Efficient Motion and Grasp Planning for Humanoid Robots

Nikolaus Vahrenkamp and Tamim Asfour and Rüdiger Dillmann

Abstract The control system of a robot operating in a human-centered environments should address the problem of motion planning to generate collision-free motions for grasping and manipulation tasks. To deal with the complexity of these tasks in such environments, different motion planning algorithms can be used. We present a motion planning framework for manipulation and grasping tasks consisting of different components for sampling-based motion planning, collision checking, integrated motion planning and inverse kinematics as well as for planning of single arm grasping and dual-arm re-grasping tasks. We provide an evaluation of the presented methods on the humanoid robot ARMAR-III.

1 Introduction

Motion planning for humanoid robots with a high number of degrees of freedom (DoF) requires computationally efficient approaches to determine collision-free trajectories. The planning algorithms have to fulfill several requirements, such as low runtime, short path length or reasonable distance to obstacles. Since the motion planning problem is known to be PSPACE-hard [24], complete algorithms ([9, 26]) are time consuming and therefore less suitable for real-time tasks of highly redundant humanoid robots which operate in a human-centered environment. Over the last years efficient probabilistic, sampling-based approaches have been developed [19]. These approaches are probabilistically complete which means that if the motion planning problem does not have a solution the algorithm will run forever. To overcome this problem, an implementation will usually stop the search after a specified time and will report that a solution does not exist. This drawback is compensated by the efficiency of probabilistic planners. Efficient and problem adapted implementa-

Nikolaus Vahrenkamp · Tamim Asfour · Rüdiger Dillmann
Institute for Anthropomatics, University of Karlsruhe, Adenauerring 2, 76131 Karlsruhe, Germany,
e-mail: {vahrenkamp,asfour,dillmann}@ira.uka.de

tions of probabilistic planning approaches can be applied to real world robot systems and they are suitable for robots operating in a human-centered environment.

1.1 RRT-Based Planning

Rapidly-Exploring Random Trees (RRTs) belong to the category of sampling-based, randomized planning algorithms [16, 20]. They are widely used for single-query path planning because of their simplicity and efficiency as well as the possibility of involving differential constraints and many degrees of freedom. Several variations from the basic RRT algorithm to problem-adapted planners have been arisen in the last few years due to the multifarious application range of Rapidly-Exploring Random Trees [18]. The key advantages of the basic Rapidly-Exploring Random Tree construction, described in [16], are that the expansion of a RRT is biased toward unexplored state space and only few heuristics and parameters are needed. Since RRT-based algorithms are probabilistically complete, the coverage of the C-space gets arbitrarily close to any point in the free C-space with increasing iterations.

In principle, the basic RRT can already be used as a planner because of the fact that its vertices will eventually cover the whole collision-free configuration space C_{free} coming arbitrarily close to any specified goal configuration c_{goal}. But such a basic planner would suffer from slow convergence and bad performance, so that problem-adapted improvements are reasonable.

1.2 The Motion Planning Framework

A motion planning framework should provide different planning algorithms for selected problems like navigation, approaching, grasping and manipulating. Planning with specialized algorithms allows the use in real-world applications e.g. for service or humanoid robots. These kind of robots must be able to plan motions in reasonable time, so that the human-robot interaction is not affected by long planning time. In the optimal case the user should never be aware of the internally running planning system.

The motion planning framework shown in Fig. 1 is embedded in the robot control software and offers a pool of motion planning services which can be configured depending of the requested planning operation. The planning system selects and configures the specialized planning services, provided by the planning framework, in order to generate a collision-free trajectory that fulfills the needs of a higher level task planning. Therefore, the motion planning component communicates with inverse kinematics (IK) solvers, a collision checker and grasp planner that provides feasible grasps for manipulation tasks. The generated trajectories are delivered to lower level control system which is responsible for the execution on the robot hardware.

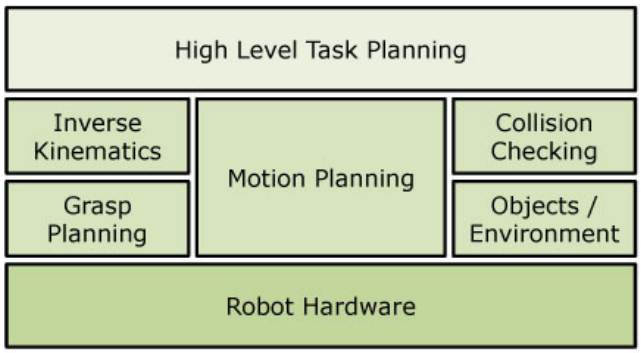


Fig. 1 The Motion Planning Framework.

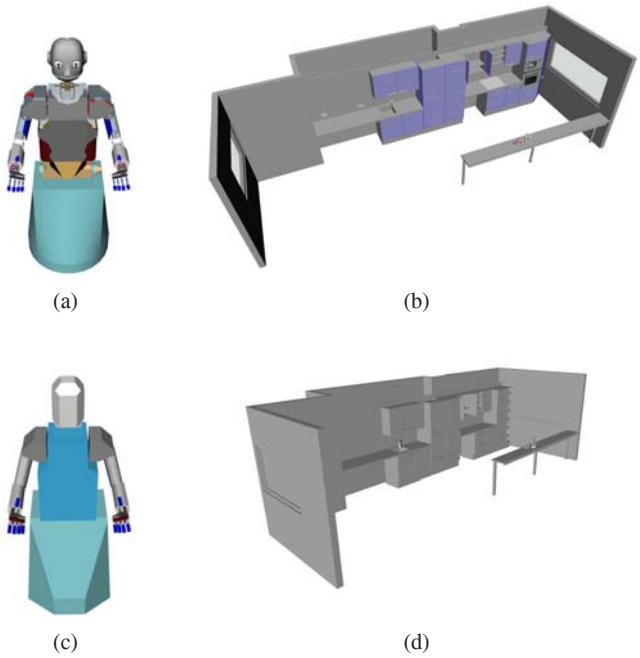
2 Collision Checks and Distance Calculations

Motion planning approaches need collision and/or distance computation methods that operate on 3D models of the robot and the environment. Several libraries are available and can be used for collision detection ([21, 14, 17]). In our framework we are using the PQP library which uses swept sphere volumes to test the collision status for 3D models [17], because of the fast and robust implementation and the included distance computation routines.

Typically the collision checking of sampled configurations is the most time consuming task in RRT planners. A typical planning query requires thousands of collision checks. Thus a RRT-planner greatly benefits from speeding up the collision check routines [29]. To achieve a faster collision computation we are using simplified collision models of the robot and the environment in which it is operating. This reduces the number of triangles from 32.500 to 650 in the case of the robot model and from 24.000 to 4.200 in the case of the environment model. The reduction leads to an average collision query time of 0.20*ms* and an average distance query time of 0.65*ms*. Compared to 0.32*ms* and 3.58*ms* for a setup with full models of the robot and the environment we could achieve a speedup of 37.5% and 81.8%. These performance evaluations have been carried out for motion planning tasks of the humanoid robot ARMAR-III in a kitchen environment (see [3]) on a Linux-Pentium4 system with 3.2 GHz .

3 Weighted Sampling

The dimensions of the configuration space C differ in the effect of the robot system in workspace. Since each dimension of C describes a different effect in workspace, the dimensions have to be weighted in order to generate a uniform sampling.



(a) (b)

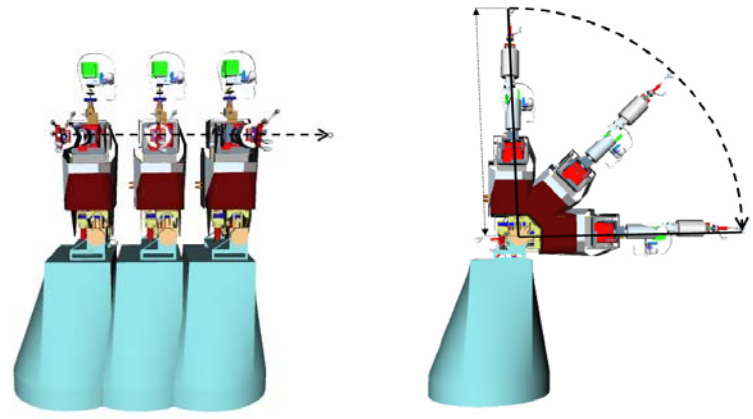
(c) (d)

Fig. 2 Full CAD models of the humanoid robot ARMAR-III (a) and the kitchen environment (b). The simplified models (c),(d) are used for collision checks and distance calculations.

There are several ways of doing joint weighting. A simple technique is a manual and fixed weighting with $w_i > w_j$, if the center of the axis of joint i is further away from the tool center point of the robot manipulator than from the center of the axis of joint j. These distances however are different in each single configuration and depending on the actual robot state they have more or less effect on the movement of the robot, so dynamically adapted joint weights would be a better choice than fixed weighting. But dynamically weight adaption increases the computation time since the distances between all joints have to be computed for every single configuration. Therefore, an upper bound for the workspace movements of each limb is used for an efficient and approximated uniform sampling.

A change ε_{trans} in a translational component of the C-space moves the robot in workspace by ε_{trans}. All other dimensions of the C-space have to be investigated explicitly to derivate the upper bound of the robot's workspace movement. Table 1 gives an overview of the maximum displacement of a point on the robot's surface when changing one unit in C. The effects of moving one unit in the different dimensions can be seen in Fig. 3.

The different workspace effects are considered by using a weighting vector w whose elements are given by the values of the workspace movements from ta-



(a) The humanoid robot ARMAR-III moving in a translational dimension.

