RDT$^+$: A Parameter-free Algorithm for Exact Motion Planning

Nikolaus Vahrenkamp, Peter Kaiser, Tamim Asfour and Rüdiger Dillmann
Institute for Anthropomatics
Karlsruhe Institute of Technology
Adenauerring 2, 76131 Karlsruhe, Germany
Email: {vahrenkamp,kaiser,asfour,dillmann}@kit.edu

***Abstract*— In this paper parameter-free concepts for exact motion planning are investigated. With the proposed RDT$^+$ approach the collision detection parameters of a Rapidly-exploring Dense Tree (RDT) are automatically adjusted until an exact solution can be found. For efficient planning discrete collision detection routines are used within the RDT planner and by verifying the results with exact collision detection methods, the RDT$^+$ concept allows to compute motions that are guaranteed collision-free. We show the probabilistic completeness of the proposed planner and present an extension for handling narrow passages.**

The algorithms are evaluated in different experiments, including narrow passages and high-dimensional planning problems, that are solved in simulation and on the humanoid robot ARMAR-III.

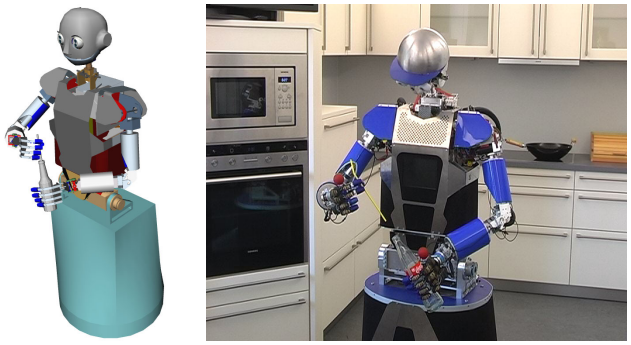

Fig. 1. The RDT$^+$-DE planner is used to compute an exact motion for the humanoid robot ARMAR-III.

I. INTRODUCTION

Sampling-based techniques are used in a wide range of motion planning algorithms. These approaches take advantage of approximating the free space $\mathcal{C}_{free}$, instead of constructing a complete representation of the valid configuration space. The two most popular approaches, the Rapidly-exploring Random Trees (RRTs) [1] and the Probabilistic Roadmaps (PRMs) [2], use sampling techniques to build up a tree or graph in $\mathcal{C}_{free}$, that is refined until a solution, i.e. a collision-free path, can be found. The efficiency of the algorithms strongly depends on the selection of the parameters and usually heuristics are utilized for choosing them. Furthermore most implementations rely on discrete collision detection (DCD) for path validation due to the easy and efficient realization. Thus a parameter has to be specified: the distance d_{col} for validating the collision-free status of a continuous path in C-Space. This value is used to generate discrete samples on the path that are checked for collisions and if none of these samples is in collision, it is assumed that the complete path lies in $\mathcal{C}_{free}$. The parameter d_{col} has to be chosen carefully, since a large value will increase the probability that a collision is missed and by choosing d_{col} too small the efficiency of the planner decreases. Even a conservative selection of d_{col} could lead to situations where thin obstacles are missed and thus a computed solution is not collision-free.

To overcome this limitation exact collision detection routines can be used that give guarantees on the collision status of path segments. In this paper we present the RDT$^+$ approach, a planning concept, where the results of a DCD-based RDT planner [3] are supervised and the collision detection parameter is automatically adjusted until an exact solution is found. We prove that the proposed planner is probabilistically complete by showing that there exists a probability greater zero, that a sequence is sampled for which the RDT$^+$ planner generates a valid solution. Furthermore, the RDT$^+$-*Dynamic Extend* (RDT$^+$-DE) planner, an extension for handling narrow passages in high dimensional configuration spaces, is developed and evaluated on the humanoid robot ARMAR-III [4] (see Fig. 1).

II. RELATED WORK

In [5] the Resolution Adaptive RRT (RA-RRT) is presented. In this work resolution adaptive strategies are used to increase the efficiency of probabilistic path planning. Therefore an adaptive rule for selecting the step size of the expansion process is proposed. The aim of this work is to increase the performance of RRT-based planners for difficult planning problems characterized by the presence of narrow passages in $\mathcal{C}_{free}$. The performance of sampling-based planning approaches can be increased by using the RA-RRT concept, but several parameters, as the initial and the final step size, have to be specified manually.

The Lazy PRM planner [6] initially assumes a configuration space that is completely collision-free so that a roadmap can be quickly constructed in the preprocessing phase. The verification of edges is shiftet to the query phase resulting in a *lazy* refinement of the representation of $\mathcal{C}_{free}$. The performance of the approach relies on the parameter N_{init} that specifies the the initial number of nodes.

Unsupervised adaptive strategies for construction of probabilistic roadmaps (PRM) are presented in [7]. The approach

avoids manual parameter tuning, which is usually needed for approaches based on learning as the Hybrid PRM [8] or the Feature Sensitive Motion Planning Framework [9]. It is demonstrated how to combine both the topology adaptation and sampler adaptation over the planning process in order to learn the best sampling strategy for roadmap construction. Nevertheless, a training set for learning the sampling strategy is needed and due to the complexity of the approach several parameters of the underlying roadmap have to be chosen by hand. In [10] several heuristics for efficient motion planning are presented. An overview of heuristics and parameter tuning techniques for motion planning algorithms can be found in [11] and [12].

