Humanoid Motion Planning for Dual-Arm Manipulation and Re-Grasping Tasks

Nikolaus Vahrenkamp* Dmitry Berenson† Tamim Asfour* James Kuffner† Rüdiger Dillmann*

*Institute for Anthropomatics
University of Karlsruhe
Haid-und-Neu Str. 7
76131 Karlsruhe, Germany
{vahrenkamp,asfour,dillmann}@ira.uka.de

†The Robotics Institute
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
{dberenso,kuffner}@cs.cmu.edu

Abstract— **In this paper, we present efficient solutions for planning motions of dual-arm manipulation and re-grasping tasks. Motion planning for such tasks on humanoid robots with a high number of degrees of freedom (DoF) requires computationally efficient approaches to determine the robot's full joint configuration at a given grasping position, i.e. solving the Inverse Kinematics (IK) problem for one or both hands of the robot. In this context, we investigate solving the inverse kinematics problem and motion planning for dual-arm manipulation and re-grasping tasks by combining a gradient-descent approach in the robot's pre-computed reachability space with random sampling of free parameters. This strategy provides feasible IK solutions at a low computation cost without resorting to iterative methods that are often trapped by joint-limits. We apply this strategy to dual-arm motion planning tasks in which the robot is holding an object with the one hand in order to generate whole-body robot configurations suitable for grasping the object with both hands. In addition, we present two probabilistically complete RRT-based motion planning algorithms (J^+-RRT and IK-RRT) that interleave the search for an IK solution with the search for a collision-free trajectory and the extension of these planners to solve re-grasping problems. The capability of combining IK methods and planners are shown both in simulation and on the humanoid robot ARMAR-III performing dual-arm tasks in a kitchen environment.**

I. Introduction

When performing everyday manipulation tasks, such as putting plates in a cabinet or loading a dishwasher, humans often re-grasp the objects they manipulate. Having two arms allows people to reach for an object with one arm and place it with the other, effectively increasing the reachable space without moving in the workspace. If humanoid robots are to exploit their two-armed capabilities, they must possess computationally efficient algorithms for grasping, re-grasping and dual-arm tasks.

Because such robots are meant to operate in cluttered domestic environments, planning algorithms are needed to generate collision-free trajectories. However, planning a reaching or a re-grasping motion requires choosing a feasible grasp pose from multiple grasp poses defined by a given object and finding a configuration of the robot's joints which places the robot's end-effectors to the selected grasp pose. Thus, the planning algorithm must decide which of the feasible grasp poses should be selected and determine the robot's joint configuration for that pose. In the case of re-grasping, the object position which allows grasping by the second hand must be calculated.

Finding a robot configuration that places the end-effector at a given pose is known as the Inverse Kinematics (IK) problem. Though analytical solution of IK is possible for some manipulators which have no more than 6 DoF[1], the humanoid robot ARMAR-III robot [2] has two 7 DoF arms and a 3 DoF hip.

In this paper, we present a novel IK solver for ARMAR-III, which uses a combination of gradient descent in pre-computed reachability spaces and random-sampling of free parameters (Sec. II and III) and show how to apply our approach to one or two arm queries with fixed or varying object poses. In the case of a varying object pose, the search for a collision-free and graspable object pose is part of the inverse kinematics task and the result consists of a robot configuartion and a 6D object pose. This approach to solving IK is extremely efficient, requiring only a few milliseconds to solve a query as opposed to more time-consuming iterative IK algorithms (e.g. [3]) which are often trapped by joint limits.

This paper also presents two probabilistically complete algorithms for planning reaching and re-grasping motions (Section IV): the J^+-RRT, which is an extension of the single-tree RRT-JT approach[4] and IK-RRT, which is a bidirectional RRT that samples IK solutions while planning. The advantage of the J^+-RRT is that it does not require an IK solver, so it can be used for robots where no efficient IK solver is available, however it usually takes a long time to find a path in cluttered environments. The advantage of the IK-RRT is its low computation cost, however, it requires an efficient IK solver such as the one presented in this paper. In section V we present how both planners are extended to generate collision-free trajectories for dual-arm re-grasping tasks. Simulation and experimental results on the humanoid robot ARMAR-III are shown in section VI.

II. Single Arm IK Solver

To reach and grasp a fixed object with one hand, the IK-problem has to be solved. In the case of ARMAR-III, the operational workspace can be increased by additionally considering the three hip joints of the robot, which leads to a kinematic chain with ten DoF. Our approach to solving the IK problem uses a combination of gradient descent in reachability space and random sampling of free parameters.

