Efficient Motion Planning for Humanoid Robots using Lazy Collision Checking and Enlarged Robot Models

Nikolaus Vahrenkamp, Tamim Asfour and Rüdiger Dillmann
Institute of Computer Science and Engineering
University of Karlsruhe,
Haid-und-Neu-Strasse 7, 76131 Karlsruhe, Germany
Email: {vahrenkamp,asfour,dillmann}@ira.uka.de

***Abstract*— Motion planning for humanoid robotic systems with many degrees of freedom is an important and still generally unsolved problem. To give the robot the ability of acting and navigating in complex environments, the motion planner has to find collision-free paths in a robust manner. The runtime of a planning algorithm is critical, since complex tasks require several planning steps where the collision detection and avoidance should be accomplished in reasonable time. In this paper we present an extension of standard sampling-based techniques using Rapidly Exploring Random Trees (RRT). We extend the free-bubble path validation algorithm from Quinlan, which can be used to guarantee the collision-free status of a C-space path between two samples. By using enlarged robot models it is possible to avoid costly distance calculations and therefore to speed up the planning process. We also present a combined approach based on lazy collision checking that brings together the advantages of fast sampling-based and exact path-validated algorithms. The proposed algorithms have been evaluated by experiments on a humanoid robot in a kitchen environment and by a comparison to a validation based on Quinlan's free bubbles approach.**

I. INTRODUCTION

Motion planning is an important and generally unsolved topic for robotic systems with many degrees of freedom (DoF). A motion planning algorithm should find a trajectory of a kinematic chain that is collision-free and moves the tool center point (TCP) from a start to a goal position. The algorithms have to fulfill several requirements, such as low runtime, short path length or high obstacle distance. Since the motion planning problem is known to be PSPACE-hard [1], complete algorithms ([2], [3]) are too time consuming and therefore less suitable for real-time tasks of highly redundant robots which operate in a cluttered environment. Over the last years probabilistic, sampling-based approaches have been developed which are able to find solutions efficiently [7]. 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 no solution exists. The major drawback of these implementations is the fact that valid solutions could not be recognized when stopping the calculations.

If we search a trajectory for a robot with n DoFs we could decompose the n-dimensional configuration space (C-space) into two sub-spaces: C_{obst} representing all configurations that result in a collision in the workspace and C_{free} the complement of C_{obst}.

$$C = C_{obst} \cup C_{free} \tag{1}$$

Motion planning algorithms search a path from a given start configuration c_{init} to a goal configuration c_{goal} that lies completely in C_{free} and thus is collision-free in the workspace. The start and goal configurations are maintained by the inverse kinematics component of the robot system.

When planning with complex obstacle structures, explicit modeling of obstacles as C-space regions can be difficult and time consuming. One possibility to accomplish the modeling is the approximation of the C_{obst} regions by generalized quadtrees [4] or an adaptive volumetric grid [5]. Since usually only small regions of the C-space are utilized for planning, no explicit models for the complete C-space are required. In this paper we will concentrate on techniques that avoid the explicit modeling of C-space obstacles, which increases the processing time, especially for systems with many degrees of freedom. For that reason our work is based on the Rapidly Exploring Random Trees (RRT) that are described in more detail at section IV.

A more detailed overview about planning algorithms can be found in the motion planning book from LaValle et al. that is also available online [7].

II. PLANNING FRAMEWORK

A motion planner which has to be used in a real-time applications must fulfill several requirements. The planner should be fast and the planned trajectories should be adopted to a changing environment. Examples of motion planning methods that address dynamic environments are given in [19] and [20]. With these approaches it is possible to build a planner that is able to react on dynamic obstacles, but for a system with 43 DoFs, like our humanoid robot ARMAR-III [6], the described planning algorithms are not practical. A complex robot system needs a multi-resolutional planning system that is able to combine different planning algorithms with varying levels of robot details. A path planning algorithm that should find a way for the mobile platform, can use a lower resolution for the hand models, e.g. by turning off the kinematic chain and regarding the complete hand as one joint with a bounding box. On the other hand a

dexterous manipulation task should be planned with a higher resolution hand model. The planning system for ARMAR-