(b) The effect in workspace when changing the *C*-space value for the dimension associated with the torso pitch joint.

Fig. 3 Workspace effects of joint movements.

ble 1. In Eq. 1 the maximum workspace movement $d_{WS}(\mathbf{c})$ of a C-space path $\mathbf{c} = (c_0, ..., c_{n-1})$ is calculated.

$$d_{WS}(\mathbf{c}) = \sum_{i=0}^{n-1} w_i c_i \tag{1}$$

To sample a *C*-space path between two configurations $\mathbf{c}_1$ and $\mathbf{c}_2$, the vector $\mathbf{v}_{step}$ is calculated (Eq. 2). For a C-space displacement of $\mathbf{v}_{step}$ it is guaranteed that the maximum workspace displacement is 1mm.

$$\mathbf{v}_{step}(\mathbf{c}_1, \mathbf{c}_2) = \frac{(\mathbf{c}_2 - \mathbf{c}_1)}{d_{WS}(\mathbf{c}_2 - \mathbf{c}_1)} \tag{2}$$

The maximal workspace step size ε_{ws} can be specified in millimeters, which allows to control the granularity of the planning algorithms in an easy way. The step size ε_{ws} is used to generate $n = \lceil \frac{d_{ws}}{\varepsilon_{ws}} \rceil - 1$ intermediate sampling configurations $\mathbf{c}_k$ on the path between two configurations $\mathbf{c}_1$ and $\mathbf{c}_2$.

$$\mathbf{c}_k = \mathbf{c}_1 + k\varepsilon_{ws}\mathbf{v}_{step}(\mathbf{c}_1, \mathbf{c}_2), \;\; k = (1, .., n) \tag{3}$$

This sampling of the C-space path $(\mathbf{c}_2 - \mathbf{c}_1)$ guarantees that the workspace movements of the robot for two successive intermediate configurations is smaller than the upper limit of ε_{ws} mm.

Since the collision status of a path is gained by checking all discrete samples for collisions, it is not guaranteed that intermediate configurations don't result in a collision. This guarantee can be given by Quinlan's *Free Bubble* approach [23] or

an extension presented in [29] where enlarged robot models and lazy path validation approaches are used to efficiently validate complete path segments.

Table 1 Worst case workspace movement for ARMAR-III.

Degree of freedom	*mm*	*Degree of freedom*	*mm*	*Degree of freedom*	*mm*	*Degree of freedom*	*mm*
Platform Translation x	1	Head Pitch	300	Arm Elbow	390	Hand Index 1	70
Platform Translation y	1	Head Roll	300	Wrist 1	150	Hand Index 2	70
Platform Rotation	1 176	Head Yaw	300	Wrist 2	150	Hand Middle 1	70
Torso Pitch	1 176	Arm Shoulder 1	700	Wrist 3	150	Hand Middle 2	70
Torso Roll	1 176	Arm Shoulder 2	700	Hand Thumb 1	70	Hand Ring	70
Torso Yaw	1 176	Arm Shoulder 3	390	Hand Thumb 2	70	Hand Pinkie	70

4 Planning Grasping Motions

Grasping an object is a typical and often needed operation for a service robot that operates in a human-centered environment. In some cases the task of grasping an object includes a suitable grasping pose which implicit defines the target configuration of the grasping motion, but in most cases there is no need of limiting the grasping to one specific grasp pose. In this section we want to show how the search for collision-free grasping configurations can be included in the planning process. In [7] a goal distance heuristics is used to implicitly compute reachable goal configurations during the planning process. Instead of using heuristic workspace goal functions, we are using predefined grasping poses in order to be able to use non-symmetric objects. In [5] a subset of feasible grasps is pre-calculated via a scoring function and the planning is done with the so determined target poses. The BiSpace approach, presented in [11], builds up two search trees, one in the C-space and one in the goal space and a global solution is searched by connecting the search spaces. The motion planning algorithms described in [6] work on continuous workspace goal regions instead of single target configurations. The JT-RRT approach, presented in [31], avoids the explicit search for IK solutions by directing the RRT extensions toward a 3D workspace goal position. Therefore the transposed Jacobian is used to generate C-space steps out of a workspace goal direction. The JT-RRT approach can be useful when no IK-solver is present for a robot system and only a grasping pose in workspace is known. Since there is no explicit C-space target configuration defined, the approach could not be implemented as a bi-directional RRT and the advantages of the Bi-RRT algorithms can not be applied.

Our proposed algorithms unite the search for collision-free motions with the search for solutions of the Inverse Kinematics (IK) problem to one planning scheme. The planners are initialized with a set of grasping poses which are used to calculate feasible target configurations. The computation of feasible grasping poses is done

during the planning process and thus the planning is not limited to an potentially incomplete set of target configurations.

4.1 Predefined Grasps

If an object should be grasped with an end effector of the robot, a collision-free trajectory has to be planned in order to bring the hand to a pose P_{grasp} which allows applying a feasible grasp. This grasping pose is defined with respect to the pose of the target object and could be derived by applying the grasp-specific transformation T_i.

For each object which should be grasped or manipulated by the robot, a collection of feasible grasps is stored in a database. This collection of feasible grasps hold information about the transformations between the end effector and the final grasping position, the type of grasp, a preposition of the end effector and some grasp quality descriptions. These database entries can be generated automatically (e.g. with GraspIt! [22]) or, like in the following examples, by hand. A wok with 15 feasible grasps for each hand of the humanoid robot ARMAR-III can be seen in Fig. 4(a).

To grasp an object o (located at position P_o) with the end effector e by applying the grasp g_k of the feasible grasp collection gc_o^e, the Inverse Kinematics problem for the pose P_k^o has to be solved.

$$P_k^o = T_k^{-1} * P_o \tag{4}$$

4.2 Randomized IK-Solver

To grasp a fixed object with one hand, the IK-problem for one arm has to be solved. In case of the humanoid robot ARMAR-III, the operational workspace can be increased by additionally considering the three hip joints of the robot. This leads to a 10 DoF IK problem. Typically, an arm of a humanoid robot consists of six to eight DoF and is part of a more complex kinematic structure. If an analytical method exists for solving the IK problem for one arm, a randomized algorithm can be constructed which randomly samples the preceding joints and uses the analytical IK-solver for determining the final arm configuration. This probabilistic approach increases the operational workspace of the robot arm and is suitable for randomized planning algorithms.

For ARMAR-III we use a specialized analytic approach for solving the 7 DoF IK problem for one arm where all possible elbow positions are computed and, depending on the parameterization, the best one is chosen [1]. If there are multiple solutions, the behavior can be adjusted. Either the one with the lowest accumulated joint movement or a random solution out of the set of possible results is chosen. In most cases it is desirable to consider the joints of the hip since the reachable

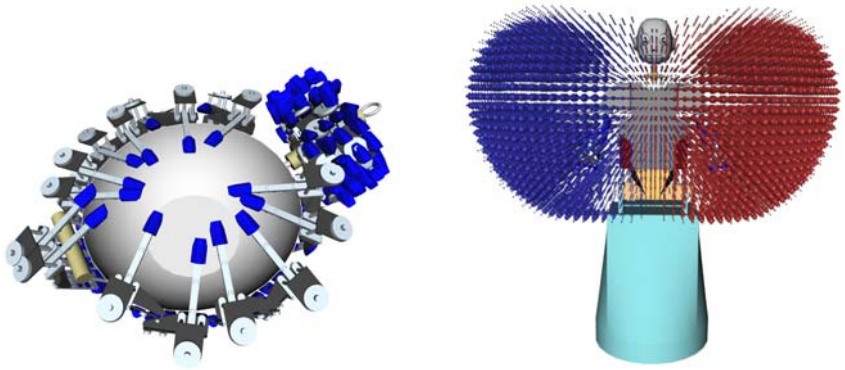
workspace increases significantly when using additional degrees of freedom. In this case the three hip joints of ARMAR-III are randomly sampled until an IK query is successfully answered. If a configuration was found which brings the end effector to the desired pose, the IK solution has to be checked against self-collisions and collisions with obstacles in order to avoid invalid configurations. If the collision checker reports any collisions, the solution is rejected and the search is continued.

The approach is probabilistically complete, which means if time goes to infinity the algorithm will find a solution if at least one exists. To avoid endless runtime, the search for an IK solution is stopped after a specific number of tries and it is assumed that there is no valid result. Since this IK-solver is used within a probabilistic planning algorithm, this approach fits well in the planning concept.

4.2.1 Reachability Space

The use of a reachability space can speed up the randomized IK-solver. The reachability space represents the voxelized 6D-Pose space where each voxel holds information about the probability that an IK query can be answered successfully [4, 11]. It can be used to quickly decide if a target pose is too far away from the reachable configurations and therefor if a (costly) IK-solver call makes sense.

The reachability space of the two arms of ARMAR-III is shown in Fig. 4(b). Here the size of the three dimensional projections of the 6D voxels is proportional to the probability that an IK query within the extend of the voxel can be answered successfully. The reachability space is computed for each arm and the base system is linked to the corresponding shoulder.



(a) (b)

Fig. 4 (a) An object (wok) with predefined grasping positions for two arms of ARMAR-III. (b) The 3D projection of the reachability spaces for both arms of ARMAR-III.