A. Randomized IK-Solver

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 6DoF of an arm, a randomized algorithm can be constructed which randomly samples the preceding joints (such as the hip) 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 [5]. 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 selected. In addition to this IK solving it is desirable to consider the joints of the robot's 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 collisions, the solution is rejected and the search is continued.

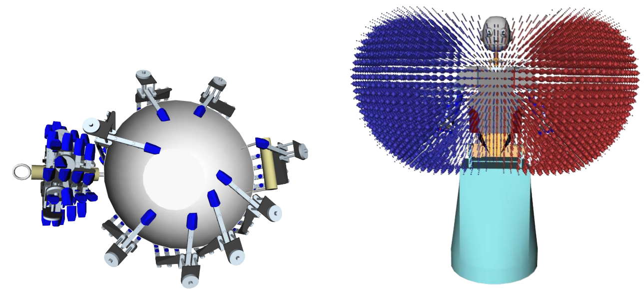
The approach is probabilistically complete, which means if time goes to infinity the algorithm will find a solution if at least one exists. To avoid endless runtimes, the search for an IK solution is stopped after a specific number of tries and it is assumed that there is no valid result. Since this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

B. 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 [6, 7, 8]. 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 spaces can be determined 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 kinemtics to determine the pose of the end effector and thus the corresponding 6D voxel [6]. An analytic approach of generating a representation of the reachability is presented in [9].

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).



(a) (b)

Fig. 1. (a) An object (wok) with predefined grasp positions for the right hand of ARMAR-III. (b) The 3D projection of the reachability spaces for both arms of ARMAR-III.

C. Gradient Descent in Reachability Space

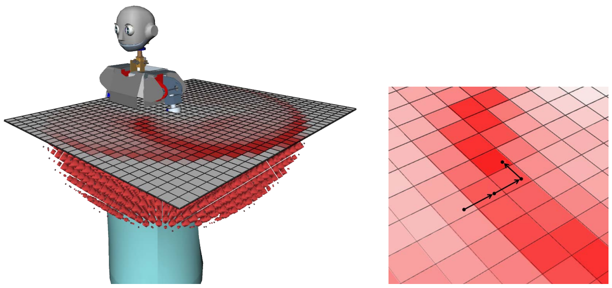
For further speedup we propose a gradient descent approach which can be used to optimize the search for a graspable object pose. If an object pose was found, where the corresponding reachability space entry lies above a threshold, we apply a search for a local maximum. This is done by checking the neighbor 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.

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.

D. 10 DoF IKSolver 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 seven arm joints. This 10 DoF kinematic chain leads to a large reachable workspace and thus enables the robot to perform grasping and manipulation operations without moving the robot's mobile platform.

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. An exemplary result of the IK-solver in a



(a) (b)

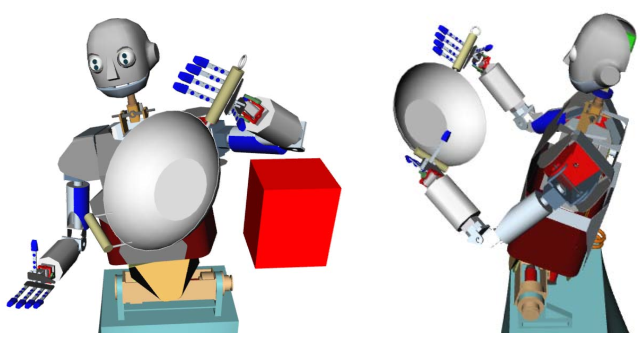
Fig. 2. (a) A 2D view of the reachability space of ARMAR-III. (b) The 2D projection of a gradient descent optimization. The color intensity is proportional to the probability that a pose inside the voxel is reachable.

partly blocked scene is shown in Fig. 3(a). The results of table 1 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 1

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



(a) Single arm IK solver in a scene with an obstacle. (b) Dual-Arm IK-solver in an empty scene.

Fig. 3. Exemplary results of the 10 DoF single arm (a) and the 17 DoF dual-arm IK solvers (b). The IK algorithms provide feasible joint configurations and a collision-free object pose.

III. 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 DoF 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.

A. 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 successfull 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 orientational part does not have any restrictions.