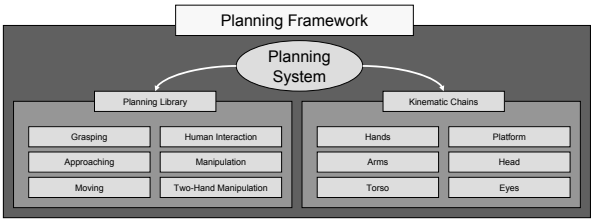


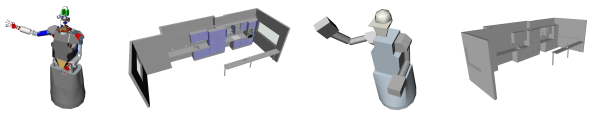
Fig. 1. The Planning Framework.

III can choose between different planning algorithms that are adopted to the needs of the planning problem. E.g. path planning for a mobile platform does only need an approximated, but very fast planner that is able to find a way for the robot to a desired position. In this case not all joints of the robot are needed for planning, thus the efficient planning allows us to react on dynamic objects like humans passing the way. For manipulation tasks it is important to create collision-free trajectories for the arms of the robot. In comparison to the mobile platform we have less clearance to operate, and thus we need a planner that operates with respect to a higher degree of detail. We present an approach for planning the motion for a humanoid arm with seven DoFs that is used within the motion planning framework. We show how it is possible to decrease the average planning time without loosing the collision-free guarantee that is needed for a robot interacting in a real world environment.

III. 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. There are a lot of libraries that can be used for collision detection ([15], [16], [17]). We are using the PQP library [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. 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 20.000 to 450 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 $0.20ms$ and an average distance query time of $0.65ms$. Compared to $0.32ms$ and $3.58ms$ for a setup with full models of robot and environment we could achieve a speedup of 37.5% and 81.8%. Like all tests presented here, these performance evaluations have been carried out on a Linux-Pentium4 system with 3.2 GHz.



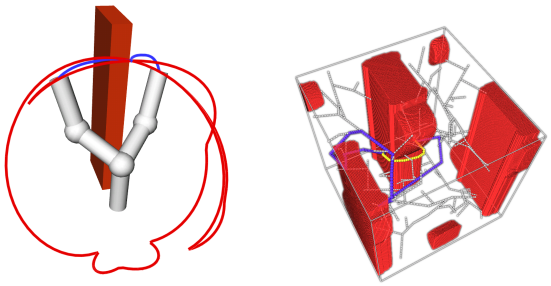
(a) (b) (c) (d)

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

IV. SAMPLING-BASED MOTION PLANNING

Sampled C-Space

The idea of sampling-based motion planning is to search a collision-free C-space path without explicitly modeling the C_{obst} regions. A point in the C-space represents a workspace configuration of the robot. Fig. 3(a) shows a sample robot manipulator with three DoFs in front of an obstacle. The corresponding C-space is shown in Fig. 3(b), where the red dots represent configurations that result in a workspace collision.



(a) (b)

Fig. 3. (a) A three joint robot system in front of an obstacle. (b) The sampled C-space and a RRT with original and optimized solution paths.

For a given start and goal configuration, the planner has to find a path through the C-space while avoiding sample points which belong to C_{obst}. Because the modeling of C_{obst} takes a long time we just sample the needed path points on the fly while searching a way through the C-space (see [7]).

RRT

There are several planning algorithms implementing this basic sampling-based concept ([7], [8]). Good results can be achieved with rapidly exploring random trees (RRT), introduced by Steve LaValle and James Kuffner in [9] and [10]. RRT-based algorithms reduce the global planning problem to local C-space path search problems which can be solved easier.

The RRT-approach builds up a C-space tree that uniformly covers the free space. The RRT-CONNECT algorithm proposed in [10] samples the C-space randomly and tries to extend the existing tree by connecting to the new samples with straight lines. It has been showed that RRTs generate a uniform covering of the C-space and that it is possible to construct RRT-based, resolution complete planners [13]. An example of a three dimensional RRT is given in Fig. 3(b).

