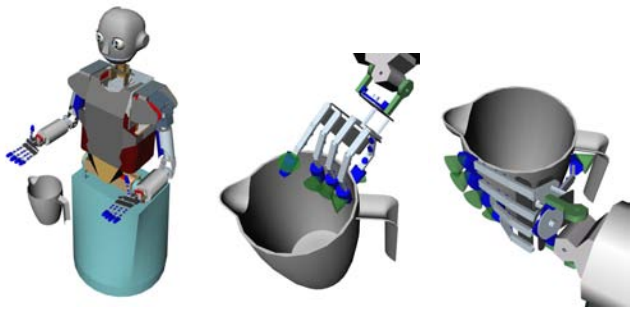

Fig. 11. To evaluate the grasp quality over time, a grasping motion for the left and the right end effector is searched. Once a solution is found, the *Grasp–RRT* planner is not stopped. Instead it is tried to find a solution that results in a better grasp evaluation. The setup is depicted on the left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

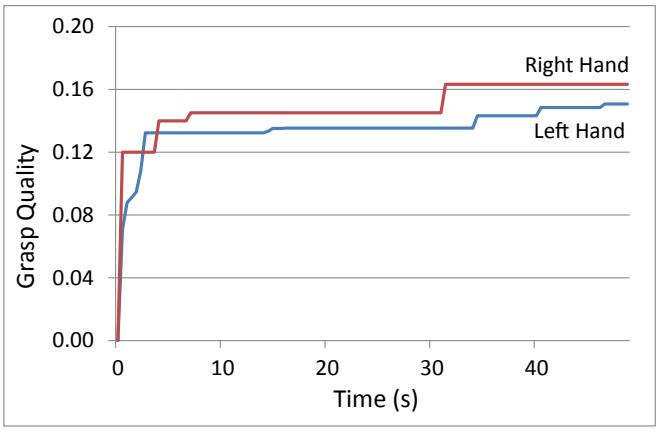

Fig. 12. The quality of the resulting grasp increases over time. In this exemplary setup the first force closure grasp was found after 70 ms respectively 530 ms. When high quality grasps are needed several seconds have to be spent until the quality of a resulting grasp cannot be increased significantly.

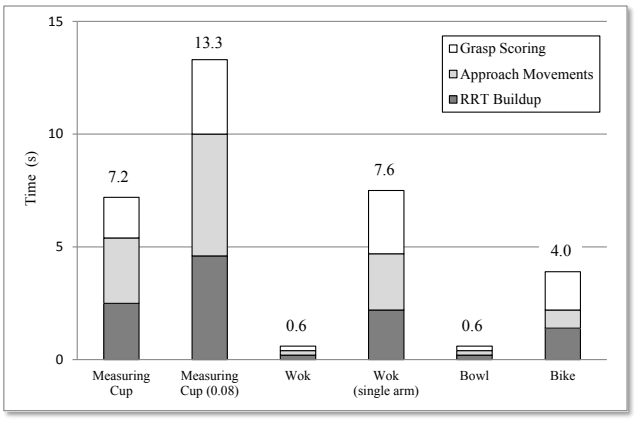

Fig. 13. Overview of the average performance measurements.

TABLE I
PERFORMANCE EVALUATION.

	Planning Time (seconds)				# Ap.	# Gr.
	Total	RRT	Ap.	Score	Traj.	Scores
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

and Table I. The runtime 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 are distinguished, pointing out that the parameter setup was well balanced since approximatively the same amount of time is spent for building up the RRT, computing the approach directions and for evaluating the grasping poses. The last two columns of Table I show the number of approach trajectories which 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.