B. Reachability Space

Since the computational costs of IK solver calls could be high, the search for feasible object poses can be speed up by the use of reachability spaces. During the IK search loop, the analytical 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 is discarded and a new sample is generated. If the IK-probability is high enough it is likely that the costly IK-Solver calls will succeed and that the pose is valid.

TABLE 2

PERFORMANCE OF THE DUAL-ARM IK SOLVERS.

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

The resulting run times of the the dual-arm IK-solvers are shown in table 2. 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.A n exemplary result of the dual-arm IK-solver is shown in Fig. 3(b).

IV. MOTION PLANNING FOR SINGLE ARM REACHING

The proposed planning algorithms combine the search for collision free motions with the search for solutions of the IK problem in 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 target configurations is done during the planning process and thus the planning is not limited to an incomplete set of targets.

A. 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 grasping pose P_{grasp} which allows

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

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_k. For each object which should be grasped or manipulated by the robot, a set of feasible grasps is stored in a database. This set hold information about the transformations between the end effector and the final grasping position, the type of grasp, a pre-position of the end-effector and additional grasp quality descriptions. These database entries can be generated automatically (as in [10] or [11]) or manually, like in the following examples. A wok with 15 feasible grasps for the right hand of the humanoid robot ARMAR-III can be seen in Fig. 1(a).

To grasp an object o (located at position T_o) with the end-effector e by applying the grasp g_k from the feasible grasps set gc_o^e, the IK problem for the pose P_k^o has to be solved.

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

It is possible to calculate an IK solution for each pose of each grasp candidate in the database and to use this set of configurations as targets for the planning process. This will lead to a planning scheme where the search for solutions 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 an IK solution which was not considered. 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 pre-calculation has to be done for a large number 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. Note that we have developed similar planning algorithms that take as input continuous regions in the workspace in previous work [12].

B. Jacobian Pseudoinverse-Based RRT (J^+-RRT)

The RRT-JT approach, presented in [4], avoids the explicit search for IK solutions by directing the RRT extensions towards a workspace goal pose. Therefore the transposed Jacobian is used to generate C-Space steps out of a workspace goal direction. The RRT-JT approach can be useful when no IK solver for a robot system is present and only a grasping pose in workspace is known. Since there is no explicit C-space target configuration defined, the approach can not be implemented as a bi-directional RRT and the advantages of the Bi-RRT algorithms can not be applied.

The J^+-RRT is an extension of the RRT-JT approach:

- Instead of the transposed Jacobian, the Pseudoinverse is used to compute goal directed C-space extension steps.
- Multiple workspace goals are defined through a set of feasible grasps.
- Instead of a three dimensional positions, full 6D poses are used as workspace targets.

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

```
RRT.AddConfiguration(q_start);
while (!TimeOut()) do
    ExtendRandomly(RRT);
    if (rand() < p_ExtendToGoal) then
        Solution ← ExtendToGoal(RRT, p_obj, gc);
        if (Solution ≠ NULL) then
            return PrunePath(Solution);
    end
end
```

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

```
grasp ← GetRandomGrasp(gc);
p_target ← ComputeTargetPose(grasp);
q_near ← GetNearestNeighbor(RRT, p_target);
repeat
    p_near ← ForwardKinematics(q_near);
    Δ_p ← p_target − p_near;
    Δ_q ← J⁺(q_near) * LimitCartesianStepSize(Δ_p);
    q_near ← q_near + Δ_q;
    if (Collision(q_near) || !InJointLimits(q_near)) then
        return NULL;
    RRT.AddConfiguration(q_near);
until (Length(Δ_p) > Threshold_Cartesean) ;
return BuildSolutionPath(q_near);
```

The pseudo code of the J^+-RRT planner is given in algorithm 1. The planner is initialized with the starting configuration q_{start}, the workspace pose p_{obj} of the object and a set of feasible grasps ($gc = \{g_0, .., g_k\}$). 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. The *ExtendToGoal* method is called with some probability at each iteration of the planner.

In Fig. 4(a) a resulting RRT of a J^+-RRT planner in an empty scene 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 [13]. 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 (see Alg. 2). The figure shows that the search is focused around the grasping object but in most cases the joint limits and collisions between the hand and the object (wok) prevent the generation of a valid solution trajectory.

C. IK-RRT

To speedup the planning, an IK solver could be used in order to generate goal positions during the planning process. The planner uses as input a set of feasible grasping poses, which, combined with the pose of the object, defines a set of workspace target poses. These poses are used as input for the

IK solver.

