Adaptive Motion Planning for Humanoid Robots

Nikolaus Vahrenkamp*, Christian Scheurer*, Tamim Asfour*, James Kuffner† and Rüdiger Dillmann*

*Institute of Computer Science and Engineering
University of Karlsruhe
Haid-und-Neu Str. 7
76131 Karlsruhe, Germany
{vahrenkamp,scheurer,asfour,dillmann}@ira.uka.de

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

***Abstract*— Motion planning for robots with many degrees of freedom (DoF) is a generally unsolved problem in the robotics context. In this work an approach for trajectory planning is presented, which takes account of the different kinematic parts of a humanoid robot. Since not all joints of the robot are important for different planning phases, the RRT-based planner is able to adapt the number of DoF on the fly to improve the performance and the quality of the results. The runtime of the approach is evaluated in comparison to a standard RRT planner. Futhermore several extensions to the algorithm are investigated.**

I. INTRODUCTION

Motion planning for highly redundant robot systems with many degrees of freedom is a challenging field of research. For a given task, the algorithms have to find trajectories for all involved joints in a fast and robust manner. The general motion problem is PSPACE hard [1], which means the planning time can dramatically inrease for systems with many degrees of freedom. To solve this problem, approximated algorithms have been developed, which promise a faster search for solutions, but with the restriction that some solutions are not noticed [2].

A humanoid robot has several subsystems, like arms, hands and a head, which should be involved into the planning process. In [3], an approach is presented where the number of active joints changes dynamically in order to adapt the volume of the reachable workspace. In [4], a multi-level planning scheme is presented where the planning complexity is iteratively increased by adding nonholonomic constraints at each planning level. The planner in [5] 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 [6], 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 approach that is presented here, picks up the idea of adaptively controlling the number of DoF used for planning and thus improving the planning performance. The planner uses different kinematic subsystems of the robot which are selected depending on the planning phase. E.g. a typical planning task like grasping an object in a cupboard can be divided into subgoals. The robot has to move in front of the object, then a reaching process of the arm should bring the tool center point (TCP) next to the target and finally the hand should grasp the object. The different phases do not need to be planned with full detail of the robot, since some joints are not important for the subtask. The adaptive planner, described in this paper, selects the kinematic subsystems depending on the distance to the target. The planner is based on efficient single-query RRT algorithms ([7], [8], [9]) which are used to build up a tree of collision free configurations. The proposed algorithms are evaluated in comparison to a reference implementation of the RRT algorithm. As testcase the simulation environment of the humanoid robot ARMAR-III (Fig. 1) is used for planning a grasping task.

Fig. 1. The humanoid robot ARMAR-III.

II. RRT-BASED PLANNING

Rapidly-Exploring Random Trees (RRTs) belong to the category of sampling-based, randomized planning algorithms ([8], [7]). They are widely used for single-query path planning because of their simplicity and effciency as well as the possibility of involving differential constraints and many degrees of freedom. Several variations from the basic RRT algorithms to bi-directional and other problem-adapted planners have been arisen in the last few years due to the multifarious application range of Rapidly-Exploring Random Trees [10].

The key advantages of the basic Rapidly-Exploring Random Tree construction, described in [8], are that the expansion of a RRT is biased toward unexplored state space and only few heuristics and parameters are needed. Furthermore RRT-based algorithms are resolution complete, i.e. with increasing iterations the apporach gets arbitrarily close to any point in the free C-space.

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 improvements by biasing the search toward a goal are reasonable. Two simple RRT-based planner named *RRT-GoalBias* and its enhancement *RRT-GoalZoom* are implementing this technique [2]. *RRT-GoalBias* selects the goal configuration as random state in a small number of iterations, so that the methods Extend or Connect are trying to generate a straight-line path toward the goal. Introducing too much bias on the other hand can also lead to poor efficiency because the planner gets trapped in local minima during the search process. For this reason, *RRT-GoalZoom* as a further improvement of *RRT-GoalBias* was introduced. Here, a region around the goal instead of one individual configuration is used to bias the search. This goal region is in addition controlled by the closest RRT vertex in the tree, so that its extent is changed dynamically. In some cases, a configuration picked from this goal region rather than from the whole state space is used to try a connection to it [10].

Since the resulting trajectories are generated by connecting random configurations in C-space, the movement of the robot is not optimal. To get a smooth and short solution path a postprocessing step is necessary which converts the result of the RRT-based planning to a homotopic path which comes close to an optimal solution. A simple and powerful approach, which is used to smooth the results of Fig. 5 and 6, randomly searches collision free shortcuts in the solution path ([4], [6]). Further improvements can be achieved by iteratively increasing the obstacle distance of all path points until a minimum distance is reached [11].