Guaranteeing Collision-Free Paths

In all sampling-based approaches, the sampling resolution of the C-space can be specified with a resolution parameter. The choice of the resolution parameter affects the quality of the result as well as the runtime of the algorithm. If the resolution is too high, the runtime will be unnecessary long. On the other hand, with a low resolution, the planner will run fast but might not consider some obstacles. This leads to another problem of sampling-based approaches: The collision status of the connection between two neighboring sampled configurations on the path is unknown. Regardless which sampling resolution is chosen, there is no guarantee that the path between two neighboring samples is collision-free.

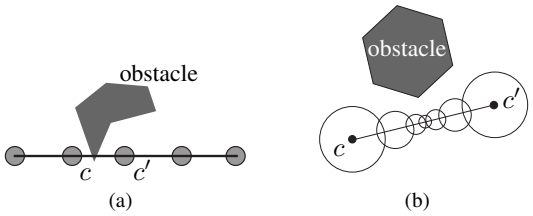


Fig. 4. (a) C_{free} samples and a collision path. (b) A valid C-space path covered by free bubbles.

Figure 4(a) shows such a situation. The C-space path between two sampled configurations c and c' is blocked by an obstacle. A sampling-based planner will not detect a collision since c and c' are in C_{free}.

To overcome this problem Quinlan has introduced in [19] an approach, which can be used to guarantee a collision-free path between two C-space samples. Quinlan calculates bubbles of free space around a configuration and therefore can guarantee the collision-free status of a path segment by overlapping these bubbles along the segment. For a given C-space path $\overline{cc'}$, Quinlan derives an upper bound for the distance any point on the geometry of the manipulator can travel for a given change in the configuration. This upper bound can be regarded as a metric $\delta(c, c') \mapsto \Re$ in the C-space [24]. Let $c = (c_1, ..., c_n)$, $c' = (c'_1, ..., c'_n)$ and r_i the maximum distance of a point on the surface of segment i to any point on the entire geometry, then the metric can be defined as follows:

$$\delta(c, c') = \sum_{i=1}^{n} r_i |(c'_i - c_i)| \tag{2}$$

In [24], the free bubble of a given configuration c is defined using the metric δ and the obstacle distance d_{obst}, where d_{obst} defines the minimum distance between the robot in configuration c and the obstacles.

$$\mathcal{B}(c) = \{c' \in C : \delta(c, c') \leq d_{obst}\} \tag{3}$$

With this approach a lower bound for the guaranteed collision-free path in the C-space is given. In order to validate if a complete path is collision-free, we have to choose samples along the path so that their free bubbles overlap. If all of the chosen samples are collision-free one can conclude that the complete path is also collision-free. An effective realization can be achieved by a divide and conquer algorithm (see [21]) that recursively splits the path in two sub-paths until the free bubbles of the two endpoints overlap (Fig. 4(b)). We break the recursion and report a collision if the obstacle distance falls below a minimum to avoid long (or even endless) running times if a path is not collision-free. In all of our tests this distance was set to $2mm$.

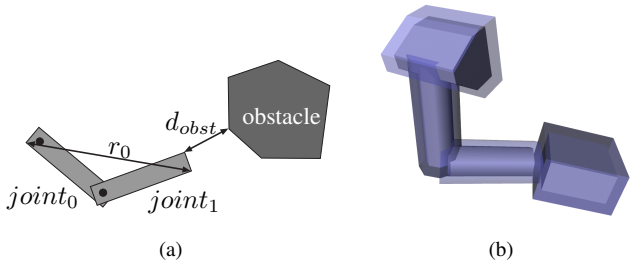


Fig. 5. (a) 2D robot arm showing the maximum distance r_0 and the minimum obstacle distance d_{obst}. (b) Collision and Enlarged Model of the right arm. The free space distance was set to $20mm$ for visualization purposes.

Because we are calculating an upper bound of the joint movement and thus a lower bound for the free bubble radius, the radius tends to get very small if a configuration results in a low obstacle distance. This behavior results in numerous distance and collision calculations and slows down the whole planning algorithm. Furthermore, the distance calculations are very expensive compared to the simple collision status determination. A comparison between a purely sampling-based and a free bubble validated RRT planner shows that the validation of each path is very expensive even when taking simplified collision models (see table I).