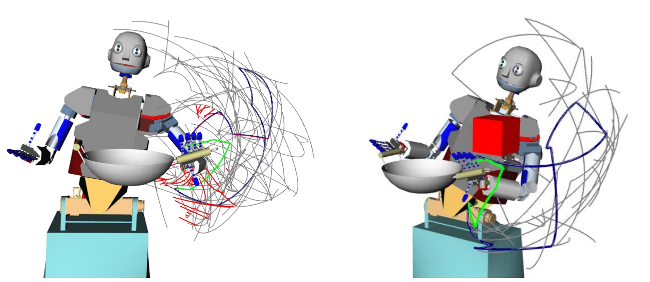
Algorithm 3: IK-RRT(q_{start},p_{obj},gc)

```
RRT1.AddConfiguration(q_start);
RRT2.Clear();
while (!TimeOut()) do
    if (#IKSolutions == 0 || rand() < p_IK) then
        grasp ← GetRandomGrasp(gc);
        p_target ← ComputeTargetPose(p_obj, grasp);
        q_IK ← ComputeIK(p_target);
        if (!Collision(q_IK)) then
            RRT2.AddConfiguration(q_IK);
    else
        q_r ← GetRandomConfiguration();
        if (RRT1.Connect(q_r) & RRT2.Connect.(q_r))
        then
            Solution ← BuildSolutionPath(q_r);
            return PrunePath(Solution);
        end
    end
end
```

The IK-RRT algorithm works as follows:

- Initialization: The forward part of the Bi-RRT algorithm is initialized with a start configuration, the backward tree is empty until an IK solution is found.
- The planning loop grows the two trees and tries 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 perfomed. When a feasible IK configuration q_{IK} is found, it is added to the backward tree and the new node is marked as a solution node.

Since the IK search is probabilistic complete for the set of grasps and the RRT-Connect algorithm is known to be probabilistic complete [14], the IK-RRT approach is probabilistic complete. This means, that as time goes to infinity the algorithm will find a solution if at least one exists.



(a) An empty scene (J^+-RRT). (b) A scene with an obstacle (IK-RRT).

Fig. 4. The results of the J^+ and the IK-RRT planner. The solution is marked blue, the optimized solution is shown in green.

In Fig. 4(b) results of the IK-RRT approach are 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. 4(a)).

V. Motion Planning for Dual-Arm 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 grasped object and the other arm to the re-grasping position.

A. Dual-Arm J^+-RRT

The dual-arm J^+-RRT is an extension of the J^+-RRT approach presented in section IV-B.

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 definied by the set of grasps. These grasping poses lead to a set of transformations between the both hands, defining all dual-arm configurations for re-grasping. The *ExtendToGoal* part of the J^+-RRT approach (see Alg. 1) has to be adapted for the dual-arm algorithm. Instead of moving one arm towards a fixed goal pose, the two end-effectors are moved towards each other in order to produce a configuration where the object can be grasped with both hands. The *DualArmExtendToGoal* part selects a random grasp and the configuration with the smallest distance between the two end-effector poses and tries to move both arms towards a re-grasping pose. This is done by alternately moving the arms towards 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 excluded form further goal extension steps.

Algorithm 4: $DualArmExtendToGoal(RRT, gc)$

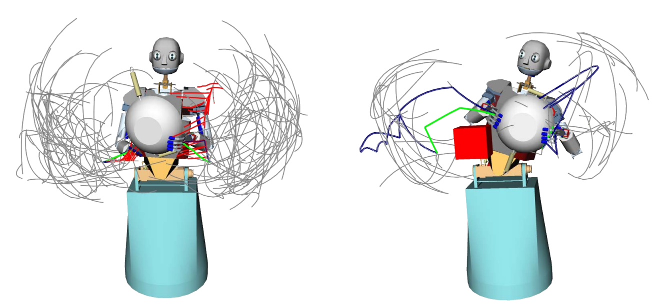
```
grasp ← GetRandomGrasp(gc);
n ← GetNodeMinDistanceTCPs(RRT);
while (!Timeout()) do
    n ← MoveLeftArm(n, grasp);
    if (!n) then
        return NULL;
    n ← MoveRightArm(n, grasp);
    if (!n) then
        return NULL;
    if (HandOffPoseReached(n, grasp)) then
        return BuildSolutionPath(n);
end
```