A. Discrete vs. Continuous Collision Detection

Methods for discrete collision detection tend to be fast compared with algorithms for continuous collision detection. Nevertheless, checking a complete path in C-Space must be realized by creating samples on the path and when all these samples are collision-free, it is assumed the complete path is collision-free. Depending on the sampling parameter, there is a probability of missing obstacles which can be avoided by utilizing exact collision detection. In this paper, we compare two approaches for exact collision detection with discrete sampling-based collision determination. The first approach is based on the free-bubble concept of Quinlan [13], whereas the second one uses conservative advancement methods for continuous collision detection [14].

1) Free Bubbles: Quinlan introduced the free-bubble concept in [13], to give a guarantee that a path in C-Space is collision-free. There, a method is presented to compute an upper bound for the distance any point on the geometry of the manipulator can travel for a given change in the configuration. By setting this value in relation to the obstacle distance in workspace, the length of a guaranteed collision-free path segment can be determined. To compute the exact collision status of a path in C-Space, intermediate samples have to be generated, so that their *free bubbles* overlap. An efficient implementation can be realized by a recursive bisecting algorithm [15], [16].

2) Continuous Collision Detection: Several approaches are known for continuous collision detection (CCD) when operating on two convex models, both moving from a start pose to a goal pose (see e.g. [17],[18]). In the context of robot motion planning approaches are needed that can handle non-convex objects as well as articulated motions for linked systems, e.g. robot arms.

The approach presented in [19] can handle polygon soups for CCD, but the performance decreased for large rotations. In [20] a CCD-algorithm for articulated motions is presented, that approximates a linear motion of the involved links, where each link of the model should be a polyhedron. Redon et al. extend the approach of [19] to be able to handle articulated models by using *arbitrary in-between motions* to approximate the motion structure of the system [21].

Applying continuous collision detection algorithms to motion planning problems is addressed in [22], where the C2A algorithm [14] is used for performing CCD. The approach is based on conservative advancement, originally developed for convex polytopes as described in [23]. The C2A algorithm can handle polygon soups efficiently, gives a guarantee for the collision status of motions and is available as open source [24]. Since the current implementation just supports rigid motions, no complex kinematic chains of a robot system can be handled. Nevertheless, we show how this approach can be used for planning collision-free 6D trajectories of a *free flying* robot. This enables us to compare this state-of-the-art approach with algorithms based on discrete collision detection, showing how future CCD algorithms covering articulated motions of complex kinematic chains could perform.

III. Parameter-Free Motion Planning

Avoiding parameters for motion planning means that all heuristics that are usually chosen *by hand* have to be eliminated. In the following we identify several stages where heuristic parameters are used for planning:

C-Space: Each dimension of the C-Space represents the position of a related joint of the robot. Since the effect in workspace may differ when moving different joints, the C-Space dimensions have to be unified (e.g. one can think of two joints: a translational and a revolute joint for which the C-Space units are given in mm and radians respectively). This unification can be achieved by weighting, as we proposed in [25]. Since the weights represent the maximal spatial movement, a point on the robot's surface could perform when moving one unit in the corresponding dimension in $\mathcal{C}$, upper bounds can be determined automatically. In Eq. (1), the weight w_i is derived as the maximum over all kinematic chains K starting with the i-th joint. For a kinematic chain K, the maximum extent in workspace of all joints $k \in K$ is summed by finding the two points p and q on the corresponding surface A_k which have maximum distance.

$$w_i = \max_K \left(\sum_{k \in K} \max_{p,q \in A_k} |p - q| \right) \tag{1}$$

With these upper bounds $(w_0, \ldots, w_{n-1})$, the granularity of sampling in C-Space can be adjusted by a resolution parameter d_{sample} representing the maximum allowed workspace displacement of two consecutive samples (see [25]).

Another approach is presented in [26], where the DISP-metric is used to determine the spatial displacement caused by a path in C-Space. The approach in [3] is based on upper bound calculations for workspace displacements and does not rely on any weightings. Thus it can be used for parameter-free unifying of the C-Space dimensions and adequate sampling.

RDT: RRT-based approaches require the specification of a sampling parameter d_{sample}. This parameter is used to create intermediate configurations on path segments, having the advantage, that the search for nearest neighbors can be

implemented easily. Usually another parameter d_{col} is used to specify the step size of discrete collision detection.

The Rapidly-exploring Dense Tree (RDT) planner builds up a tree in $\mathcal{C}_{free}$ by connecting random samples with straight line path segments [3]. In contrast to RRT-based concepts, no intermediate samples along the path are added to the tree structure and thus the tree structure of RDTs is sparse compared to RRTs. Furthermore, the nearest neighbors (NN) search is different to RRT-based planners, since long edges can exists in an RDT. Usually, the NN-algorithm operates on vertices, which is sufficient for RRTs where only short edges exist. When large edges are present, the NN-search has to be able to handle path segments for finding the nearest neighbor of a configuration to a tree in C-Space. Similar to the BiRRT approach bidirectional planning is realized by the BiRDT algorithm that extends two trees until a connecting configuration is found. In case an exact algorithm for determining the collision status of path segments is used, the bidirectional RDT approach does not rely on any parameters. Unfortunately, the performance decreases when using CCD algorithms for path verification (see Sec. V), hence we propose the RDT$^+$ approach where discrete collision detection is used and the parameter d_{col} is automatically adjusted until an exact solution is found.