G. Comparative Study

In the following experiment, we simulate the application of the *Grasp–RRT* approach for grasping an unknown object in order to compare the proposed approach with classical stepwise algorithms for planning grasping motions (e.g. [19]). Therefore we use an imperfect 3D 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 utilizing any pre-computed sets of grasps. Hence, RRT- and IK-RRT-related planning approaches must initially create such grasping information in order to solve IK-queries to determine target configurations for planning, whereas the IK-RRT algorithms continuously samples IK-solutions until a valid grasping motion can be determined. Grasp planning is done 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. Further, an efficient IK-solver must be present for the RRT and IK-RRT planner in order to determine collision-free IK-solutions to one of the planned grasps. Since in general the majority of the planned grasps are not reachable by the manipulator, we perform a filter step to generate a sub-set of reachable grasps. This is done by utilizing pre-computed 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. Further, no precomputed reachability information is needed to speed up any IK-queries. In Fig. 14 the 3D mesh, the start and the final configuration, that was generated during planning, are depicted.

In Table II the results of the RRT, IK-RRT and *Grasp–RRT* approach are shown. Initially the three approaches

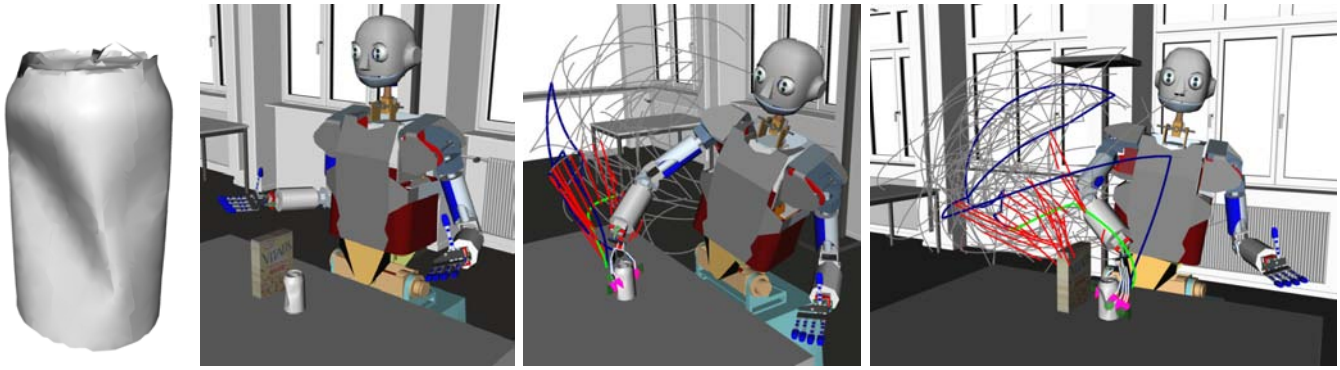

Fig. 14. A 3D-model that was generated with shape retrieval algorithms [24] and the start and configuration, that was used to compare the results of different approaches. Two visualizations of the results of the *Grasp–RRT* approach are shown on the right. The RRT is visualized in workspace, whereas 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 II
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 Obstacle, 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

are compared in a scene where the target object is reachable without difficulty. In this situation, it was sufficient to plan 50 grasps in order 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 seconds. Before planning a collision-free motion with the RRT approach an IK-solver for 10 DoF is queried in order to generate a feasible target configuration. This step, together with filtering the grasps according to their reachability took 32 ms on average. Due to 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 pre-computing a set of feasible grasps, in 339 ms on average. Row 3-6 of Table II show the results when an additional obstacle is introduced in the scene (see Fig. 14 on the right). Due to the limited workspace, 200 grasps have to be planned for the RRT and IK-RRT approaches in order to serve a dense set of grasps that can be used for reliable IK-solving. Hence, the overall planning time increased to over 10 seconds for the RRT and IK-RRT algorithms. The *Grasp–RRT* planner was able to solve the problem in 4.2 seconds on average. As shown in Table II, 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. Further, the *Grasp–RRT* approach is a single directional planner which is known to be slower compared to 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.