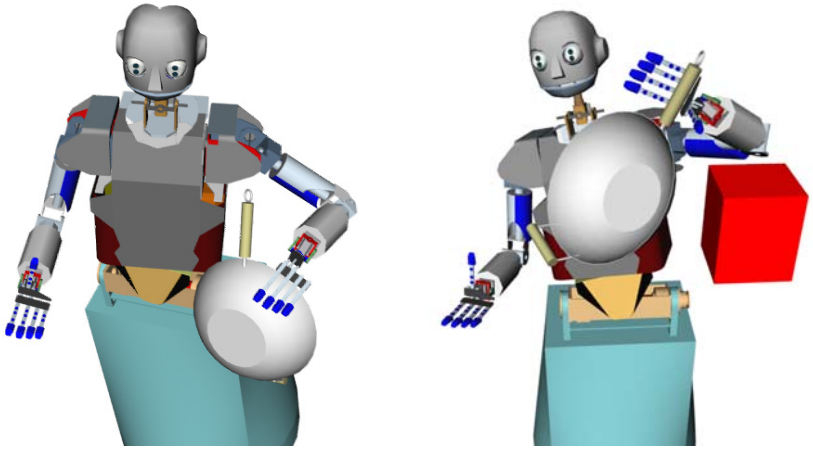
The reachability spaces can be computed by solving a large number of IK requests and counting the number of successful queries for each voxel. Another way

of generating the reachability space is to randomly sample the joint values while using the forward kinematics to determine the pose of the end effector and thus the corresponding 6D voxel [4]. An analytic approach of generating a representation of the reachability is presented in [15].

Since the reachability space is linked to the shoulder, it moves when setting the three hip joints randomly in the search loop of the probabilistic IK-solver. For this reason, the target pose P_k, which is given in the global coordinate system, is transformed to the shoulder coordinate system and the corresponding voxel of the resulting pose P'_k is determined. The analytical IK-solver is only called if the entry of this voxel is greater than zero or a given threshold.

4.2.2 A 10 DoF IK-Solver for Armar-III

The most convenient kinematic chain for reaching or grasping an object with ARMAR-III consists of the three hip joints followed by 7 arm joints. This 10 DoF structure leads to a large reachable workspace and thus enables the robot to perform grasping and manipulation operations without moving the base platform.



(a) An empty scene. (b) A scene with an obstacle.

Fig. 5 Solutions of the 10 DoF IK-solvers.

To measure the performance of the 10 DoF IK-solver, the wok with 15 associated grasping poses is set to a random pose in front of the robot. Then the IK-solvers with and without reachability space are called in order to find a valid configuration for bringing the end effector to one of the 15 grasping poses. Two exemplary results of the IK-solver in an empty and a partly blocked scene are shown in figure 5. The results of table 2 are determined by building the averages of 100 IK queries

with different object poses [1]. The average runtime and the number of calls of the analytical 7 DoF IK-solver are given for setups with/without reachability space and in scenes with/without obstacles. It turns out that the use of the reachability space speeds up the IK-solver enormously and it allows the use of these approaches in real world applications.

Table 2 Performance of the 10 DoF IK-Solvers.

	Without Obstacle		With Obstacle	
	Avg. Runtime	# IK calls	Avg. Runtime	# IK calls
Without Reach. Space	1 404 ms	101.9	2 880 ms	217.3
With Reach. Space	60 ms	6.1	144 ms	13.6

4.3 RRT-Based Planning of Grasping Motions with a Set of Grasps

In most cases the task of grasping an object should not be limited to one specific grasp. Instead of configuring the motion planner with one explicit target configuration, the framework can choose a set of grasp candidates and configure the motion planning services accordingly. It is possible to calculate an IK solution for each grasp and to use this set of configurations as targets for the planning process. This will lead to a planning scheme where the search for a solution is limited to the pre-calculated IK solutions. Since in general there are infinite numbers of solutions for the IK problem, the planner could fail although there is a valid motion for a not considered IK solution. Furthermore, it can be time consuming to calculate the IK solutions for every grasping pose in advance. If the feasible grasps are densely sampled, the precalculation has to be done for a lot of workspace poses. These problems can be avoided, if the search for valid IK solutions is included in the planning process.

The following two sections present two algorithms that determine an IK solution while planning. Both of these algorithms take as input the grasp set for the object and output a joint-space trajectory to reach the object.

4.3.1 J^+-RRT

The J^+-RRT planner can be used to search a collision-free grasping trajectory without the explicit implementation of an IK-solver. During RRT-buildup the planner tries to connect the existing tree to predefined 6D grasping poses defined in workspace. In case a grasping pose was successfully connected to the RRT, a solu-

[1] These performance evaluations have been carried out on a DualCore system with 2.0 GHz.

tion was found and the resulting trajectory is optimized with standard path pruning techniques [12].

Algorithm 1: J^+-RRT(q_{start},p_{obj},gc)

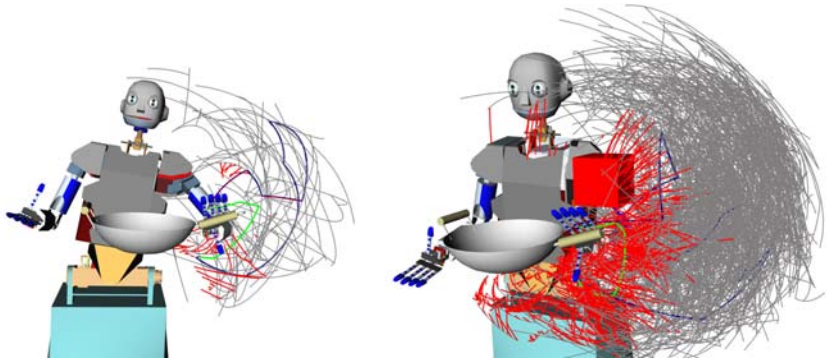
```
1  RRT.AddConfiguration(q_start);
2  while (!TimeOut()) do
3      ExtendRandomly(RRT);
4      if (rand() < p_ExtendToGoal) then
5          Solution ← ExtendToGoal(RRT, p_obj, gc);
6          if (Solution ≠ NULL) then
7              return PrunePath(Solution);
8      end
9  end
```

Algorithm 2: $ExtendToGoal(RRT, p_{obj}, gc)$

```
1  grasp ← GetRandomGrasp(gc);
2  p_target ← ComputeTargetPose(grasp);
3  q_near ← GetNearestNeighbor(RRT, p_target);
4  repeat
5      p_near ← ForwardKinematics(q_near);
6      Δ_p ← p_target − p_near;
7      Δ_q ← J^+(q_near) * LimitCartesianStepSize(Δ_p);
8      q_near ← q_near + Δ_q;
9      if (Collision(q_near) || !InJointLimits(q_near)) then
10         return NULL;
11     RRT.AddConfiguration(q_near);
12 until (Length(Δ_p) > Threshold_Cartesean) ;
13 return BuildSolutionPath(q_near);
```

In Alg. 1 the main part of the J^+-RRT planner is shown. The planner is initialized with the starting configuration q_{start}, the workspace pose p_{obj} of the object and a set of feasible grasps ($gc = \{g_0, .., g_k\}$) defined relatively to the object. The RRT algorithm is used to build up a tree of reachable and collision-free configurations. When a new configuration is added to the tree, the corresponding workspace pose of the hand is stored with the configuration data in order to use it later for calculating the approach movements. Approach movements are generated by the *ExtendToGoal* method which is called with some probability at each iteration of the planner. In Alg. 2 a random grasp is chosen and, combined with the pose of the manipulation object, the workspace target pose p_{target} of the end effector is determined. The workspace metric described in Eq. 8 is used to determine the nearest neighbor q_{near} out of all RRT-entries. q_{near} is marked, so that in case of a a failed extension loop, it will never be chosen twice as a nearest neighbor. Then the Jacobian $J(q_{near})$ is calculated and it's Pseudoinverse $J^+(q_{near})$ is built by single value decomposition. The workspace pose difference Δ_p between the nearest node and p_{target} is used to calculate a delta in C-space which biases the extension toward the target. If the

new configuration is collision-free and does not violate joint limits, the tree is extended by q_{new}. The workspace directed C-space extension is performed until the new configuration is not valid or p_{target} is reached, which means that a solution was found.



(a) An empty scene. (b) A scene with an obstacle.

Fig. 6 The results of the J^+-RRT algorithm. The red parts are generated by the *ExtendToGoal* method of the approach.

In Fig. 6 a resulting RRT of a J^+-RRT planner in an empty scene and in an environment with an obstacle is depicted. The resulting grasping trajectory and it's optimized version are shown in blue and green. The optimized version was generated by standard path pruning techniques [12]. The red parts of the search tree have been generated by the *ExtendToGoal* part of the approach, where the pseudoinverse Jacobian is used to bias the extension to a grasping pose. The figure shows, that the search is focused around the grasping object, but in most cases the joint limits and collisions between hand and wok prevent the generation of a valid solution trajectory. The performance of the J^+-RRT planner can be seen in table 3. It turns out that the usability of the approach is limited in cluttered scenes because of the long runtime.