Narrow Passages: The convergence of sampling-based motion planning algorithms degrades when narrow passages exist in the free C-Space. This is caused by the greedy behavior of PRM- and RRT-based approaches, that need to run a long time until the granularity of the randomized extension steps is fine enough to find a way through narrow sections. Several approaches have been proposed to improve the performance in such cases, most of them propose a customized sampling strategy to be used instead of uniformly distributed sampling:

The Gaussian sampling strategy [27] searches for a pair of two randomly chosen configurations c_1 and c_2. The first configuration is uniformly distributed, the second one Gaussian distributed around c_1. Such a Gaussian pair is considered valid if either $c_1 \in \mathcal{C}_{free}$ and $c_2 \in \mathcal{C}_{obs}$ or the other way around. If a valid Gaussian pair is found, the non-colliding configuration in $\{c_1, c_2\}$ is taken as a new sampling point.

The idea of bridge sampling [28] is to improve Gaussian sampling in matters of not wasting too many sampling points near uninteresting parts of the boundaries of $\mathcal{C}_{free}$. The bridge test searches for a Gaussian pair $\{c_1, c_2\} \subset \mathcal{C}_{obs}$. If such a pair is found, the center of the connecting segment between c_1 and c_2 is taken as a new sampling point.

The dynamic domain RRT [29] (DD-RRT) is a variation of the RRT-algorithm in which every node n of the search tree holds a specific radius $r(n)$, initially set to ∞. A sampling configuration c is only connected to the nearest node c_{near} of the search tree, if $|c - c_{near}| < r(c_{near})$. The radius $r(n)$ is adjusted whenever an attempt to connect the node n to some sampling configuration is made. In the original DD-RRT algorithm the radius is set to a constant R if this attempt fails and otherwise left to ∞. The adaptive dynamic domain RRT (ADD-RRT) in [30] increases or decreases the radius by a factor α whenever it was previously less than ∞ and the attempt succeeded respectively failed.

The use of these approaches introduces new parameters: the variance for the Gaussian distribution when applying the Gauss- or Bridge sampling strategy, the values R and α in case the (A)DD-RRT algorithm is used. Also further approaches as the EET-RRT [31], Retraction-based RRT [32], the disassembly-based motion planner [33] are based on parameters, that have to be chosen carefully.

Since we are interested in a parameter-free algorithm that is able to handle narrow passages, we propose a local adaption of the maximal length of an RDT-based extension step in order to adjust the tree expansion according to the surrounding in C-Space (see Sec. IV-B).

IV. RDT$^+$

In order to construct a parameter-free motion planning algorithm that is efficient and robust we propose the RDT$^+$ approach. This approach combines the advantages of both concepts: avoiding parameters by the BiRDT-based planner and using discrete collision detection for efficient planning. When DCD methods are used, the BiRDT planner relies on the parameter d_{col}, the distance for sampling a continuous path segment for collision detection. This value is supervised and adapted by the RDT$^+$ planner until an exact solution is found. Therefore, d_{col} is initialized with $|c_{goal} - c_{start}|$, a value that will be too large in almost every case, but d_{col} is automatically reduced by the RDT$^+$ planner until an adequate value is reached.

To determine whether the current RDT-instance reported a valid solution, exact path checking routines are used for verifying the results. This has the advantage, that the slow exact path verification methods do not have to be applied for the complete search tree, but only on the solution path, whose length is usually small compared to the whole tree. Depending on this lazy exact path verification the parameter d_{col} of the BiRDT-planner is bisected until a guarantee for an exact solution is achieved (see Alg. 1). Note, that the BiRDT-planner is completely restarted and all performed collision computations are discarded, in case the result was not valid. This behavior is not optimal, but as showed by experiments, the RDT$^+$-planner is significantly faster than a BiRDT-planner performing exact path verification for the whole planning process.

A. Probabilistic Completeness

In this section, the sketch of a proof is presented, showing that the RDT$^+$ algorithm is probabilistically complete. This means, that in case a solution exists the probability of finding it goes to 1 as the running time goes to infinity. Unfortunately it cannot be argued, that the RDT$^+$ planner is probabilistically complete since it successively starts multiple instances of the probabilistically complete RDT planner. This argumentation would hold, when exact path verification routines are used, but since DCD methods are employed by

Algorithm 1: $RDT^{+}(c_{start}, c_{goal})$

$d \leftarrow |c_{goal} - c_{start}|$;
repeat
  $BiRDT \leftarrow$CreateInstance(c_{start}, c_{goal});
  $BiRDT.SetCollisionParameter(d)$;
  $Solution \leftarrow BiRDT.run()$;
  $d \leftarrow d/2$;
until *(ExactPathVerification(Solution))*;
return $Solution$;

the RDT planner, there is no guarantee that a solution path is completely collision-free.

The argumentation used in the following sketch is based on the fact, that there is a probability greater zero that a sequence of points is sampled for which the RDT planner will create a collision-free solution path. If the RDT planner reports a solution that was created by such a sequence, it will pass the verification step of the RDT^{+} planner and the algorithm terminates. For a sequence of samplings that results in an invalid path, the verification step of the RDT^{+} planner rejects the solution and another instance of the RDT planner is started. The use of DCD methods for path verification within the RDT planner does not affect the argumentation: Either, the solution path is completely collision-free and the DCD methods also report this, or the path is in collision which was not detected by the RDT algorithm but by the RDT^{+} planner with its exact collision detection methods.

Sketch of a proof: For clarity we use an uni-directional RDT-Algorithm throughout this sketch. An extension to a bidirectional planner can be realized by considering two uni-directional trees.

We assume that a continuous intersection-free solution path $S \subset \mathcal{C}_{free}$ exists, that starts with c_{start} and ends with c_{goal}. It is known, that $\mathcal{C}_{obst}$ is a closed subset of $\mathcal{C}$ and thus $\mathcal{C}_{free}$ is open. Hence, for every point $p \in S$, there is a radius r_p such that the open ball $B_p := B(p, r_p) \subset \mathcal{C}_{free}$. Because of S being compact, we can apply the Heine-Borel theorem [34] and get a finite subset

$$C \subset \{B_p | p \in S\}$$

which still covers S.