III. WEIGHTED SAMPLING

The dimensions of the configuration space 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.

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} millimeters. All other dimensions of the C-space have to be investigated explicitly to derivate the upper bound of the robot's workspace movement. Table I gives an overview of the maximum displacement of a point on the robot's surface when changing one unit in C.

TABLE I
WORST CASE WORKSPACE MOVEMENT FOR ARMAR-III.

Degree of freedom	*mm*	*Degree of freedom*	*mm*
Platform Translation x	1	Arm Elbow	390
Platform Translation y	1	Wrist 1	150
Platform Rotation	1176	Wrist 2	150
Torso Pitch	1176	Wrist 3	150
Torso Roll	1176	Hand Thumb 1	70
Torso Yaw	1176	Hand Thumb 2	70
Head Pitch	300	Hand Index 1	70
Head Roll	300	Hand Index 2	70
Head Yaw	300	Hand Middle 1	70
Arm Shoulder 1	700	Hand Middle 2	70
Arm Shoulder 2	700	Hand Ring	70
Arm Shoulder 3	390	Hand Pinkie	70

The effects of moving one unit in the different dimensions can be seen in Fig. 2.

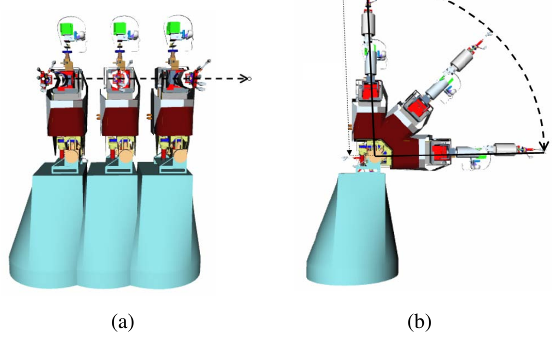


Fig. 2. (a) The humanoid robot ARMAR-III moving in a translational dimension. (b) The effect in workspace when changing the C-space value for the dimension associated with the torso pitch joint.

The different workspace effects are considered by using a weighting vector w whose elements are given by the values of the workspace movements from table I. In Eq. 1 the maximum workspace movement $d_{WS}(\vec{c})$ of a C-space path $\vec{c} = (c_0, ..., c_{n-1})$ is calculated.

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

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

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

The maximal workspace stepsize ε_{ws} can be specified in millimeters, which allows to control the granuality of the planning algorithms in an easy way. The stepsize ε_{ws}

is used to generate $n = \lceil \frac{d_{ws}}{\varepsilon_{ws}} \rceil - 1$ intermediate sampling configurations $\vec{c}_k$ on the path between two configurations $\vec{c}_1$ and $\vec{c}_2$.

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

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

IV. 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 [12], is not able to find solutions in reasonable time. The 43-dimensional C-space is not suitable for searching free space paths, even when using large step-sizes for approximation.

To be able to find solutions in a high dimensional C-space, we introduce a planning scheme which adaptively changes the number of joints used for planning. 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. 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.

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 II

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 concidering 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 choosen 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 postprocessing step, e.g. the head can be adjusted to see the target.

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 V.

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 envirmonmet 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 choosen 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 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. 3 for the subsystems *Platform, Torso, Right Arm, Right Wrist* and *Right Hand*.

Table III 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 number of joints could not increased any more and the planner should go on until a global solution is found.

TABLE III

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

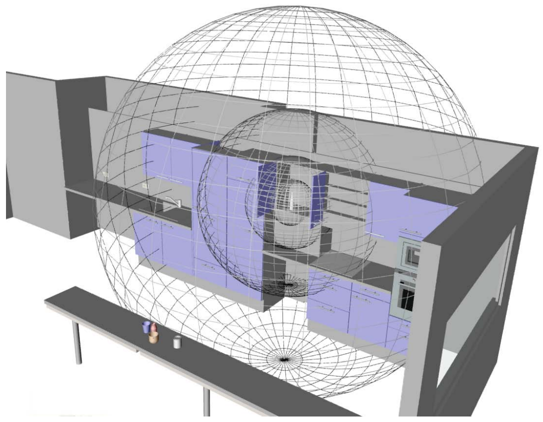


Fig. 3. The extent of the planning levels around a target object for five subsystems of ARMAR-III.

V. EXTENSIONS TO IMPROVE THE PLANNING

Randomly Extending Good Ranked Configurations

As described in [13] 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.