4.3.2 A workspace metric for the nearest neighbor search

The metric used for determining the nearest workspace neighbor combines the Cartesian distance $d_{position}$ with the rotational distance $d_{orientation}$ of two poses.

$$d_{position}(p_0, p_1) = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2} \tag{5}$$

To compute the orientational distance, the angle between the two orientations is computed. Since the orientational component of the pose is represented with quaternions, the angle between two quaternions q_0 and q_1 has to be computed. This can

be done by computing q_{dif}, the quaternion representing the rotation between q_0 and q_1.

$$q_{dif}(q_0, q_1) = q_0^{-1} * q_1 \tag{6}$$

The rotational distance is set to the angle of q_{dif}, which can be retrieved like in equation 7.

$$d_{orientation}(q_0, q_1) = 2 * acos(q_{dif}(q_0, q_1).w) \tag{7}$$

The final metric is the weighted sum of the Cartesian and the rotational distance. In our experiments we set the factor α to a value so that a difference in the orientation of 1 degree has the same effect as a translation of $3mm$.

$$d(p_0, p_1) = \alpha * d_{position} + (1 - \alpha) * d_{orientation} \tag{8}$$

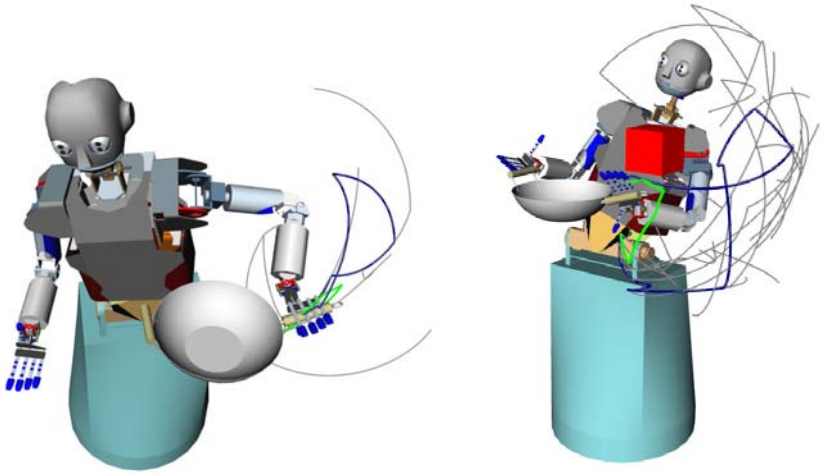
4.3.3 IK-RRT

To speedup the planning, an IK-solver can be used in order to generate goal configurations during the planning process. The planner is configured with a set of feasible grasping poses, which, combined with the pose of a manipulation object, defines a set of workspace target poses. These poses are used as input for the IK-solver calls. As shown in Alg. 3 the IK-RRT algorithm is initialized by adding the start configuration to the first RRT while the second RRT is empty until an IK solution was found. During the planning loop, both trees are extended with standard RRT-based methods and it is tried to connect them via an intermediate configuration. With some probability, a random grasp out of the set of feasible grasps is chosen and a call to the randomized IK-solver is performed. When a feasible IK configuration q_{IK} is found, it is added to the second tree and the new node is marked as a solution node.

Since the IK search is probabilistically complete for the set of grasps and the RRT-Connect algorithm is known to be probabilistically complete [16], the IK-RRT approach is probabilistically complete. This means, that as time goes to infinity the algorithm will find a solution if at least one exists.

At figure 7 a result of the IK-RRT approach is shown. The original and the optimized solution path are depicted in blue and green. Due to the bi-directional approach of the IK-RRT algorithm the search tree is much smaller compared to the results of the J^+-RRT approach (Fig. 6).

The comparison of the average performance of the two proposed planners J^+-RRT and IK-RRT can be seen in table 3. Both planners are queried 100 times with a random object position in front of the robot. The task was to find a collision-free trajectory to one of the 15 associated grasps in two scenes, one with and one without an obstacle. Furthermore both planners take care of self-collisions and collisions with the target object. The results of table 3 point out that the IK-RRT approach is much faster than the J^+-RRT algorithm.



(a) An empty scene. (b) A scene with an obstacle.

Fig. 7 The results of the IK-RRT planner. The solution is marked blue, the optimized solution is shown in green.

Algorithm 3: IK-RRT(q_{start},p_{obj},gc)

```
1  RRT1.AddConfiguration(q_start);
2  RRT2.Clear();
3  while (!TimeOut()) do
4      ExtendRandomly(RRT1);
5      ExtendRandomly(RRT2);
6      if (#IKSolutions == 0 || rand() < p_IK) then
7          grasp ← GetRandomGrasp(gc);
8          p_target ← ComputeTargetPose(p_obj, grasp);
9          q_IK ← ComputeIK(p_target);
10         if (q_IK != NULL & !Collision(q_IK)) then
11             RRT2.AddConfiguration(q_IK);
12     end
13     q_r ← GetRandomConfiguration();
14     if (RRT1.Connect(q_r) & RRT2.Connect.(q_r)) then
15         Solution ← BuildSolutionPath(q_r);
16         return PrunePath(Solution);
17     end
18 end
```

Table 3 The performance of the proposed planning approaches.

	Without Obstacle Avg Runtime	With Obstacle Avg Runtime
J^+-RRT	2 032 ms	18 390 ms
IK-RRT	140 ms	480 ms

5 Dual Arm Motion Planning for Re-Grasping

To plan a re-grasping motion with two arms, two problems have to be solved. The configuration for handing off the object from one hand to the other hand must be determined. This configuration must bring the object, which is grasped with one hand, to a position where the other hand can apply a feasible grasp. This search also includes choosing which grasp should be applied with the second hand. The configuration is only valid if there are no collisions between the arms, the environment, the object and the robot. Furthermore there must exist a collision-free trajectory which brings the arm with the attached object and the other arm to the re-grasping position.

5.1 Dual Arm IK-Solver

If the robot should re-grasp or hand-off an object, the search for a valid re-grasping configuration includes a collision-free object pose and a valid and collision-free IK-solution for both arms. This leads to a 23 dimensional IK problem, where the combination of the 6D object pose, three hip joints and 7 DoF for each arm results in a 23 dimensional solution vector.

5.2 Random Sampling

To find a reachable object pose in the workspace of the robot, the 6D pose of the object and the configuration of the three hip joints can be sampled randomly until a call of the IK-solver is successful for one of the poses P_k^o. Therefore the Cartesian position of the object is limited to the extend of the reachable workspace and the orientation part does not have any restrictions.

5.3 Reachability Space

Since the computational costs of IK-solver calls could be high, the search for feasible object poses can be sped up by the use of reachability spaces. During the IK

search loop, the analytic 7-DoF IK-solvers are only called, if the IK-probability of at least one left and at least one right grasping pose in the corresponding reachability space is above a threshold. If the IK-probability is below that threshold, the random generated hip configuration and object pose are discarded and a new sample is generated. If the the IK-probability is high enough it is likely that the costly IK-Solver calls will succeed and that the pose is valid.

5.4 Gradient Descent in Reachability Space

For further speedup we propose a gradient descent approach which can be used to optimize the search for a graspable object pose. If an object pose was found, where the corresponding reachability space entry lies above a threshold, we apply a search for a local maximum. This is done by checking the neighbor voxels of the reachability space. If there is a voxel with a higher reachability space entry and the new pose is collision-free, the object 6D position is moved toward this voxel by the extend of the corresponding dimensions of a voxel. The new position then lies inside the voxel with the higher reachability entry. This is repeated until there are no neighbors with higher entries which means the position is at a local maximum of the discretized reachability distribution.

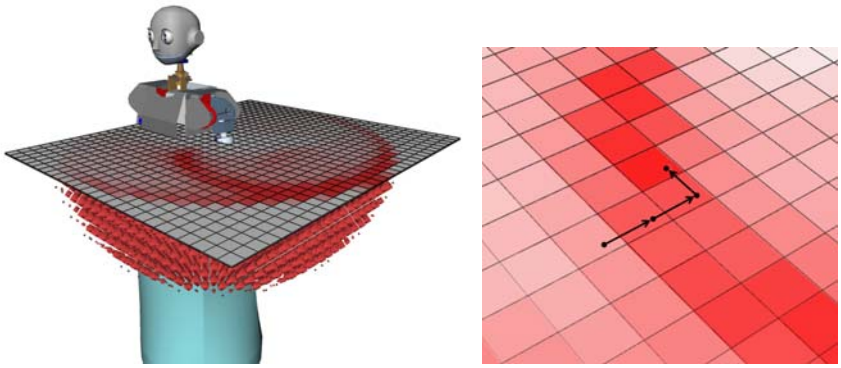
If a feasible object pose for re-grasping is needed, the gradient descent approach can be used with two reachability spaces by searching a local maximum for the left and the right grasping pose. Therefore the sum of both reachability space entries is optimized by moving the object pose until a local maximum is reached.

To avoid loosing the probabilistic completeness by applying the discretized reachability space and the gradient descent approach, these extensions to the original algorithm are only used with some probability during the search loop. Thus, the theoretical behavior of the IK-solvers remain untouched while the performance can be considerably increased.

The resulting run time of the the dual arm IK-solvers are shown in table 4. The IK-solver returns a valid object position and the corresponding joint configuration for the hip and both arms. In this configuration the object and the robot are in a collision-free state and a grasp can be applied for the left and the right hand (row 1). The second row shows the performance of the IK-solver when the object is already grasped with one hand.

Table 4 Performance of the Dual Arm Grasp-IK-Solvers.

	Without Obstacle		With Obstacle	
	Avg Runtime	# IK calls	Avg Runtime	# IK calls
Flexible grasp selection	47 ms	3.3	161 ms	6.5
Object grasped with left hand	162 ms	3.2	220 ms	4.3



(a) (b)

Fig. 8 (a) A 2D view of the reachability space of ARMAR-III. (b) The 2D projection of a gradient descent optimization. The color intensity is proportional to the probability that a pose inside the voxel is reachable.

5.5 Dual Arm J^+-RRT

The Dual Arm J^+-RRT is an extension of the J^+-RRT approach presented in section 4.3.1.