VI. CONCLUSION

In this work, a planning approach for computing grasping trajectories was presented. Compared to existing state-of-the art planners, the proposed *Grasp–RRT* planner 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 inverse kinematic 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 to stepwise approaches, where a grasp is selected from a set of precomputed grasps in order 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 heuristic for grasp selection or IK solving and thereby a more general way of generating grasping motions can be achieved.

Further, it was shown that 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 multi-fingered hands. This is achieved by independently consider the generation of grasping trajectory hypotheses for each end effector. These single arm motions are investigated in order to find a feasible combination resulting in bimanual configuration with 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 by the experiments in Section V, collision-free motions for a large variety of single arm and bi-

manual grasping tasks can be efficiently planned with the *Grasp–RRT* planners. Depending on the environment and the task, the planning time varies from less than a second to 7.6 seconds. If runtime is not crucial, higher grasp qualities can be achieved by extending the time that is spent for planning.

Further improvements may be achieved by considering constraints within the grasp quality evaluations. Such constraints may result from demands of post-grasping actions such as lifting or object specific manipulations which should be performed after grasping. Also task specific constraints can be taken into account in order 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 non-convex 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].

APPENDIX

The quality of a grasp is an important aspect for the selection of the best candidate from the set of grasps resulting from grasp planning. A common approach to evaluate the quality of grasps is the construction of the Grasp Wrench Space (GWS), which describes the 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 the 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].

a) 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, an 6D force wrench $w_{i,j}$ can be constructed as shown in Eq. (1), 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 λ guarantees scale invariance, which can be useful in comparing grasps on different objects [13].

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

Based on Eg. 1, the grasp wrench space can be built by determining the convex hull over the union of all wrenches as described in [27].

In Fig. 15 a grasp together with the resulting contacts and the corresponding friction cones is shown on the left. The force sub space of the GWS is visualized in the middle by setting the torque components to zero. On the right the torque sub space is shown, that was built by disabling the force components.

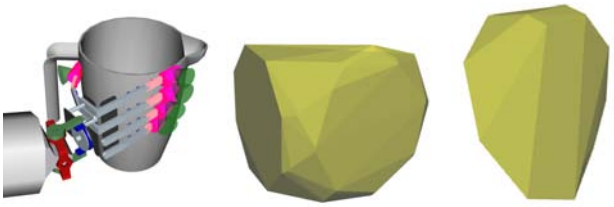

Fig. 15. Two visualizations of the 6D grasp wrench space that was generated for the depicted grasp. The friction cones are visualized at the corresponding contact points. The sub space representing the forces is depicted in the middle. On the right the covered torques are visualized.

b) Object Wrench Space: In [13] the Object Wrench Space (OWS) was introduced as a representation of all potential grasps that can be applied to the 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 Fig. 16, 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 6D 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 OWS is analyzed in order to determine a reference quality value. This ensures that an invariant grasp quality score is computed by the evaluation algorithm.

Fig. 16. Two 3D visualizations of the 6D object wrench space. The visualization in the middle shows the projection in force space and the projection in torque space is shown on the right. The shown OWS subspaces were build by setting the torque respectively the force to zero.

c) Grasp Quality: Several approaches to determine the quality of a grasp wrench space can be found in 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 the GWS. Further, 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 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 3D force space was investigated instead of analyzing 6D 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 high efficient grasp quality measurement component within the *GraspStudio* library of *Simox* [28], which offers the fast determination of object and grasp wrench spaces.

This implementation allows efficient computation for 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.

Additionally the OWS ϵ value (denoted with ϵ') is determined in a precomputing step. With ϵ' a reference value for a *perfect grasp* is computed that 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. The objects that were used in this article resulted in an ϵ' value between 0.4 and 0.85. Finally the quality $\varrho \in [0, 1]$ of a grasp is computed as follows:

$$\varrho = \frac{\epsilon}{\epsilon'}. \tag{2}$$

Note, that the grasp quality evaluation component of the *Grasp–RRT* approach is exchangeable, allowing to incorporate other algorithms for determining the grasp quality. Further, 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. A more in-depth discussion of grasp wrench space analysis can be found in [14], [29] or [30].