Because of the finiteness of C, we can now choose a radius r, such that:

$$\forall s \in S : B(s, 4r) \subset \bigcup C \tag{2}$$

where $\bigcup C$ is the union of all sets in C. Analogous to the previous argumentation, we can find a finite cover D of S consisting of balls with radius r. Such coverings for an exemplary solution path S in a two dimensional C-Space are visualized in the left sketch of Fig. 2.

Now we present an iterative way to create a cover $E = \{E_1, \dots E_n\} \subset D$ of S, that fulfills the following condition: With $\Gamma_i := E_i \cap E_{i+1}$, the path that is inducted by randomly sampled points

$$s_1 \in \Gamma_1, s_2 \in \Gamma_2, \dots, s_{n-1} \in \Gamma_{n-1} \tag{3}$$

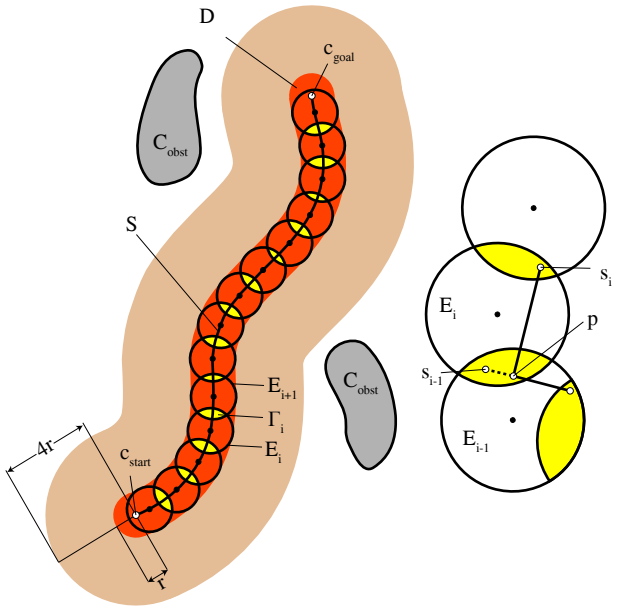

Fig. 2. Left: An exemplary solution path S with both coverings of extend $4r$ and r together with a sequence of overlapping balls. Right: The point s_i is connected to p, the nearest neighbor on the current search tree.

all successively connected by straight segments, lies completely in $\mathcal{C}_{free}$ and hence is a valid solution.

For making things easier, we define $D_p \in D$ as **one** ball of D that contains $p \in S$, knowing, that this choice is not necessarily unique.

1) We start with $E_1 = D_{c_{start}}$ and $E = \{E_1\}$.
2) If c_{goal} is contained in one of the balls in E, we finish the iteration.
3) If not, the set $E = \{E_1, \dots, E_i\}$ of all balls that we added so far together with their union $\bigcup E$ is extended as follows:
   Consider the point p, at which the solution path S quits $\bigcup E$ for the last time. (When S re-enters $\bigcup E$ and thus leaves $\bigcup E$ more than once, the resulting detours can be ignored since we are only interested in a valid solution and not in a path homotopic to S.) The point p has to be in ∂E_i, where E_i is the ball that was most recently added to E. If not, we had already considered p in a previous step of the procedure.
4) All considered sets in D are open, hence $p \notin \bigcup E$, hence there has to be a ball $E_{i+1} := D_p \in D \setminus E$ which contains p and $E_i \cap E_{i+1} \neq \emptyset$
5) $E := E \cup \{E_{i+1}\}$ and go on with step 2.

As far as $\Gamma_1, \Gamma_2, \dots, \Gamma_{n-1}$ are all non-nullsets in $\mathcal{C}$, there is a strictly positive probability, that a sampling-based motion-planning algorithm will sample $s_1, s_2, \dots, s_n$ as defined in Eq. (3).

Since the RDT-algorithm doesn't grow a solution path by connecting the sampling points successively to one long polygonal chain, we have to make sure, that the RDT will construct a suitable solution path out of the sampled points $s_1, \dots, s_n$.

To see this, consider the sampling point s_i. We know that there is a previously sampled point s_{i-1}, such that $|s_i - s_{i-1}| < 2r$ as depicted on the right of Fig. 2. Hence, whichever point p the RDT wants to connect s_i to, its distance to s_i is smaller than $2r$. Thus, the distance from p to the center of the ball containing s_i is at most $3r$. Because of Eq. (2), we know that the connecting segment between p and s_i lies completely in $\mathcal{C}_{free}$. From this it follows that every RDT extension step produces a path segment that is completely collision-free when a sequence of points $s_1, \ldots, s_n$ is sampled.

Finally, to be sure, c_{goal} is connected properly, we take a closer look at random sampling method of the RDT algorithm. As described in [3] there is a given probability that this method returns the goal configuration instead of a uniformly sampled configuration. Hence, there is a probability greater zero that the $(n+1)$th point returned by the sampling method is the first one that is equal to c_{goal} and by arguing in the same way as before we know that the last segment is collision-free.

Note, that this proof does not rely on any collision detection routines and it holds even when no collision detection is performed within the RDT planners. Hence, the use of collision detection methods by the RDT planner does only speed up the planning process and can be seen as an heuristic.