As an evaluation setup we chose a model of the right arm of our robot operating in a kitchen environment (Fig. 7(b)). The robot arm has seven DoFs. The sampling size was set to 0.04 radians. In 6% of the test runs the sampling-based approach generates a path that is not valid because the robot runs into collisions between two path points. An approach to improve sampling-based planning is given in section VI. Furthermore, the results point out that the distance calculations of the free bubble approach are time consuming compared to the high number of collision checks of the purely sampling-based algorithm.

V. ENLARGED ROBOT MODELS

The long run time of the free bubble approach arises from the high number of workspace distance calculations. With the enlarged model approach we propose a method to guarantee a collision-free status of a path without any distance computations. This results in a faster path validation and thus in a speedup of the planning algorithm. There are approaches like [11] and [12] using enlarged obstacle models to ensure safety distances. The approach presented here differs from these works in two ways. We apply the enlargement on the robot, since the robot joint models are known to be convex

which make the enlargement easier. The second and main difference is that our approach uses the enlarged models to avoid time-expensive distance calculations for guaranteeing collision-free path segments.

Construction of the Enlarged Robot Models

For convex collision models, the enlarged models can be constructed by slightly scaling up the original robot 3D models so that the minimum distance between the surfaces of the original and the enlarged model reaches a lower bounding. We call this lower bounding the free space distance. Figure 5(b) shows the original collision model of the right arm and the transparent enlarged models ($20mm$ free space distance). If non-convex objects are used we have to decompose them to convex models in a preprocessing step. The enlarged robot model is created by re-uniting the scaled convex objects to one joint model. There are several decomposition algorithms available ([14]), an implementation is provided with the SWIFT++ library [18]. But in most cases (like with the robot models of ARMAR-III) the simplified collision models of the joints will be convex and there is no need of any decomposition steps.

Using the Enlarged Robot Models

In collision free situations a lower bound of the obstacle distance for the original model can be set directly, i.e. without any distance calculations. To guarantee the collision-free status of a path segment, Quinlan's path validating algorithm can be used. Equation (3) can now be expressed with the free space distance as fixed parameter:

$$\mathcal{B}(c) = \{c' \in C : \delta(c, c') \leq d_{freespace}\} \tag{4}$$

Using the lower bound for the distance results in smaller free bubble radii and thus in more sampling calculations along a path segment. This overhead is compensated by avoiding the slow distance calculations.

TABLE I

COMPARISON OF DIFFERENT PLANNING ALGORITHMS

Planning Algorithm	Avg Time	Result not Valid	Collision Checks	Dist. Calc.
Sampled	3,00s	6%	17.940	0
Free Bubbles	8,28s	0%	6.565	4.753
Enlarged $1mm$	31,36 s	0%	54.584	0
Enlarged $5mm$	7,14 s	0%	13.562	0
Enlarged $10mm$	4,59 s	0%	9.636	0
Enlarged $20mm$	7.21 s	0%	18.822	0
Lazy Collision Check	*2.54 s*	*0%*	*6.090*	*0*

The test setup was identical to the sampling-based and free bubble test runs and thus the results of table I can be compared directly. Large changes in the free space distance affects the runtime of the planner. Equation (4) shows that the size of a free bubble directly depends on the free space distance $d_{freespace}$. Thus the number of samples and therefore the number of collision checks needed for path validation also depends on $d_{freespace}$.

Choosing a low value ($d_{freespace} = 1mm$) results in a large amount of collision checks for path validation. A large free space distance ($20mm$) slows down the planning process because the enlarged collision models reduce the free space which renders the path finding problem in a more difficult way for the planner. All test runs generated a valid solution and the best performance could be achieved by setting the free space distance to $10mm$.

The enlarged model planner is not as fast as the pure sampling-based approach (see table I), but it can guarantee that the solution is collision-free and it is up to 45% faster than the free bubble algorithm.

Limitations

The enlarged robot model is constructed by scaling up the collision models, hence the algorithm could get into trouble in narrow workspace situations. If the free space in which the robot could operate is strongly limited and the free space distance is set to a high value, the collision checker will not find a solution, even if there is a possible way in such difficult environment.