Instead of defining the target by a fixed workspace pose and a set of grasps, the object is attached to a hand and thus the target is implicitly defined by the set of grasps. These grasping poses lead to a set of transformations between the both hands, defining all dual arm configurations for a re-grasping procedure. The *Extend-ToGoal* part of the J^+-RRT approach (see Alg. 1) has to be adapted for the dual arm algorithm. Instead of moving one arm toward a fixed goal pose, the two end effectors are moved toward each other in order to produce a configuration where the object can be grasped with both hands. The *DualArmExtendToGoal* part of the algorithm selects a random grasp and the configuration with the smallest distance between the two end effector poses and tries to move both arms toward a re-grasping pose. This is done by alternately moving the arms toward the corresponding goal poses in workspace. Thus the pseudoinverse Jacobians are calculated for every step and sample configurations are generated. These samples are tested for collision and violations of joint limits and added to the RRT. If a re-grasping pose can be reached a solution to the planning problem was found, otherwise the chosen RRT nodes are marked in order to exclude them for further goal extension steps.

Algorithm 4: $DualArmExtendToGoal(RRT, gc)$

```
1  grasp ← GetRandomGrasp(gc);
2  n ← GetNodeMinDistanceTCPs(RRT);
3  while (!Timeout()) do
4      n ← MoveLeftArm(n, grasp);
5      if (!n) then
6          return NULL;
7      n ← MoveRightArm(n, grasp);
8      if (!n) then
9          return NULL;
10     if (HandOffPoseReached(n, grasp)) then
11         return BuildSolutionPath(n);
12 end
```

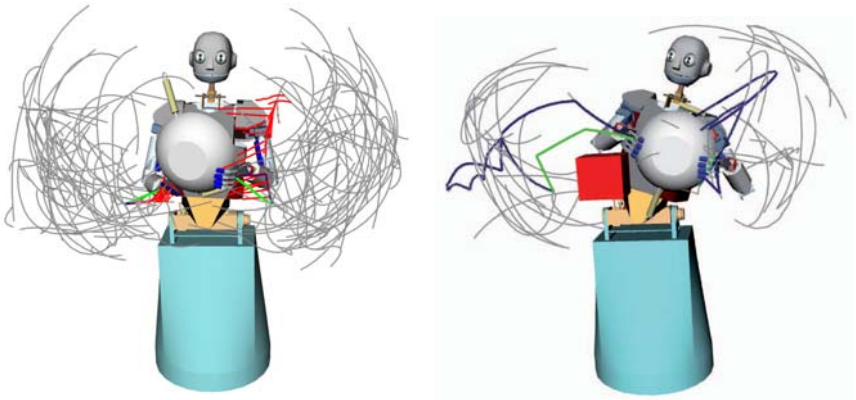
Algorithm 5: MoveLeftArm($n, grasp$)

```
1  p_left ← TCPLeft(n);
2  p'_left ← TargetPoseLeft(grasp);
3  Δ_p ← p'_left − p_left;
4  Δ_q ← J+(q_left) * LimitCartesianStepSize(Δ_p);
5  q_left ← q_left + Δ_q;
6  if (Collision(q_left) || !InJointLimits(q_left)) then
7      return NULL;
8  return BuildNewConfigurationLeft(n, q_left);
```

5.6 Dual Arm IK-RRT

With the IK-solver methods of section 5.1 it is possible to generate feasible configurations for a re-grasping procedure. The search for this configurations can be included in a RRT-based planner like described in section 4.3.3. The dual arm IK-solver is used to generate IK solutions during the planning process. These IK solutions include a valid pose of the object with the corresponding joint configuration of hip and both arms for grasping the object with both hands. The algorithm 3 has to be adapted slightly to include the Dual Arm IK-solver. Instead of a predefined object pose, the object is attached to the kinematic structure of one arm and thus the IK-solver just operates on the set of feasible grasps. The resulting Dual Arm IK-RRT planner can be used for building collision-free re-grasping trajectories in cluttered environments.

The result of the dual arm re-grasp planners are shown in table 5. The planners where queried 100 times and the average planning time was measured in scenes with and without an obstacle. In this evaluation the wok is grasped with the left hand and a re-grasping trajectory is searched. Again, the Dual Arm IK-RRT planner is much faster than the Dual Arm J^+ approach because of the possibility to take advantage of the bi-planning approach.



(a) The re-grasping motion is planned with the Dual Arm J^+-RRT. The red parts are generated by the *ExtendToGoal* part of the algorithm.

(b) Dual Arm IK-RRT: The wok is grasped with the left hand and the collision-free solution trajectory results in a re-grasping configuration.

Fig. 9 The results of the Dual Arm J^+ and the Dual Arm IK-RRT planner. The solution is marked blue, the optimized solution is shown in green.

Table 5 Performance of the dual arm re-grasping planners.

	Without Obstacle Avg Runtime	With Obstacle Avg Runtime
J^+-RRT	1 662 ms	5 192 ms
IK-RRT	278 ms	469 ms

5.7 Planning Hand-off Motions for two Robots

The proposed algorithms can be used to generate collision-free re-grasping motions for two robots. Instead of considering two arms of one robot, two arms of two different robot systems can be used as input for the planning algorithms.

A result of such a re-grasping motion can be seen in figure 10. The performance of the two arm hand-off planning algorithms is similar to the one robot case. From the algorithmic point of view the only difference between the one robot and the two robot problem are the additional hip joints of the second robot.

5.8 Experiment on ARMAR-III

In this experiment ARMAR-III is operating in a kitchen environment where the partly opened cabinet and a box are limiting the operational workspace of the robot. A planner for dual-arm re-grasping is used to find a hand-off configuration and to plan a collision-free hand-off motion for both arms. The resulting trajectory moves

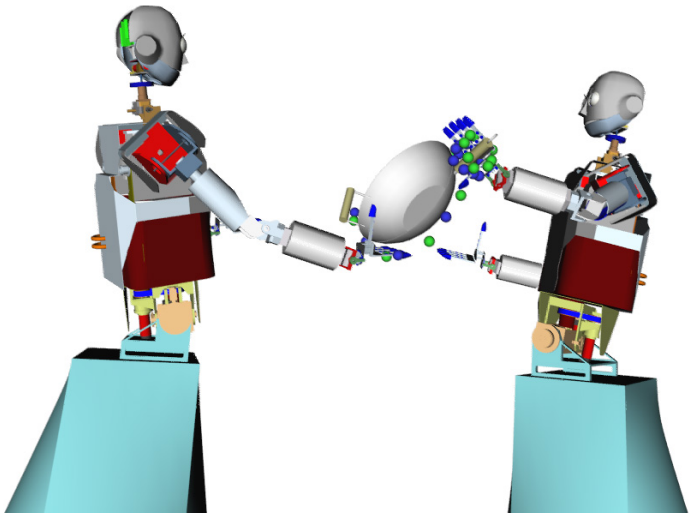


Fig. 10 A hand-off configuration for two robots.

both hands and the plate that is already grasped with the right hand, to the hand-off position and after re-grasping the arms are moved to a standard pose. This real world experiment shows how the dual-arm re-grasping planners enable the humanoid robot ARMAR-III to hand-off objects from one hand to the other.

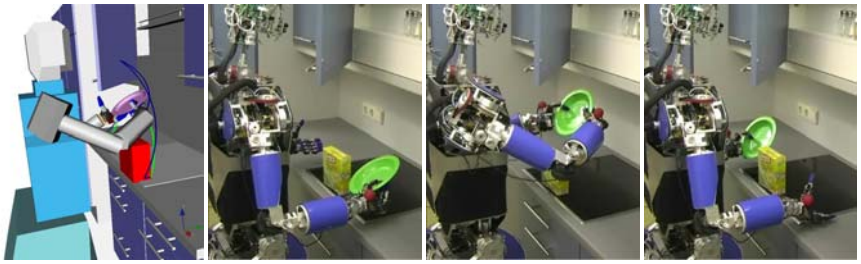


Fig. 11 The humanoid robot ARMAR-III is re-grasping a plate in the kitchen.

6 Adaptive Planning

Since a complex robot system has many degrees of freedom, a planner considering all the joints of the robot, could suffer from the high dimensionality of the configuration space. A RRT-based planner using standard techniques to generate a motion trajectory for a humanoid robot with 43 DoF, like ARMAR-III [2], is not able to

find solutions in reasonable time. The 43-dimensional C-space is not suitable for searching collision-free paths, even when using large step sizes for approximation.

A humanoid robot has several subsystems, like arms, hands and a head, which should be involved into the planning process. In [28], an approach is presented where the number of active joints changes dynamically in order to adapt the volume of the reachable workspace. In [27], a multi-level planning scheme is presented where the planning complexity is iteratively increased by adding non-holonomic constraints at each planning level. The planner in [32] is able to automatically adjust 4 DoF of a humanoid robot depending on the detected environmental situation. This planning scheme strongly depends on the situation detecting which can be difficult if many joints are used for planning. In [10], a multi-level planner is presented, which starts with a low C-space resolution and increases the sampling resolution to finer levels when a coarse solution was found.

The idea of changing the number of DoF during the planning process is picked up for the adaptive planning approach [30]. The proposed planner benefits from the partly reduced dimensionality of the configuration space since free space that has to be covered by the RRT is limited. The general planning approach can be used to build a planner which unites coarse and fine planning tasks. E.g. navigation planning usually considers a low dimensional C-space for searching a collision-free trajectory for positioning the robot, but reaching or grasping tasks need a more complex selection of joints for planning. The adaptive planner for ARMAR-III combines the coarse search for positioning the platform with the finer levels of planning reaching motions for an arm and the dexterous motions for planning the movements of the five finger hand. As shown in the experiments, the planning times could be noticeable decreased and the resulting planner is fast enough for the use on a real robot platform.