B. Handling Narrow Passages

Several approaches for handling narrow passages can easily be combined with the previously discussed RDT$^+$ algorithm, but introduce new parameters as showed earlier. We will now discuss a parameter-free variation for handling narrow passages that is related to the idea of locally approximating $\mathcal{C}_{free}$ as proposed by the adaptive dynamic domain RRT-algorithm. Instead of changing the sampling strategy as done by the ADD-RRT, where all randomly sampled configurations are discarded when they are not lying within the dynamic domain radius, the ***BiRDT-DynamicExtend (BiRDT-DE)*** approach introduces a new extension algorithm that considers the local approximation of $\mathcal{C}_{free}$.

Algorithm 2: BiRDT-DynamicExtend(c_{start}, c_{goal})

$d_{ext} \leftarrow |c_{goal} - c_{start}|$;
AddConfig ($Tree_1, c_{start}, d_{ext}$);
AddConfig ($Tree_2, c_{goal}, d_{ext}$);
while *(*$!TimeOut()$*)* **do**
  $c_{random} \leftarrow$ SampleRandom();
  $(c', Success_E) \leftarrow$ ExtendRDT($Tree_1, c_{random}$);
  $Success_C \leftarrow$ ConnectRDT($Tree_1, c'$);
  if *(*$Success_E$ & $Success_C$*)* **then**
    return BuildSolution($Tree_1, Tree_2$);
  Swap($Tree_1, Tree_2$);
end
return $NULL$;

The BiRDT-DE approach is able to handle narrow passages by locally adapting the extend parameter d_{ext} that is stored for each node of the search tree. Initially this value is set to $|c_{goal} - c_{start}|$ and by bisecting it when an extension step fails or doubling it when it was successful, the shape of $\mathcal{C}_{free}$ is approximated. The search loop in Alg. 2 is based on the classical RRT-CONNECT approach, described in [1], but the adapted *ExtendRDT* and *ConnectRDT* methods are used for extending the search trees. In Alg. 3, the EXTEND step is described: First c_{NN}, the nearest configuration on the edge e is determined and by considering the local extend radius of the edge e, the length of the extension step is limited to d_{ext}. The CONNECT method in Alg. 4 is used to build new edges of the search tree. If the path between c_{NN} and c is collision-free, a new edge is added to the tree and the corresponding extend radii of e are doubled. Otherwise, the edge to the maximal valid position on the path c_{NN} to c is added and the extend radius is bisected. The effects of both methods are depicted in Fig. 3: In the left figure the *Extend-RDT* method is used to determine d_{ext} and c' in order to limit the length of the extend step. The right figure shows how the extend radii are bisected by the *Connect-RDT* method when an edge is not completely collision-free.

By using the RDT-DE approach within the RDT$^+$ concept, the RDT$^+$-DE algorithm is defined similar to Alg. 1.

Algorithm 3: ExtendRDT($Tree, c$)

$e \leftarrow$ NearestNeighborEdge($Tree, c$);
$c_{NN} \leftarrow$ NearestNeighborOnEdge($Tree, e, c$);
$d_{ext} \leftarrow (e.c_1.d_{ext} + e.c_2.d_{ext})/2$;
$c' \leftarrow \begin{cases} c & , |c - c_{NN}| \leq d_{ext} \\ c_{NN} + \frac{d_{ext}(c - c_{NN})}{|c - c_{NN}|} & , otherwise \end{cases}$
if *(CollisionFree(*c_{NN}, c'*))* **then**
  return (c',ConnectRDT($Tree, c'$));
return ($c', false$);

Algorithm 4: ConnectRDT($Tree, c$)

$e \leftarrow$ NearestNeighborEdge($Tree, c$);
$c_{NN} \leftarrow$ NearestNeighborOnEdge($Tree, e, c$);
$d_{ext} \leftarrow (e.c_1.d_{ext} + e.c_2.d_{ext})/2$;
if *(CollisionFree(*c_{NN}, c*))* **then**
  $e_{new} \leftarrow$ AddEdge($Tree, e, c_{NN}, c$);
  $e.c_1.d_{ext} \leftarrow 2e.c_1.d_{ext}$;
  $e.c_2.d_{ext} \leftarrow 2e.c_2.d_{ext}$;
  return $true$;
else
  $c' \leftarrow$ GetMaxValidPosition(c_{NN}, c);
  $e_{new} \leftarrow$ AddEdge($Tree, e, c_{NN}, c'$);
  $e.c_1.d_{ext} \leftarrow e.c_1.d_{ext}/2$;
  $e.c_2.d_{ext} \leftarrow e.c_2.d_{ext}/2$;
  $e_{new}.c_1.d_{ext} \leftarrow d_{ext}/2$;
  $e_{new}.c_2.d_{ext} \leftarrow d_{ext}/2$;
  return $false$;
end

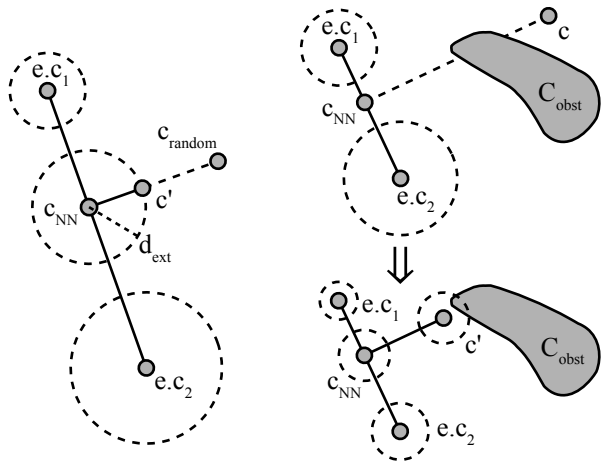

Fig. 3. Left: The EXTEND method cuts the extension path to the local extend radius d_{ext} of the edge. Right: The CONNECT method reduces the extend radii due to an unsuccessful connection.