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 ([14], [15]). To support the RRT-based planner, a bi-directional planning appoach can be used, which builds up trees from the start and goal configuration and iteratively tries to connect them [8], [16].

The adaptive planner has to be adopted slightly to support the bi-directional search. The foreward 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 adoption 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.

Focussing 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 focussing 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 emphazise a specific area in workspace an adaption of the *GoalZoom* algorithm is used. Sometimes, instead of an arbitary 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 choosen 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 towards the goal, has to hold the constraint, that the maximum workspace distance d_{WS} is smaller than $d_{PlanningArea}$ (Eq. 4).

$$d_{WS}(\vec{c}_{goal}, \vec{c}_{zoom}) < d_{PlanningArea} \tag{4}$$

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

VI. EXPERIMENTS

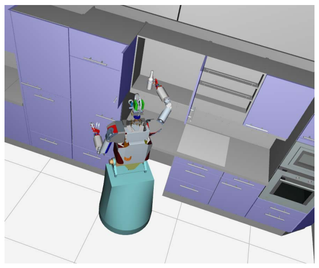
Setup

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. Thus the total number of triangles can be reduced from 40.000 to 1.000. The robot is operating in a kitchen environment, which is also modeled with full and reduced resolution for visualization and collision checking [9].

The robot should start in front of the shelf and find a trajectory for grasping an object in the cupboard. 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 out 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. The configuration for grasping the target object can be precalculated and stored with the object data, e.g. the GraspIt! simulator can be used [17].

The collision checking routines are a critical point for RRT-based planning, since the planner spends most of the time testing robot configurations for collisions with the environment. Fast and robust collision checking routines are given by the PQP implementation which uses swept sphere volumes to test the collision status for 3D models [18].


(a) distance view

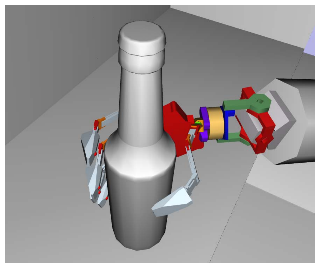

(b) narrow view

Fig. 4. Target position of the planning task.

The maximum workspace stepsize ε_{ws} was set to $30mm$ for collision checking on path segments. Since ε_{ws} is a worst-case approximation, the real stepsizes 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 $90mm$ in all tested planners.

All test runs have been carried out on an Intel Core 2 Duo Linux System with 2.16 GHZ and 2GB RAM.

RRT-Connect (A)

A first evaluation of the planning problem described above, has been done by using a RRT-Connect one-way planner. 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 (100000) 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.

Adaptive Planning (B)

The adaptive planning 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.

Enhanced Adaptive Planning (C)

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

TABLE IV
UNIDIRECTIONAL PLANNING

	Planning succeeded	Avg. Planning time (success)	Avg. number of RRT nodes	Avg. number of collision checks
A	37.5 %	98.6s	30 907	252 605
B	43.5 %	31.3s	10 089	83 017
C	57.5 %	14.2s	5 123	40 522

Bi-Planning: RRT-Connect (D)

The RRT-Connect planner 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. The planning succeeded in every test run and the average runtime was measured with three seconds (table V row D).

A planned RRT with original (blue) and optimized (green) solution paths are depicted in Fig. 5.

Bi-Planning: Adaptive Planning (E)

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 was implemented and tested as described in section V. The adaptive reduction of the subsystems combined with a bi-planner results in an average planning time of 477 milliseconds (table V row E).

Bi-Planning: Adaptive Planning with Enhancements (F)

The *GolZoom*-enhancement, described in section V, which noticeable increased the planning performance for unidirectional planners, just decreases the planning time slightly for the adaptive bi-directional planner. As shown in table V the average planning time decreases from $477ms$ to $423ms$. Fig. 6 shows a typical RRT with original and reduced solution paths for this configuration of the planning algorithm.

TABLE V
BI-DIRECTIONAL PLANNING

	Planning succeeded	Avg. Planning time	Avg. number of RRT nodes	Avg. number of collision checks
D	100.0 %	3037ms	784	4 802
E	100.0 %	477ms	477	1 443
F	100.0 %	423ms	388	1 181

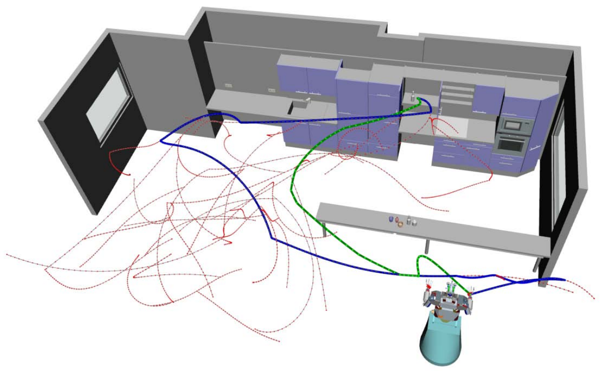


Fig. 5. RRT-Connect Bi-Planning: The RRT with original (blue) and optimized (green) TCP paths.