6.1 Adaptively Changing the Complexity for Planning

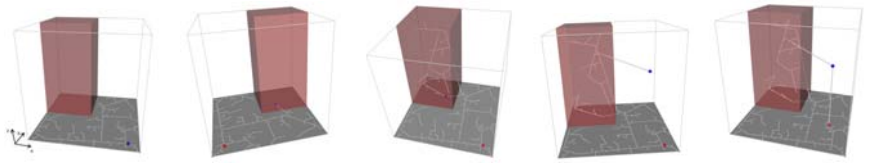
To accelerate the planning process, we want to introduce an adaptive RRT-based planning scheme which is able to change the dimensionality of the C-space adaptively. This concept is implemented for unidirectional and bi-directional planners. To explain the algorithm, first the unidirectional method is described, the bi-directional planner is introduced in section 6.4.

To configure the planner, the workspace is divided into *Planning Areas*, which represent an implicit knowledge of the planning problem. The definition in workspace allows it to easily define and adjust the structure of the these areas. For each area the corresponding set of joints, the *Planning Levels*, are defined.

6.2 A Three Dimensional Example

In figure 12 (a)-(e), a simple three dimensional example of the adaptive planning algorithm is shown. In this example the workspace is equal to the C-space, which means that a configuration $\{x, y, z\}$ leads to a similar workspace position. The effect of increasing and decreasing the *Planning Areas* and *Planning Levels* is visualized. The complete workspace except the red box represents the planning area 1 with the corresponding planning level 1, which includes the two C-space dimensions x and y. The red box represents planning area 2 with the planning level 2, covering the complete C-space.

In figure 12 (a) the search tree grows in planning level 1 until a new configuration falls in planning area 2 (Fig. 12 (b)). When a new configuration lies in planning area 2, the planning level is changed to planning level 2, which means that from now on the search tree is extended in all three C-space dimensions (Fig. 12 (c)). In figure 12 (d) the new configuration lies outside the current planning area and thus the planning level should be decreased. For this reason, a path to a configuration in the next lower planning level is searched and the planning goes on in planning level 1 (Fig. 12 (e)).



(a) (b) (c) (d) (e)

Fig. 12 Adaptive Planning: A three dimensional example.

6.3 Adaptive Planning for ARMAR-III

To use the adaptive planning scheme for the humanoid robot ARMAR-III, the general approach is adjusted. The use of kinematic subsystems of ARMAR-III allows an easy and straight-forward definition of planning areas with corresponding sets of joints.

6.3.1 Kinematic Subsystems

To divide the planning problem into smaller subsets, where the use of a RRT-based planning algorithm is more promising, we define different subsystems of the robot. These subsystems are robot specific and like the kinematic structure they have to be defined once for a system.

Table 6 Subsystems of ARMAR-III.

Subsystem	*Involved Joints*	*# Joints*
Platform	Translation x,y, Rotation	3
Torso	Pitch, Roll, Yaw	3
Right Arm	Shoulder 1,2,3, Elbow	4
Right Wrist	Wrist 1,2,3	3
Right Hand	Thumb 1,2, Index 1,2, Middle 1,2, Ring, Pinkie	8
Left Arm	Shoulder 1,2,3, Elbow	4
Left Wrist	Wrist 1,2,3	3
Left Hand	Thumb 1,2, Index 1,2, Middle 1,2, Ring, Pinkie	8
Head Neck	Tilt, Pitch, Roll, Yaw	4
Head Eyes	Eyes Tilt, Eye Pan Right, Eye Pan Left	3

With these subsystems for the robot, we are able to reduce the complexity of the planning process by considering only the systems that are needed for a given planning task. The planning framework decides which systems are included in the planning process and configures the planning algorithms automatically. For example, the task of grasping an object out of the cupboard, may need the subsystems *Platform, Torso, Right Arm, Right Wrist* and *Right Hand* to get involved for planning. The chosen subsystems result in a 21 dimensional configuration space which is used for searching a path in C_{free}. The joints which are not considered, remain in their standard pose and can be adopted in a post-processing step, e.g. the head can be adjusted to see the target.

6.3.2 The Approach

The planner starts with a low dimensional C-space in order to move the robot in the environment to a position that is near to the target object. For this positioning in the environment the detailed kinematic structures of the robot (e.g. the finger joints) are not considered, since they do not support the planning process in this rough planning step. If the planner has found a path in C-space which brings the robot near to the target object, or if the reduced planning failed, more joints are used to allow a more detailed planning. Which joints or subsystems are chosen to increase the complexity depends on the planning task. This planning scheme is performed until a solution is found or the full complexity of the robot is reached.

A parameter, which directly affects the planning time, is $d_{PlanningArea}$, the minimum workspace distance of the Tool Center Point (TCP) to the target configuration. When the TCP distance falls below this value the planner changes to the next level and increases the number of involved subsystems. For this reason the TCP workspace distance to the goal configuration is calculated for each new configuration that is added to the RRT.

To avoid a manual definition of $d_{PlanningArea}$ for each planning level, the minimum TCP distance for the first level is set to the doubled maximum reaching distance (in

case of AMRAR III, this is 1176 mm) and the following values are calculated by iteratively bisecting $d_{PlanningArea}$. The extent of the planning levels are shown in Fig. 13 for the subsystems *Platform, Torso, Right Arm, Right Wrist* and *Right Hand*.

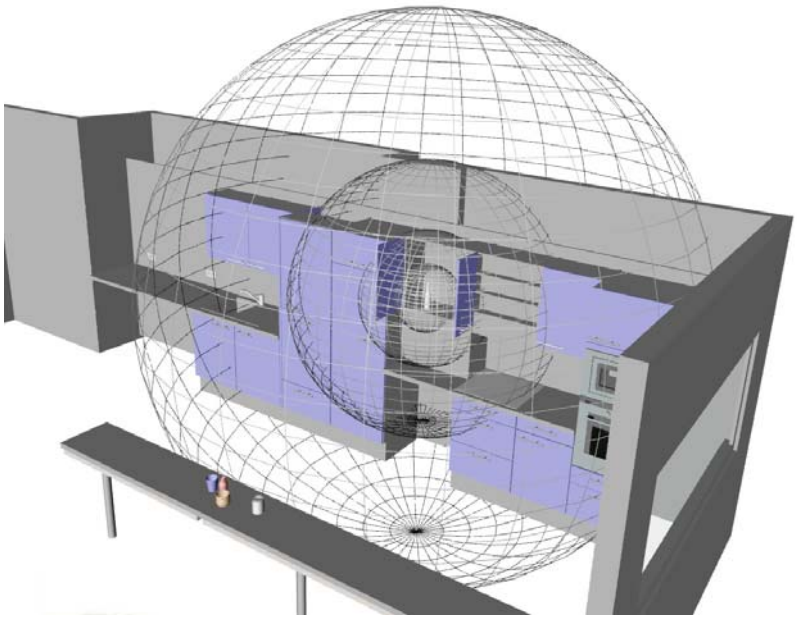


Fig. 13 The extent of the planning levels around a target object for five subsystems of ARMAR-III.

Table 7 shows the extent of the planning levels for five subsystems of ARMAR-III. The value for the *Right Hand*-subsystem is set to zero, since the number of joints could not increased any more and the planner should go on until a global solution is found.

Table 7 Extent of the planning levels for ARMAR-III

Subsystem	$d_{PlanningArea}$ *(mm)*
Platform	2 352
Torso	1 176
Right Arm	588
Right Wrist	294
Right Hand	0

Algorithm 6: AdaptivePlanning(c_{start}, c_{goal})

```
1  PlanningArea ← 1;
2  RRT.addConfig(c_start);
3  while (!TimeOut()) do
4      c_r ← RandomConfig(PlanningArea);
5      c_nn ← GetNearestNeighbor(RRT, c_r);
6      if (RRT.Connect(c_nn, c_r) && !IsInPlanningArea(c_r, PlanningArea)) then
7          PlanningArea ← ChangePlanningArea(c_r);
8      if (IsMaximalPlanningArea(PlanningArea) && rand() < p_goal) then
9          c_nn ← GetNearestNeighbor(RRT, c_goal);
10         if (RRT.Connect(c_nn, c_goal) then
11             return BuildSolution();
12     end
13 end
```

Algorithm 7: RandomConfig(*PlanningArea*)

```
1  for (i ← 0; i < RRT.dimension; i ← i + 1) do
2      if (IsDimensionInPlanningArea(i, PlanningArea)) then
3          c[i] ← RandomValue(DimBoundaryLo[i], DimBoundaryHi[i]);
4      else
5          c[i] ← StandardValue(i);
6  end
7  return c;
```

6.4 Extensions to Improve the Planning Performance

6.4.1 Randomly Extending Good Ranked Configurations

As described in [7] each node in the RRT can hold a ranking of it's configuration, which can be used to support the planning. The ranking is calculated as the workspace distance of the TCP from the current to the goal configuration. To improve the planning performance, the planner sometimes chooses one of the last k best ranked nodes and does an extension step to an arbitrary direction. To avoid trapped situations, failures are counted and configurations with many failed extend steps are removed from the ranking.

6.4.2 Bi-Planning

The planning algorithm should find a trajectory for a given start and goal configuration of the robot. In most cases the target configuration is more critical than the

start configuration, since typical planning tasks will generate grasping or reaching trajectories. Hence the target configurations often result in low obstacle distances and thus in limited free space to operate. These situations are difficult for sampling-based planners, since only short paths in C_{free} can be found [13, 8]. To support the RRT-based planner, a bi-directional planning approach can be used, which builds up trees from the start and goal configuration and iteratively tries to connect them [16, 25].