The effect of using different free space distances is shown in table I and II. A low free space distance results in a high number of collision checks for path validation. If the free space distance is too high, the planner will not find a solution because of the enlarged collision models. In our test scenarios a free space distance of $10mm$ (approximately 1% of the arm length) was a good tradeoff between increasing the performance and restricting the workspace.

TABLE II

EFFECT OF CHANGING THE FREE SPACE DISTANCE

Free Space Distance	C_{obst} ratio	C_{obst} overhead
$0mm$	19.71 %	0.00 %
$1mm$	19.81 %	0.50 %
$5mm$	20.84 %	5.73%
$10mm$	22.71 %	15.22 %
$20mm$	25.34 %	28.56 %

The results in table II were generated by uniformly sampling the seven dimensional C-space of the arm. The setup is the same as used for the planning tests. The seven degrees of freedom of the robot arm build up the C-space, every sample is checked for collisions in the workspace. The C_{obst} ratio describes how many percent of the sampled configurations ($\sim$227K) do result in a collision. The different free space distances affect the amount of free C-space. By setting the free space distance to $1mm$, the resulting C_{obst} is enlarged by 0.5%. The effect of using a high value can be seen in the case of $20mm$ free space distance, where the obstacle regions are enlarged by 28.56% and the planner has less space to operate.

VI. LAZY COLLISION CHECKING

Approach

In [22] a lazy collision checking approach was presented, in which the collision checks for C-space samples (milestones) and path-segments are decoupled. We want to pick

up the idea of lazy collision checking to speed up the planning process and introduce a two-step planning scheme. In the first step the standard sampling-based RRT algorithm searches a solution path in the C-space. This path is known to be collision-free at the path points, but the path segments between these points could result in a collision. In the second validation step we use the enlarged model approach to check the collision status of the path segments of the solution path. If a path segment between two consecutive configurations c and c' fails during the collision test, we try to create a local detour by starting a subplanner which searches a way around the C-space obstacle (see Fig. 6). Thus, we do not guarantee the complete RRT to be collision-free on creation, instead we try to give a collision-free guarantee of the sampling-based solution afterward and reduce the costly checks to the path segments.

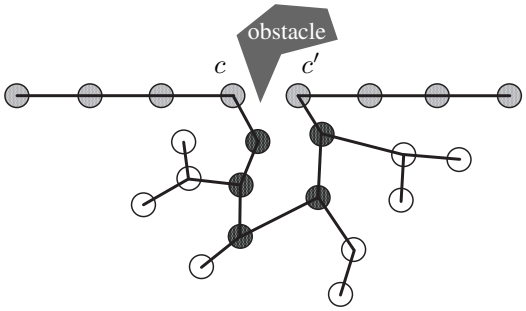


Fig. 6. validated collision path

Searching the detour

To ensure that the path segments of the newly created detour are collision-free, we are using a subplanner based on the RRT-EXTEND algorithm (see [10]) with enlarged models. The use of the RRT-EXTEND planner allows us to build up a locally growing RRT, which can efficiently find a regional solution path. To support the subplanning process we are not using c and c' as start and goal configurations, since they are probably near to an obstacle and thus it will be difficult for the planner to find a collision-free way. Instead we are using randomly chosen positions on the solution path in front of c and behind of c'.

Since our goal is to efficiently find a detour, we are starting the subplanning process with strict parameters, thus the planning cycles are set to a maximum of 100 tries. If the planner does not find a detour within these number of extension steps, the search is stopped and another subplanner is started with different start and goal configurations. As described later, the replanning module does find a solution fast, since the planning problem is regional bounded and the termination of not promising planning calculations combined with the randomly chosen start and goal positions increases the chance of efficiently finding a detour.

Evaluation

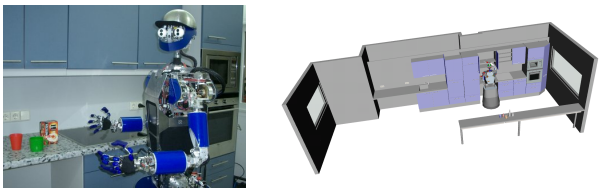
Again we are using the same robot and environment setup. The solutions generated with the purely sampling-based RRT planner are validated with the enlarged model approach. Because of the validation step we can set the sampling resolution to a higher value in order to find the solutions faster; the distance between two sampled configurations was set to 0.2 radians. The validating module has to check every solution if all path segments are guranteed collision-free, this check needed 0.47 seconds in average. In 37.5% of the cases this guarantee can not be given for one or more path segments and the subplanning process is started. Not all of these solutions do really result in a collision in workspace, we just can not guarantee the collision-free status of each path segment. All solutions have been successfully validated by the subplanner module.