V. Evaluation

A. A free flying spaceship: 6D C-Space

In this experiment[1], a path for a free flying spaceship in a scene with 500 randomly placed obstacles is searched. The C-Space is six-dimensional, covering the position and orientation of the spaceship. In Table I the results of several planning approaches are listed. The *BiRRT* planner (row 1) was setup with a sampling size of 20 and 5 for adding new configurations and for DCD-sampling respectively. Note that these values are given for the translational components and the rotational dimensions of the C-Space are weighted by the factor 62.9, which is the extend of the spaceship. The results point out, that the planner is able to find solutions very quickly due to the discrete collision detection, but 26.7% of the results are not collision-free. Different parameter setups for the *BiRRT* planner are evaluated (row 2 and 3 in Table I) and it can be seen, that more reliable results can be achieved by decreasing the sampling sizes at the cost of performance. The BiRDT approach (row 4 and 5 in Table I) is able to find exact motions, but since exact algorithms for path verification are used, the planning time has increased. Best results can be achieved with the RDT^+ planners (row 6 and 7 in Table I), where most of the time is spent for verifying the results of the underlying BiRDT-based planners. The difference in the planning time between the RDT^+_{FB} and the RDT^+_{C2A} approach is caused by the use of different algorithms for exact path verification. A comparison to the RDT^+-DE concept is given in the last two rows. Here, the *Dynamic Extend* extension for handling narrow passages is used and it can be seen, that this approach introduces some overhead in planning time compared to the RDT^+ planners when no narrow passages are present. A planned motion together with the search tree is depicted in Fig. 4.

[1]All experiments have been carried out on a 2.4 GHz System

TABLE I

Average performance of the different planning approaches.

	Run Time	Valid	# CD
$\text{BiRRT}_{(5,20)}$	81ms	73.3%	2 011
$\text{BiRRT}_{(1,5)}$	417ms	93.3%	11 856
$\text{BiRRT}_{(0.1,1)}$	4 650ms	100.0%	149 849
BiRDT_{FB}	3 444ms	100.0%	1 922
BiRDT_{C2A}	2 357ms	100.0%	333
RDT^+_{FB}	1 018ms	100.0%	5 046
RDT^+_{C2A}	256ms	100.0%	3 192
$\text{RDT}^+\text{-DE}_{FB}$	1 813ms	100.0%	4 299
$\text{RDT}^+\text{-DE}_{C2A}$	613ms	100.0%	5 407

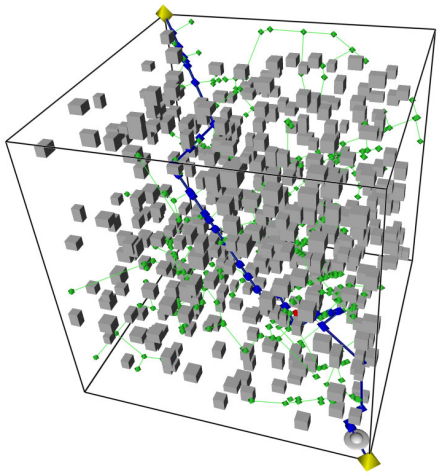

Fig. 4. The RDT^+ planner was used to compute an exact motion through a scene with 500 randomly placed obstacles.

B. Straw and Bottle

In this experiment the RDT^+-DE planner is used to compute a collision-free motion in a high-dimensional C-Space with the goal of putting a straw in a bottle. Therefore, the humanoid robot ARMAR-III holds both objects in its hands as shown left in Fig. 5. Since both arms are used for planning, the resulting C-Space is 14-dimensional and both arms with hands and objects together with the static part of the robot are mutually considered as obstacles to avoid self-collisions. Please note, that the upright orientation of the bottle is not considered as a constraint in this experiment, since this is not topic of this work. A visualization of the resulting search trees in workspace can be seen in the right image of Fig. 5 and the execution of a planned motion on ARMAR-III is shown in Fig. 6. For accurate execution of the planned bimanual motion, visual servoing techniques are used as proposed in [35].

Since the C2A approach does only support straight line movements, no articulated multi-body systems, where succeeding joints perform curved motions, can be handled with this approach. Thus, only the *free bubble* (FB) algorithm can be used for this planning setup. Please note that the FB algorithm can be used even when multiple kinematic chains of one system are accessed, since the motion bound for the complete system is based on the accumulated maximal allowed motions of all joints.

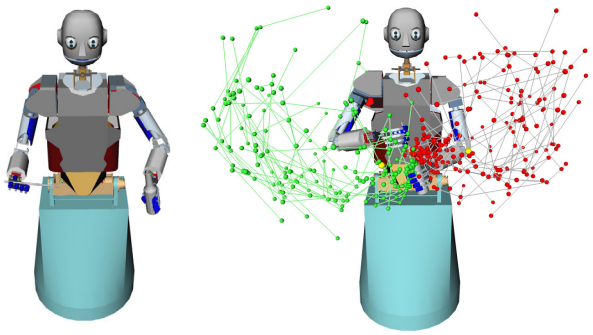

Fig. 5. The start and goal configuration of the ARMAR-III experiment. The search trees, generated by the RDT$^+$-DE planner are visualized in the right figure.