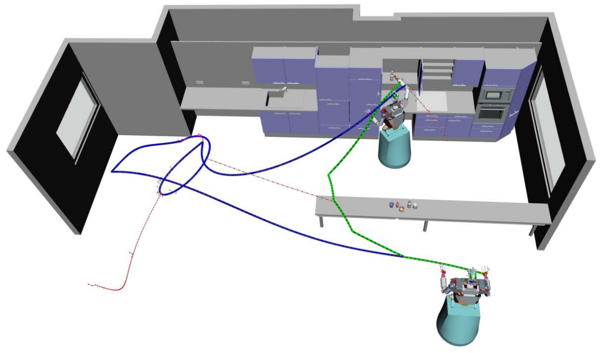


Fig. 6. Adaptive Bi-Planning: The RRT with original (blue) and optimized (green) TCP paths.

VII. CONCLUSIONS AND FUTURE WORK

Conclusions

An adaptive planning approach based on the RRT algorithm was presented.

The algorithms have been evaluated with a common planning problem for a humanoid robot. As testcase a grasping task in a kitchen environment was choosen 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 unsucessful 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.

Future Work

A more general planner, based on the methods presented in this paper, is desireable. It would be interesting to investigate approaches which automatically choose the correct subsystems and adaptively change the planning levels. A supervising module can monitor the succes of the algorithm and by adjusting the parameters on the fly, the planning algorithm can be adopted to the needs of the planning phases. To increase the performance of the suggested algorithms, the nearest neighbor search can be accelerated by using efficient algorithms like kd-trees or the ANN approach presented in [19].

VIII. 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).

REFERENCES

[1] J. H. Reif, "Complexity of the mover's problem and generalizations (extended abstract)." in *FOCS*, 1979, pp. 421–427.

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

[3] N. E. Sian, K. Yokoi, S. Kajita, and K. Tanie, "A framework for remote execution of whole body motions for humanoid robots." in *Humanoid Robots, 2004 4th IEEE/RAS International Conference on*, vol. 2, Nov. 2004, pp. 608–626.

[4] S. Sekhavat, P. Svestka, J. Laumond, and M. Overmars, "Multi-level path planning for nonholonomic robots using semi-holonomic subsystems." 1996.

[5] E. Yoshida, "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*, 2005, pp. 3378–3383.

[6] P. C. Chen and Y. K. Hwang, "SANDROS: A dynamic search graph algorithm for motion planning." *IEEE Transactions on Robotics & Automation*, vol. 14, no. 3, pp. 390–403, 1998.

[7] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning." *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001.

[8] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning." in *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'2000), San Francisco, CA*, April 2000.

[9] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Efficient motion planning for humanoid robots using lazy collision checking and enlarged robot models," in *Intelligent Robots and Systems, IROS*, October 2007.

[10] S. LaValle and J. Kuffner, "Rapidly-exploring random trees: Progress and prospects." 2000, in Workshop on the Algorithmic Foundations of Robotics.

[11] R. Geraerts and M. H. Overmars, "Clearance based path optimization for motion planning." in *International Conference on Robotics and Automation(ICRA'04)*, 2004, pp. 2386–2392.

[12] 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.

[13] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour, "An integrated approach to inverse kinematics and path planning for redundant manipulators." in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, May 2006, pp. 1874–1879.

[14] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces." *International Journal of Computational Geometry and Applications*, vol. 9, no. 4/5, pp. 495–, 1999.

[15] V. Boor, M. Overmars, and A. Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners." in *IEEE International Conference on Robotics and Automation*, 1999, pp. 1018–1023.

[16] G. Sanchez-Ante, "Single-query bi-directional motion planning with lazy collision checking." Ph.D. dissertation, ITESM, Campus Cuernavaca, Mexico, 2001.

[17] A. T. Miller, "Graspit!: a versatile simulator for robotic grasping." Ph.D. dissertation, 2001.

[18] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes." Department of Computer Science, University of North Carolina, Tech. Rep., 2000.

[19] A. Yershova and S. M. LaValle, "Improving motion planning algorithms by efficient nearest-neighbor searching." *Transactions on Robotics*, vol. 23, pp. 151–157, February 2007.