The adaptive planner has to be adapted slightly to support the bi-directional search. The forward tree which starts the search from the start configuration is build like in the unidirectional case. The second tree which starts at the goal configuration needs some changes in the algorithm:

The planner starts with full resolution from the goal configuration. Each new configuration c_n, that is added to the RRT, is checked whether $d_{tcp-Target}$, the TCP workspace distance of the TCP between the new and the goal configuration, is greater than the current extent of the planning level. In this case the planning level is decreased and the further planning is done in a C-space with less dimensions.

With this extension of the adaptive planning algorithm, the bi-planner builds up two trees, one starting at the start configuration and one inverted tree starting at the goal configuration. The bi-planning loop generates random configurations and tries to connect them to both trees. If this connection succeeds for both trees, a global solution was found. If the connection fails, the trees are extended as long as possible until a collision occurs. This behavior supports the covering of the free configuration space and leads to a fast and robust planning algorithm.

6.4.3 Focusing the Search to the Area of Interest

By defining the planning levels, areas of interests with different extent are defined around the target object. The planning process can be accelerated by focusing the search to these areas. Since the global goal trajectory is unknown, the search should not be limited to one area, otherwise a solution can be overseen. To achieve a focus on a target area, an adaption of the classical RRT-Connect and RRT-Extend algorithms is proposed. The standard extension of the C-space tree will connect a random configuration c_r to c_{nn}, the nearest neighbor of the existing tree. This behavior guarantees a uniform coverage of the C-space, which is a helpful property for a global planning problem, but in a locally bounded planning task the planning time will increase, since many areas of the C-space which are not important for planning are unnecessarily investigated. To emphasize a specific area in workspace an adaption of the *GoalZoom* [19] algorithm is used. Sometimes, instead of an arbitrary configuration c_r, a more promising configuration c_{zoom} next to the goal configuration c_{goal} is used to extend the tree. The maximum distance d_{zoom} between c_{goal} and the randomly chosen c_{zoom} depends on the current planning level. To avoid the introduction of another parameter, d_{zoom} is defined in workspace and set to the current planning extent value $d_{PlanningArea}$. This means that a random position c_{zoom}, used to bias the search toward the goal, has to hold the constraint, that the maximum

workspace distance d_{WS} is smaller than $d_{PlanningArea}$ (Eq. 9).

$$d_{WS}(\mathbf{c}_{goal}, \mathbf{c}_{zoom}) < d_{PlanningArea} \tag{9}$$

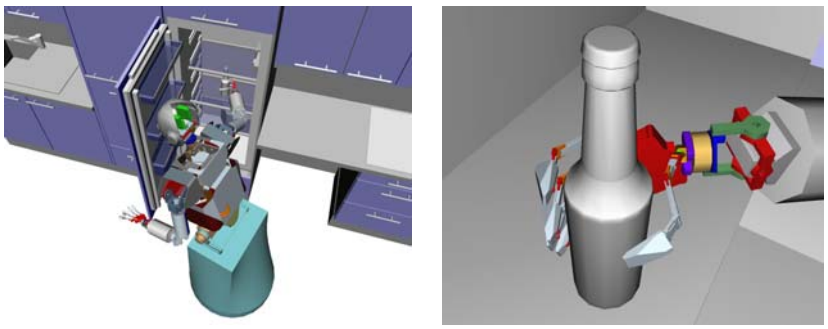
In the bi-directional case the enhancement works as follows: The randomly chosen configuration, for which a connection to both trees is tested, sometimes is chosen in the surrounding of the start or goal configuration, whereby the distance depends on the planning level.

6.5 Experiments

For evaluation the simulation environment of the humanoid robot ARMAR-III is used. The robot model has 43 DoF and for each limb there are two 3D models, one for visualization and one simplified model for collision checking purposes. The robot is operating in a kitchen environment, which is also modeled with full and reduced resolution for visualization and collision checking as described in section 2.

The starting position of the robot is located outside the kitchen and a trajectory for grasping an object is searched. In this experiment the target object is placed inside the fridge (Fig. 14(a)). For this task, the planner uses the subsystems *Platform, Torso, Right arm, Right Wrist* and *Right Hand*. In our test setup the subsystem for the right hand consists of 6 instead of 8 joints because the two middle and the two index finger joints are coupled and thus are counted like one DoF. The overall number of joints used for planning and therefore the dimensionality of the C-space is 19.

We make the assumption that a higher level task planning module has already calculated a goal position for grasping the object, thus c_{goal}, the target in C-space, is known.



(a) distance view (b) narrow view

Fig. 14 Target position of the planning task.

The maximum workspace stepsize ε_{ws} was set to 30*mm* for collision checking on path segments. Since ε_{ws} is a worst-case approximation, the real step sizes are significant lower. When adding a path segment to the RRT, intermediate nodes are generated in order to support the nearest neighbor search. These nodes are generated with a maximal workspace distance of 90*mm* in all tested planners. [2]

6.5.1 Unidirectional Planning

A first evaluation of the planning problem described above, has been done by using a RRT-Connect one-way planner (A). The planner builds up a tree, covering the free C-space, and randomly tries to connect this tree to c_{goal}. The results point out, that the planner often has difficulties to escape from local minima and find a correct solution path to the goal configuration. These local minima may occur from situations when the right hand gets under the cupboard or when the position of the finger joints is disadvantageous. In more than 60% of the test cases, the planning was stopped, because a time limit of 10 minutes or a limit in RRT nodes (100.000) was exceeded. If the planner did not find a solution within these limitations, it is not sufficient for the use in a real-world scenario.

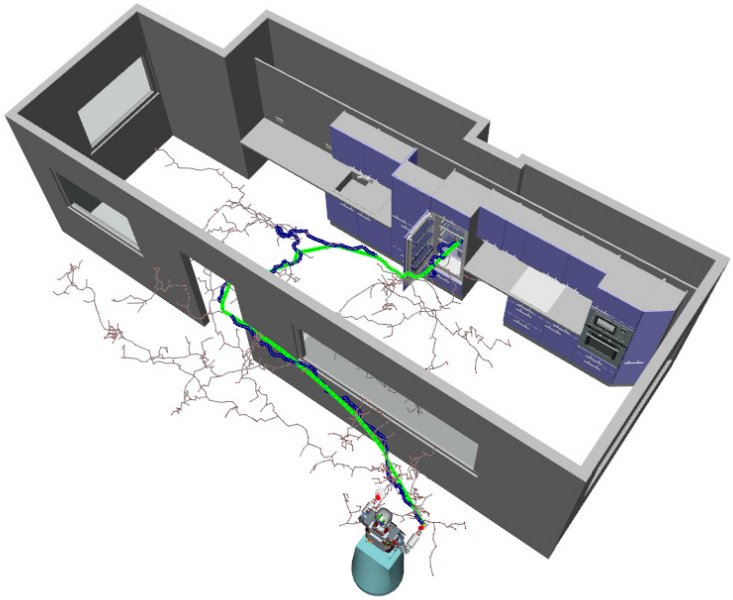


Fig. 15 RRT Bi-Planning: The search tree with original (blue) and optimized (green) TCP paths.

[2] These tests have been carried out on an Intel Core 2 Duo Linux System with 2.16 GHz.

The adaptive planning (B) performs better than the RRT-Connect method. More test runs succeeded and the planning time can be reduced by over 60 %. Nevertheless a lot of planning cycles failed.

For further improvement of the adaptive planning, the *GoalZoom*-enhancement and the random extend steps of section 6.4 have been implemented and tested. The planner (C) benefits from these improvements and therefore the planning time and the number of failed planning runs can be decreased.

Table 8 Unidirectional planning

	Planning Succeeded	Avg. Planning Time (success)	Avg. Number of RRT Nodes	Avg. Number of Collision Checks
RRT-Connect (A)	37.5 %	98.6 s	30 907	252 605
Adaptive Planning (B)	43.5 %	31.3 s	10 089	83 017
Extended Adapt. Pl. (C)	57.5 %	14.2 s	5 123	40 522

6.5.2 Bi-Directional Planning

The RRT-Connect planner (A) often fails due to local minima in which the search frequently gets stuck. To support the planning, the bi-planning approach was implemented for the RRT-Connect algorithm (D). The planning succeeded in every test run and the average runtime was measured with three seconds (table 9 row D).

A planned RRT with original (blue) and optimized (green) solution paths are depicted in Fig. 15.

Although the adaptive planner (B) achieves better results than the RRT-Connect planner (A), there are also settings in which the planning fails. The results of the bi-directional RRT-Connect planner (D) point out, that the planning can benefit a lot from building up two search trees. The adaptive bi-planner (E) was implemented and tested as described in section 6.4. The adaptive reduction of the subsystems combined with a bi-planner results in an average planning time of 477 milliseconds (table 9 row E).

The *GoalZoom*-enhancement, described in section 6.4, which noticeable increased the planning performance for unidirectional planners, just decreases the planning time slightly for the adaptive bi-directional planner (F). As shown in table 9 the average planning time decreases from 477*ms* to 423*ms*. Fig. 16 shows a typical RRT with original and reduced solution paths for this configuration of the planning algorithm.