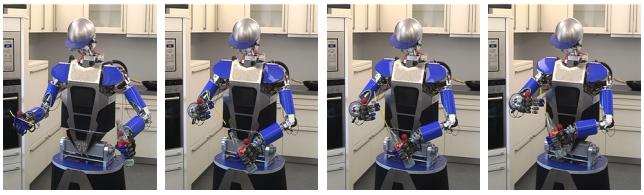

Fig. 6. Execution of the planned motion on ARMAR-III.

As shown in Table II the planning problem could not be solved with the Adaptive Domain RRT, due to the long time that is needed to find suitable samples in this high dimensional configuration space. This is caused by the sampling strategy of the ADD-RRT approach, where a randomly chosen sample is discarded when it lies outside the dynamic domain radius of it's nearest neighbor and in the given C-Space the dynamic domain radii of the search tree nodes around the goal configuration are small compared to the whole C-Space. Therefore, finding a single configuration for extending the tree often resulted in discarding more than 50 million samples and it was not possible to find a parameter setup that allows to plan the motion within 20 minutes.

In the following rows (2-4) the results of a BiRRT planner using the bridge sampling strategy are presented. Here, the variance was set to 20, which is related to the weighted C-Space dimensions, so that the size of the opening of the bottle is approximately represented. With none of the parameter setups (the first index number shows the DCD collision detection parameter, the second one the sampling parameter) a satisfying result can be achieved. The percentage of collision-free motions reached 91.8 % in the last BiRRT-Bridge experiment (row 4), but the planning time increased to more than 17 minutes on average.

Exact motions are planned with the BiRDT-DE approach, using the *free bubble* approach for path verification (row 5 of Table II). The planning time was measured with 6 minutes on average and the distance computation routines were called more than 25 thousand times. The bad performance is mainly caused by the slow FB path verification process: Due to the large numbers of involved joints and the low distances between the static part of the robot and the arms, the motion bounds that are used to calculate bubbles of free space tend to get small and a huge number of intermediate configurations have to be checked until the verification is passed.

TABLE II

STRAW AND BOTTLE: AVERAGE PERFORMANCE OF THE DIFFERENT PLANNING APPROACHES.

	Run Time	Valid	# CD	# Dist
BiRRT-ADD$_{(5,20)}$	-	0.0%	-	-
BiRRT-Bridge$_{(5,20)}$	24.1s	80.0%	206.7K	0
BiRRT-Bridge$_{(0.05,5)}$	234.3s	83.3%	2.1M	0
BiRRT-Bridge$_{(0.01,2)}$	1049.5s	91.8%	9.4M	0
BiRDT-DE$_{FB}$	360.1s	100.0%	17.5K	25.5K
RDT$^+$-DE$_{FB}$	61.4s	100.0%	40.8K	3.7K

In the last row the results of the RDT$^+$-DE planner using the *free bubble* approach for path verification are shown. Since the exact path verification is not performed on the complete search tree, but on the solution paths, the performance is better compared to the BiRDT-DE algorithm. Also in this setup, the verification process takes a long time compared to planning the motion due to the described limitations of the FB approach. In Table III the accumulated run times for planning and verification for all steps of the RDT$^+$ planner can be seen. The verification time for the straw setup is more than 13 times higher than the overall time needed for planning. Hence, by using a more efficient CCD algorithm for approximating motion bounds of a large kinematic chain, the efficiency of the RDT$^+$ concept can be improved. Nevertheless, with the proposed planner, a guaranteed collision-free motion for a high dimensional planning problem with a narrow passage can be planned in 61.4 seconds on average, which is six times faster than the BiRDT planner that was enhanced by the dynamic extend approach.

TABLE III

DETAILED OVERVIEW OF THE RDT$^+$-DE PLANNING RESULTS.

	Acc. Time Planning	Acc. Time Verification	Avg DCD Sampling Size
Space Ship (6D)	228.3ms	1 579.0ms	8.6
Straw (14D)	4 302.9ms	57 127.6ms	7.4

VI. CONCLUSION AND FUTURE WORK

In this work the RDT$^+$ algorithm for parameter-free planning of exact motions was presented. The RDT-concept is used to avoid an heuristically based definition of sampling parameters and due to resolution adaptive techniques, fast discrete collision detection can be used. Therefore, the RDT$^+$ approach successively decreases the sampling distance for discrete collision detection and queries the results of a classical BiRDT planner. The resulting motions are checked with exact path verification algorithms, in order to guarantee a collision-free execution. When this verification fails and the solution of the current planning phase results in a collision, the sampling parameter is reduced to retrieve a finer approximation in the next step. For exact path verification,

two concepts are investigated: The *free bubble* approach and a native implementation of continuous collision detection.

To handle narrow passages the parameter-free BiRDT-DE planner was introduced and combined with the resolution adaptive concept. The resulting RDT$^+$-DE planner was evaluated in different scenarios, including high-dimensional planning problems with narrow passages.

The experiments showed, that the bottleneck of exact motion planning in high-dimensional configuration spaces is the algorithm for exact path verification, even when resolution adaptive techniques are used. Thus, future work will address the integration of CCD algorithms for articulated multi-body systems in order to improve the performance of the RDT$^+$-DE planner.

VII. ACKNOWLEDGMENTS

The work described in this paper was partially conducted within the EU Cognitive Systems project GRASP (IST-FP7-IP-215821) funded by the European Commission and the German Humanoid Research project SFB588 funded by the German Research Foundation (DFG: Deutsche Forschungsgemeinschaft). The authors would like to thank the C2A team for offering their CCD-software as open source [24] and especially Min Tang for her help on integration the library in Simox [16].

REFERENCES

[1] 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.

[2] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, 1994.

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

[4] 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.