Algorithm 5: MoveLeftArm($n, grasp$)

$p_{left} \leftarrow TCPLeft(n)$;
$p'_{left} \leftarrow TargetPoseLeft(n, grasp)$;
$\Delta_p \leftarrow p'_{left} - p_{left}$;
$\Delta_q \leftarrow J^+(q_{left}) * LimitCartesianStepSize(\Delta_p)$;
$q_{left} \leftarrow q_{left} + \Delta_q$;
if $(Collision(q_{left}) \,||\, !InJointLimits(q_{left}))$ **then**
    return $NULL$;
return $BuildNewConfigurationLeft(n, q_{left})$;

B. Dual-Arm IK-RRT

With the IK-solver methods presented in section II it is possible to generate feasible configurations for dual-arm re-grasping tasks. The search for these configurations can be included in a RRT-based planner as described in section IV-C. 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 the 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.



(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. 5. 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.

VI. RESULTS

A. Single Arm Reaching

In table 3 the performance of the J^+-RRT and the IK-RRT planners is compared. The average values of 100 test runs are shown and reveal that the usability of the J^+-RRT is limited in cluttered scenes because of the long run times. The IK-RRT algorithm is faster and due to the fast IK solver the planning times are suitable for the use in real world scenarios.

TABLE 3

PERFORMANCE OF THE SINGLE ARM APPROACHES.

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

B. Dual-Arm Re-Grasping

The result of the dual-arm re-grasp planners are shown in table 4. Again, the IK-RRT planner is much faster than the J^+ approach.

TABLE 4

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

C. Dual-Arm Motion Planning in a Kitchen Scenario

To evaluate the performance and capabilities of the developed algorithms in real world scenarios, a manipulation task in a kitchen environment is studied. A wok should be grasped with the right hand of the robot, a re-grasping motion has to be planned and finally the object has to be placed in a cabinet. The planning framework should be able to generate collision-free joint trajectories in reasonable time. For this example, the task of solving the IK problem and the collision-free motion planning are considered separately. This leads to a planner which looses the ability of being probabilistically complete, since the planning is limited to one set of IK solutions and if this IK solution is not reachable the planning will fail. The experiments showed, that the situation, where an IK solution is not reachable by a collision free motion, was never observed and thus this theoretical disadvantage does not affect the applicability of this manipulation planning approach. Theoretically it is possible to build a planner which is probabilistically complete. This can be done for this kind of manipulation planning problem, by searching IK solutions in parallel and for every solution an instance of the planning algorithm is started. If time goes to infinity, all possible IK solutions will be discovered and if a valid solution exists the planner will find it.

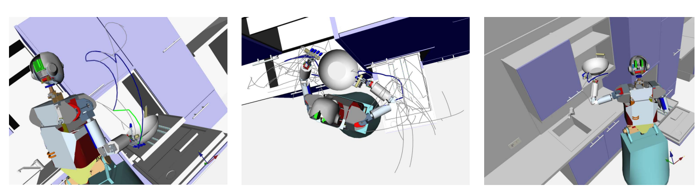


Fig. 6. The results of the three planning tasks. In the left image the wok is grasped with the right hand, then the re-grasping procedure is executed and finally the object is placed in the cabinet.

TABLE 5

PERFORMANCE OF THE KITCHEN EXPERIMENT.

	IK Solving	Motion Planning
Grasp	19.6 ms	345 ms
Re-Grasping	760.7 ms	4 702 ms
Place	22.6 ms	1 263 ms
Complete	802.9 ms	6 310 ms

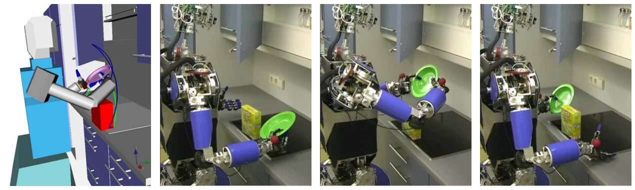


Fig. 8. The humanoid robot ARMAR-III is re-grasping a plate in the kitchen.

D. Hand-off with 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 Fig. 7. The performance of the two arm hand-off planning algorithms is similar to the one robot case of section VI-B. 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.