ACKNOWLEDGMENT

The work described in this paper was partially conducted within the German Humanoid Research project SFB588 funded by the German Research Foundation (DFG: Deutsche Forschungsgemeinschaft) and the EU Cognitive Systems project GRASP (IST-FP7-IP-215821) funded by the European Commission.

REFERENCES

[1] J. Kuffner and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000, pp. 995–1001.

[2] N. Vahrenkamp, M. Do, T. Asfour, and R. Dillmann, "Integrated Grasp and Motion Planning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, USA, Mai 2010, pp. 2883–2888.

[3] J. H. Reif, "Complexity of the mover's problem and generalizations," in *SFCS '79: Proceedings of the 20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*. Washington, DC, USA: IEEE Computer Society, 1979, pp. 421–427.

[4] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[5] E. Drumwright and V. Ng-Thow-Hing, "Toward interactive reaching in static environments for humanoid robots," in *Proceedings of the IEEE International Conference on 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 *Proceedings of IEEE-RAS International Conference on 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 *Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, November 2007.

[8] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner, and R. Dillmann, "Humanoid Motion Planning for Dual-Arm Manipulation and Re-Grasping Tasks," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, USA, October 2009, pp. 2464–2470.

[9] M. Stilman, "Task constrained motion planning in robot joint space." in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 3074–3081.

[10] A. T. Miller, "Graspit!: a versatile simulator for robotic grasping." Ph.D. dissertation, Dep. of Computer Science, Columbia University, 2001.

[11] K. Lakshminarayana, "Mechanics of form closure," ASME, Tech. Rep., 1978.

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

[13] N. Pollard, "Parallel methods for synthesizing whole-hand grasps from generalized prototypes," Ph.D. dissertation, MIT, Dept. of Electrical Engineering and Computer Science, 1994.

[14] C. Borst, M. Fischer, and G. Hirzinger, "Grasp planning: How to choose a suitable task wrench space," in *Proceedings of the IEEE International Conference on 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 *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 2, Sept. 2003, pp. 1824 – 1829 vol.2.

[16] K. Huebner, S. Ruthotto, and D. Kragic, "Minimum volume bounding box decomposition for shape approximation in robot grasping," in *Proceedings of the IEEE International Conference on 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 *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[18] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, "The columbia grasp database," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2009.

[19] Z. Xue, A. Kasper, J. Zllner, and R. Dillmann, "An automatic grasp planning system for service robots," in *Proceedings of the 14th International Conference on Advanced Robotics (ICAR)*, 2009.

[20] A. Kasper, R. Becher, P. Steinhaus, and R. Dillmann, "Developing and analyzing intuitive modes for interactive object modeling," in *Proceedings of the 9th international conference on Multimodal interfaces*, ser. ICMI '07. New York, NY, USA: ACM, 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," *Autonomous Robots*, vol. 31, pp. 87–102, May 2011.

[22] M. Gienger, M. Toussaint, and C. Goerick, "Task maps in humanoid robot manipulation," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2008.

[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," *Robotics and Autonomous Systems*, 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 *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 5226–5232.

[25] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof, "Grasp planning via decomposition trees," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 4679–4684.

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

[27] C. Ferrari and J. Canny, "Planning optimal grasps," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, May 1992, pp. 2290 –2295 vol.3.

[28] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Simox: A Simulation and Motion Planning Toolbox for C++," Karlsruhe Institute of Technology (KIT), Tech. Rep., 2010.

[29] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *Robotics Automation Magazine, IEEE*, vol. 11, no. 4, pp. 110 – 122, Dec. 2004.

[30] R. Krug, D. Dimitrov, K. Charusta, and B. Iliev, "On the efficient computation of independent contact regions for force closure grasps," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, oct. 2010, pp. 586 –591.