Because of the strict parameters of the subplanner mentioned above, 3.1 subplanning calls are needed in average to find a detour for an invalid path segment. The average planning time to find a detour was evaluated with 1.85 seconds. Since sometimes there is more than one detour to find per solution the overall validation time is 2.0 seconds.

In 62.5% of the test cases a guaranteed collision-free solution could be generated in an average time of 1.79 seconds. For 37.5% of the results one or more subplanning steps were needed to validate single path segments. All validation calls succeeded and the overall planning time increased to 3.79 seconds. Combining the validated and replanned cases leads to an average planning time of 2.54 seconds which is faster than searching a solution with the sampling-based planner (see table I).

VII. EXPERIMENT

For evaluation we use a real-world kitchen scenario for the right arm of the humanoid robot ARMAR-III (see Figure 7(a)).



(a) (b)

Fig. 7. (a) The humanoid robot ARMAR-III in a kitchen environment. (b) The planning setup.

Figure 7(b) shows ARMAR-III standing in front of a cabinet. The robot has to operate in a narrow environment, the corresponding C_{free} space is limited because of the short obstacle distances. Nevertheless the free bubble and the enlarged model based planning algorithms can deal with this situation. Figure 8 shows a narrow view of the workspace. The arm has to move from the left cabinet around the open door to the right cabinet. The red trajectory shows the movement of the hand before the solution was smoothed. The blue trajectory describes the movement of the hand after applying the smoothed solution path (To retrieve smooth trajectories, the results are optimized by searching shortcuts, see [23] for details).

A purely sampling-based RRT planner needs 3 seconds on average to find a solution for the given problem, but 6% of the solution paths result in a collision with the environment. The use of Quinlan's free bubble path validation algorithm guarantees that the solutions are collision-free, but the running time increases to over 8 seconds on average. The enlarged robot model approach generates also guaranteed collision-free solution paths, but the running time could be reduced to an average value of 4.6 seconds. By using the lazy collision check approach it is possible to find a guaranteed collision-free solution in 2.5 seconds.

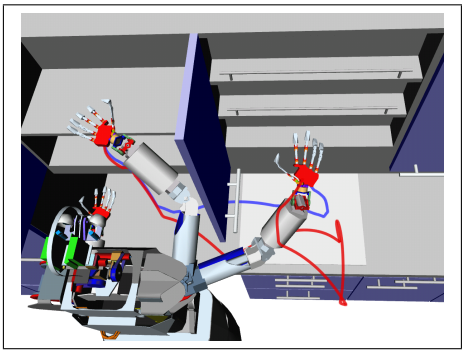


Fig. 8. Start and goal configuration of the robot arm with solution paths. The original and the optimized TCP trajectories are plotted in red and blue.

VIII. CONCLUSION

In this work we presented several approaches to the well-known RRT-based motion planning. We investigated the runtime and the quality of solutions of the different approaches. Furthermore we showed, how different parameters influence the systems behavior.

By using simplified 3D models the planning time was reduced by 30%. These simplified models have no effect on the quality of the results for our robot system because the link models are of convex shape and with the simplification we just loose some visual information that are not needed for the planning process.

Quinlan's path validation approach was modified to avoid the expensive distance calculations without loosing the collision-free guarantee. To achieve this, the enlarged robot models have been introduced as slightly scaled up collision models and it has been shown how to guarantee the collision-free status of a path segment. The evaluations demonstrated that it is possible to generate collision-free motion paths even in narrow environments. Compared to the free bubble approach the guaranteed collision-free motion planning could be accelerated up to 45 %.