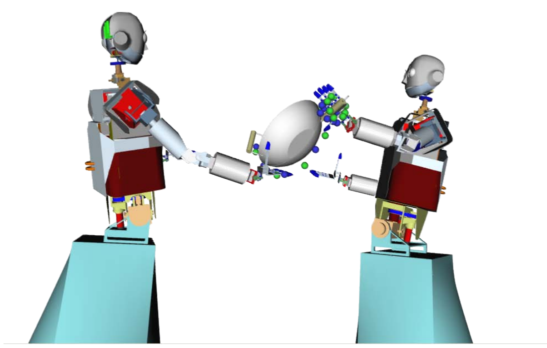


Fig. 7. A hand-off configuration for two robots.

E. 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 (see section V) is used to find a hand-off configuration and to plan a collision free hand-off motion for both arms. The resulting trajectory moves 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. Real world experiments show how the dual-arm re-grasping planners enable the humanoid robot ARMAR-III to hand-off objects from one hand to the other.

VII. CONCLUSION

We presented and compared two main strategies for motion planning of reaching and re-grasp motions including single and dual arm tasks. The search for a suitable and collision free configuration for grasping or re-grasping an object is included in the planning algorithms and thus the planners cover the search for suitable target configurations implicitly. The J^+ apporach, which doesn't need an IK solver implementation, is compared with the IK-RRT approach, which benefits from the possibility of planning bi-directional. Several planning setups are investigated and the performance of the different algorithms is evaluated in simulations and real world experiments.

The presented planners can be used to efficiently plan reaching and re-grasping tasks without defining explicit target configurations. This leads to planning algorithms which can be applied in humanoid robots and which can be addressed and parameterized easily, e.g. for a higher level task planning.

The algorithms are also applied for a multi robot planning task and the execution on the real robot ARMAR-III shows the practical usability of the presented work.

VIII. ACKNOWLEDGEMENTS

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 projects GRASP (FP7-215821). We also thank the InterACT program [15] for making this joint research project possible.

REFERENCES

[1] J. J. Craig, *Introduction to Robotics*. Reading, MA: Addison-Wesley, 1989.

[2] T. Asfour, K. Regenstein, P. Azad, J. Schröder, A. Bierbaum, N. Vahrenkamp, and R. Dillmann, "Armar-III: An integrated humanoid platform for sensory-motor control." in *IEEE-RAS International Conference on Humanoid Robots (Humanoids 2006)*, December 2006, pp. 169–175.

[3] L. Sciavicco and B. Siciliano, *Modeling and Control of Robot Manipulators*, 2nd ed. Springer, 2000, pp. 96–100.

[4] M. V. Weghe, D. Ferguson, and S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics," in *IEEE-RAS International Conference on Humanoid Robots*, November 2007.

[5] T. Asfour and R. Dillmann, "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)*, 2003.

[6] N. I. Badler, C. B. Phillips, and B. L. Webber, *Simulating Humans: Computer Graphics Animation and Control*. New York, Oxford: Oxford University Press, 1993.

[7] L. Guilamo, J. Kuffner, K. Nishiwaki, and S. Kagami, "Efficient prioritized inverse kinematic solutions for redundant manipulators," Aug. 2005, pp. 3921–3926.

[8] R. Diankov, N. Ratliff, D. Ferguson, S. Srinivasa, and J. Kuffner, "Bispace planning: Concurrent multi-space exploration," in *Robotics: Science and Systems*, June 2008.

[9] D. Kee and W. Karwowski, "Analytically derived three-dimensional reach volumes based on multijoint movements," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 44, pp. 530–544(15), 2002.

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

[11] D. Berenson and S. Srinivasa, “Grasp synthesis in cluttered environments for dexterous hands,” in *IEEE-RAS International Conference on Humanoid Robots (Humanoids08)*, 2008.

[12] D. Berenson, S. Srinivasa, D. Ferguson, A. Collet, and J. Kuffner, “Manipulation planning with workspace goal regions,” in *IEEE Int’l Conf. on Robotics and Automation (ICRA’2009),Kobe,Japan*, 2009.

[13] R. Geraerts and M. H. Overmars, “On improving the clearance for robots in high-dimensional configuration spaces,” in *IEEE/RSJ Intl Conf. on Intelligent Robots and Systems*, 2005, pp. 4074–4079.

[14] J. Kuffner and S. LaValle, “RRT-Connect: An efficient approach to single-query path planning,” in *IEEE Int’l Conf. on Robotics and Automation (ICRA’2000), San Francisco, CA*, 2000, pp. 995–1001.

[15] InterAct, “http://isl.ira.uka.de/.”