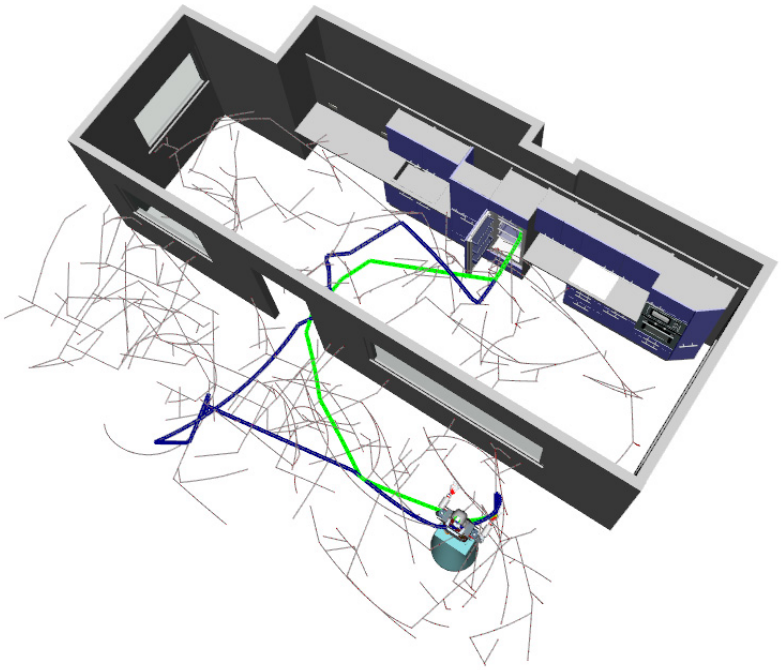


Fig. 16 Adaptive Bi-Planning: The search tree with original (blue) and optimized (green) TCP paths.

Table 9 Bi-directional planning

	Planning Succeeded	Avg. Planning Time	Avg. Number of RRT Nodes	Avg. Number of Collision Checks
RRT-Connect (D)	100.0 %	3 037 ms	784	4 802
Adaptive Planning (E)	100.0 %	477 ms	477	1 443
Extended Adapt. pl. (F)	100.0 %	423 ms	388	1 181

7 Conclusion

A service or humanoid robot can benefit from a planning framework which offers specialized approaches for motion planning. The concept presented here, includes multiple planning algorithms which can be selected and parametrized by the framework. The diversity of applications and the use in a human-centered environment necessitates fast and robust algorithms for a wide range of planning problems. A generalized approach, covering all requested needs, is desirable, but with current techniques it is not realistic to achieve an implementation that is fast enough for real world applications. For this reason we presented and evaluated several specialized algorithms which can be used within the planning framework and thus a fast system for a wide range of planning problems can be constructed.

The presented approaches can be used to include the search for Inverse Kinematics solutions in the planning process. For this reason a set of feasible grasps are stored with each manipulation object. The proposed J^+-RRT and IK-RRT planners are not limited to one specific goal configuration and thus trajectories for the often needed task of grasping an object can be generated in an fast and robust manner. The dual arm extensions of the J^+-RRT and the IK-RRT algorithms offer a possibility for efficiently planning dual arm motions for grasping, re-grasping and hand-off procedures. As shown in the experiments, the algorithms can be applied for two arms of one or two humanoid robots. The performance evaluations pointed out the possibility of using these planners on real robot platforms.

The adaptive planning approach can be used to efficiently unite different planning problems like navigation, reach and grasp planning. The algorithms have been evaluated with a common planning problem for a humanoid robot. As test case a grasping task in a kitchen environment was chosen where the planners had to find trajectories for 19 DoF of the robot ARMAR III. As a reference a standard RRT planner was used for which the poor performance and the high number of unsuccessful planning cycles have been overcome by using a bi-planning approach. The results of the adaptive planner point out that the planning time can be noticeable decreased if the planning task and the used subsystems of the robot are known. The use of several extensions combined with the adaptive approach leads to a planner which is able to find solutions for a 19 DoF grasping task in about half a second on average. The planning performance is sufficient for real world applications and the use on a hardware platform.

8 Acknowledgments

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

References

1. Asfour T, Dillmann R (2003) Human-like motion of a humanoid robot arm based on a closed-form solution of the inverse kinematics problem. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
2. Asfour T, Regenstein K, Azad P, Schröder J, Bierbaum A, Vahrenkamp N, Dillmann R (2006) Armar-III: An integrated humanoid platform for sensory-motor control. In: IEEE-RAS International Conference on Humanoid Robots (Humanoids 2006), pp 169–175
3. Asfour T, Azad P, Vahrenkamp N, Regenstein K, Bierbaum A, Welke K, Schröder J, Dillmann R (2008) Toward humanoid manipulation in human-centred environments. Robot Auton Syst 56(1):54–65

4. Badler NI, Phillips CB, Webber BL (1993) Simulating Humans: Computer Graphics Animation and Control. Oxford University Press, New York, Oxford
5. Berenson D, Diankov R, Nishiwaki K, Kagami S, Kuffner J (2007) Grasp planning in complex scenes. In: IEEE-RAS International Conference on Humanoid Robots (Humanoids07)
6. Berenson D, Srinivasa S, Ferguson D, Collet A, Kuffner J (2009) Manipulation planning with workspace goal regions. In: IEEE Intl Conf. on Robotics and Automation (ICRA)
7. Bertram D, Kuffner J, Dillmann R, Asfour T (2006) An integrated approach to inverse kinematics and path planning for redundant manipulators. In: IEEE International Conference on Robotics and Automation, pp 1874–1879
8. Boor V, Overmars M, Stappen A (1999) The Gaussian sampling strategy for probabilistic roadmap planners. In: IEEE International Conference on Robotics and Automation, pp 1018–1023
9. Canny JF (1988) The complexity of robot motion planning. MIT Press, Cambridge, MA, USA
10. Chen PC, Hwang YK (1998) SANDROS: A dynamic search graph algorithm for motion planning. IEEE Transactions on Robotics & Automation 14(3):390–403
11. Diankov R, Ratliff N, Ferguson D, Srinivasa S, Kuffner J (2008) Bispace planning: Concurrent multi-space exploration. In: Robotics: Science and Systems
12. Geraerts R, Overmars MH (2005) On improving the clearance for robots in high-dimensional configuration spaces. In: IEEE/RSJ Intl Conf. on Intelligent Robots and Systems, pp 4074–4079
13. Hsu D, Latombe JC, Motwani R (1999) Path planning in expansive configuration spaces. International Journal of Computational Geometry and Applications 9(4/5)
14. Jimnez P, Thomas F, Torras C (2001) 3D collision detection: A survey. In: Computers and Graphics, pp 269–285
15. Kee D, Karwowski W (2002) Analytically derived three-dimensional reach volumes based on multijoint movements. Human Factors: The Journal of the Human Factors and Ergonomics Society 44:530–544(15)
16. Kuffner J, LaValle S (2000) Rrt-connect: An efficient approach to single-query path planning. In: IEEE Int'l Conf. on Robotics and Automation (ICRA'2000), San Francisco, CA, pp 995–1001
17. Larsen E, Gottschalk S, Lin MC, Manocha D (2000) Fast proximity queries with swept sphere volumes. Tech. rep., Department of Computer Science, University of North Carolina
18. LaValle S, Kuffner J (2000) Rapidly-exploring random trees: Progress and prospects. In Workshop on the Algorithmic Foundations of Robotics.
19. LaValle SM (2006) Planning Algorithms. Cambridge University Press, Cambridge, U.K., available at http://planning.cs.uiuc.edu/
20. LaValle SM, Kuffner JJ (2001) Randomized kinodynamic planning. International Journal of Robotics Research 20(5):378–400
21. Lin M, Gottschalk S (1998) Collision detection between geometric models: A survey. In: IMA Conference on Mathematics of Surfaces
22. Miller AT (2001) Graspit!: a versatile simulator for robotic grasping. PhD thesis, Department of Computer Science, Columbia University
23. Quinlan S (1994) Real-time modification of collision-free paths. PhD thesis, Stanford University
24. Reif JH (1979) Complexity of the mover's problem and generalizations. In: SFCS '79: Proceedings of the 20th Annual Symposium on Foundations of Computer Science (sfcs 1979), IEEE Computer Society, Washington, DC, USA, pp 421–427
25. Sanchez G, Latombe J (2001) A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In: International Symposium on Robotics Research, Lorne, Victoria, Australia
26. Schwartz JT, Sharir M (1983) On the piano movers problem: Coordinating the motion of several independent bodies. In: Int. J. Robot. Res., pp 97–140
27. Sekhavat S, Laumond J, Overmars MH (1998) Multilevel path planning for nonholonomic robots using semiholonomic subsystems. Int J Robot Res 17:840–857

28. Sian NE, Yokoi K, Kajita S, Tanie K (2004) A framework for remote execution of whole body motions for humanoid robots. In: Humanoid Robots, 2004 4th IEEE/RAS International Conference on, vol 2, pp 608–626
29. Vahrenkamp N, Asfour T, Dillmann R (2007) Efficient motion planning for humanoid robots using lazy collision checking and enlarged robot models. In: Intelligent Robots and Systems, IROS
30. Vahrenkamp N, Scheurer C, Asfour T, Kuffner JJ, Dillmann R (2008) Adaptive motion planning for humanoid robots. In: IROS, pp 2127–2132
31. Weghe MV, Ferguson D, Srinivasa S (2007) Randomized path planning for redundant manipulators without inverse kinematics. In: IEEE-RAS International Conference on Humanoid Robots
32. Yoshida E (2005) Humanoid motion planning using multi-level DoF exploitation based on randomized method. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE Computer Society, Edmonton, Canada, pp 3378–3383