If using the fast sampling-based RRT approach it is possible to check and validate the solution with the enlarged model path validation algorithm. This lazy collision checking approach leads to an algorithm that could find solutions faster than the sampling-based approach because of the possibility to decrease the sampling resolution. Since the fast subplanning step generates detours for critical path segments, all resulting solutions are guaranteed collision-free.

IX. ACKNOWLEDGMENTS

This work has been supported by the Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center 588 "Humanoid Robots - Learning and Cooperating Multimodal Robots".

REFERENCES

[1] J. H. Reif, *Complexity of the mover's problem and generalizations.*, IEEE Symposium on Foundations of Computer Science, pages 421-427, 1979.

[2] John F. Canny, *The complexity of robot motion planning*, Cambridge, MA, USA: MIT Press, 1988.

[3] J. T. Schwartz and M. Sharir, *On the piano movers problem: Coordinating the motion of several independent bodies.*, Int. J. Robot. Res., 2(3):97–140, 1983.

[4] B. Paden,A. Mess and M. Fisher, *Path planning using a Jacobian-based freespace generation algorithm*, IEEE International Conference on Robotics and Automation, no.pp.1732-1737 vol.3, 1989.

[5] G. Varadhan,Y.J. Kim,S. Krishnan and D. Manocha, *Topology preserving approximation of free configuration space*, Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, Pages: 3041- 3048, Orlando, May 15-19, 2006

[6] 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*, IEEE-RAS International Conference on Humanoid Robots (Humanoids 2006), Dec. 2006, pp. 169-175.

[7] S. M. LaValle, *Planning Algorithms*, Cambridge University Press (also available at http://msl.cs.uiuc.edu/planning/), 2006.

[8] Yong K. Hwang and N. Ahuja, *Gross Motion Planning - A Survey*, ACM Computing Surveys 24(3): 219-291, September 1992.

[9] S.M. LaValle, *Rapidly-exploring random trees: A new tool for path planning*, TR 98-11, Computer Science Dept., Iowa State University, Oct. 1998.

[10] J. Kuffner and S.M. LaValle, *RRT-Connect: An efficient approach to single-query path planning*, IEEE Int'l Conf. on Robotics and Automation (ICRA'2000), pages 995-1001, San Francisco, CA, April 2000.

[11] P. Irvall, *Obstacle Avoidance System (OAS)*, Masters Degree Project Stockholm, Sweden, Jan 2007.

[12] E. Ferre and J.-P. Laumond, *An iterative diffusion algorithm for part disassembly*, Proc. International Conference on Robotics and Automation (ICRA04), 3149-3154, 2004.

[13] P. Cheng and S. M. LaValle. *Resolution complete rapidly-exploring random trees*, IEEE International Conference on Robotics and Automation, pages 267–272, 2002.

[14] B. Chazelle and L. Palios, *Decomposing the boundary of a nonconvex polyhedron*, Algorithmica, 17:245–265, 1997.

[15] M. Lin and S. Gottschalk, *Collision Detection between Geometric Models: A Survey*, IMA Conference on Mathematics of Surfaces, 1998.

[16] P. Jimnez, F. Thomas and C. Torras, *3D Collision Detection: A Survey*, Computers and Graphics, pages 269–285, 2001.

[17] *PQP Library*, http://www.cs.unc.edu/∼geom/SSV/.

[18] *SWIFT++*, http://www.cs.unc.edu/∼geom/SWIFT++/.

[19] S. Quinlan, *Real-time modification of collision-free paths*, Ph.D.dissertation, Stanford University, 1994.

[20] O. Brock and L. Kavraki, *Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces*, In Proc. ICRA, volume 2, pages 1469-1474, 2001.

[21] F. Schwarzer, M. Saha and J.C. Latombe, *Exact Collision Checking of Robot Paths*, Workshop on Algorithmic Foundations of Robotics, France, 2002.

[22] G. Sanchez and J.C. Latombe, *A single-query bi-directional probabilistic roadmap planner with lazy collision checking*, International Symposium on Robotics Research, Lorne, Victoria, Australia, 2001.

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

[24] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour, *An Integrated Approach to Inverse Kinematics and Path Planning for Redundant Manipulators*, IEEE International Conference on Robotics and Automation, pp. 1874-1879, May 2006.