[5] T. Sartini, M. Vendittelli, and G. Oriolo, "A resolution-adaptive strategy for probabilistic motion planning," in *Automation Congress, Proceedings of the 5th Biannual World*, vol. 14, 2002, pp. 591–596.

[6] R. Bohlin and L. Kavraki, "Path planning using lazy prm," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 1, 2000, pp. 521 –528.

[7] L. Tapia, S. Thomas, B. Boyd, and N. Amato, "An unsupervised adaptive strategy for constructing probabilistic roadmaps," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, may. 2009, pp. 4037–4044.

[8] H. Kurniawati and D. Hsu, "Workspace-based connectivity oracle: An adaptive sampling strategy for prm planning," in *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR)*. SpringerVerlag, 2006.

[9] M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. M. Amato, "A machine learning approach for feature-sensitive motion planning," in *Algorithmic Foundations of Robotics VI*, 2004, pp. 361–376.

[10] C. Eldershaw, "Heuristic algorithms for motion planning," Ph.D. dissertation, University of Oxford, UK, 2001.

[11] S. R. Lindemann and S. M. LaValle, "Current issues in sampling-based motion planning," in *Robotics Research: The Eleventh International Symposium*, P. Dario and R. Chatila, Eds., 2005, pp. 36–54.

[12] I. Sucan and L. Kavraki, "On the implementation of single-query sampling-based motion planners," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, may. 2010, pp. 2005–2011.

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

[14] M. Tang, Y. Kim, and D. Manocha, "C2A: Controlled conservative advancement for continuous collision detection of polygonal models," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, 12-17 2009, pp. 849–854.

[15] F. Schwarzer, M. Saha, and J. claude Latombe, "Adaptive dynamic collision checking for single and multiple articulated robots in complex environments," *IEEE Tr. on Robotics*, vol. 21, pp. 338–353, 2005.

[16] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Simox: A simulation and motion planning toolbox for C++," Karlsruhe Institute of Technology (KIT), Tech. Rep., 2010, [Online] http://www.sourceforge.net/projects/simox.

[17] S. Redon, A. Kheddar, and S. Coquillart, "Fast continuous collision detection between rigid bodies," *Comput. Graph. Forum*, vol. 21, no. 3, 2002.

[18] G. van den Bergen, "Ray casting against general convex objects with application to continuous collision detection," in *Journal of Graphics Tools*, 2004.

[19] S. Redon, A. Kheddar, and S. Coquillart, "An algebraic solution to the problem of collision detection for rigid polyhedral objects," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 4, 2000, pp. 3733–3738 vol.4.

[20] X. Zhang, S. Redon, M. Lee, and Y. J. Kim, "Continuous collision detection for articulated models using taylor models and temporal culling," *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, vol. 26, no. 3, p. 15, 2007.

[21] S. Redon, Y. J. Kim, M. C. Lin, and D. Manocha, "Fast continuous collision detection for articulated models," in *SM '04: Proceedings of the ninth ACM symposium on Solid modeling and applications*, Switzerland, 2004, pp. 145–156.

[22] M. Tang, Y. J. Kim, and D. Manocha, "Efficient local planning using connection collision query," Ewha Womans University, Korea, Tech. Rep., 2010.

[23] B. V. Mirtich, "Impulse-based dynamic simulation of rigid body systems," Ph.D. dissertation, University of California, Berkeley, 1996.

[24] M. Tang, Y. J. Kim, and D. Manocha, "C2A: Controlled conservative advancement for continuous collision detection of polygonal models," 2010. [Online]. Available: http://http://graphics.ewha.ac.kr/C2A/

[25] N. Vahrenkamp, C. Scheurer, T. Asfour, R. Dillmann, and J. Kuffner, "Adaptive motion planning for humanoid robots," in *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, 2008.

[26] L. Zhang, Y. J. Kim, and D. Manocha, "Efficient distance computation in configuration space," *Computer Aided Geometric Design*, vol. 25, no. 7, pp. 489–502, 2008, solid and Physical Modeling and Applications Symposium 2007.

[27] V. Boor, M. Overmars, and A. van der Stappen, "The gaussian sampling strategy for probabilistic roadmap planners," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, 1999, pp. 1018–1023 vol.2.

[28] Z. Sun, D. Hsu, T. Jiang, H. Kurniawati, and J. Reif, "Narrow passage sampling for probabilistic roadmap planning," *Robotics, IEEE Transactions on*, vol. 21, no. 6, pp. 1105–1115, dec. 2005.

[29] A. Yershova, L. Jaillet, T. Simeon, and S. LaValle, "Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, apr. 2005, pp. 3856–3861.

[30] L. Jaillet, A. Yershova, S. La Valle, and T. Simeon, "Adaptive tuning of the sampling domain for dynamic-domain RRTs," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, aug. 2005, pp. 2851–2856.

[31] M. Rickert, O. Brock, and A. Knoll, "Balancing exploration and exploitation in motion planning," in *Robotics and Automation, IEEE International Conference on*, may. 2008, pp. 2812–2817.

[32] L. Zhang and D. Manocha, "An efficient retraction-based RRT planner," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, may 2008, pp. 3743–3750.

[33] Y. Yang and O. Brock, "Efficient motion planning based on disassembly," in *Robotics: Science and Systems (RSS)*, 2005.

[34] H. Jeffreys and B. S. Jeffreys, *Methods of Mathematical Physics*. Cambridge, England: Cambridge University Press, 1988.

[35] N. Vahrenkamp, C. Böge, K. Welke, T. Asfour, J. Walter, and R. Dillmann, "Visual servoing for dual arm motions on a humanoid robot," in *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, Dec. 2009.