An Efficient Retraction-based RRT Planner

Liangjun Zhang and Dinesh Manocha
Dept. of Computer Science, University of North Carolina at Chapel Hill
http://gamma.cs.unc.edu/RRRT/

***Abstract*— We present a novel optimization-based retraction algorithm to improve the performance of sample-based planners in narrow passages for 3D rigid robots. The retraction step is formulated as an optimization problem using an appropriate distance metric in the configuration space. Our algorithm computes samples near the boundary of C-obstacle using local contact analysis and uses those samples to improve the performance of RRT planners in narrow passages. We analyze the performance of our planner using Voronoi diagrams and show that the tree can grow closely towards any randomly generated sample. Our algorithm is general and applicable to all polygonal models. In practice, we observe significant speedups over prior RRT planners on challenging scenarios with narrow passages.**

I. Introduction

Sample-based planning algorithms have been widely used to compute collision-free paths for robots in complex environments. These algorithms generate samples using randomized techniques and attempt to capture the connectivity of the free space by joining nearby samples via collision-free paths. The connectivity is represented using probabilistic roadmaps (PRMs) [1] or rapidly-exploring random trees (RRTs) [2], [3].

The performance of sample-based planning algorithms may degrade if the free space of the robot has narrow passages, i.e. small regions whose removal or perturbation can change the connectivity of the free space. These regions can also be characterized in terms of poor visibility properties [4]. Moreover, the narrow passage problem becomes more severe for RRT-based planners as compared to other randomized methods.

Many techniques have been proposed in the literature to improve the performance of these planners in narrow passages. These include use of workspace information to guide the sampling, adaption of the sample distribution based on history, and retraction-based planning. In this paper, we primarily focus on improving the performance of retraction-based planners.

One of the main steps in retraction-based planning is to retract a sample or a configuration to a more desirable region of the free space. This includes moving the samples close to the boundary of the configuration space obstacle (C-obstacle) or near the medial axes of the free space. One specific retraction strategy is to retract any *in-colliding* configuration (a configuration in C-obstacle) to the closest boundary point of C-obstacle. In practice, this is equivalent to computing the *penetration depth*, which is used to quantify the interpenetration between overlapping objects. However, exact computation of global penetration depth has very high complexity [5]. As a result, prior planners use simple heuristics to perform the retraction step, and their performance varies with the shape of the robot and the obstacles, and their relative placement.

Other approaches for handling narrow passages include dilation-based planners. These algorithms dilate the free space by considering samples that lie inside the C-obstacle space and are close to its boundary. However, most of dilation-based planners can be hard to implement as they need a robust technique to perform the dilation or shrinking operation on general polygonal models. Moreover, these techniques are mainly limited to PRM planners. Overall, no good algorithms are known for performing the retraction step efficiently on general models or combining them effectively with RRT planners.

Main Results: We present a new optimization-based retraction algorithm and an enhanced RRT planner. We formulate the retraction step as a constrained optimization problem using an appropriate distance metric in the configuration space. Our optimization algorithm performs iterative refinement in the contact space to compute a (local) minima of the objective function using local contact analysis.

We use our optimization-based retraction algorithm to improve the performance of RRT planners. We retract the samples so that they are more likely to be connected to the nearest nodes of the tree by the local planning algorithm. The resulting tree in our planner grows closely towards randomly generated samples, including in-colliding as well as free configurations. As compared to prior RRT planners, our enhanced planner generates more samples near the contact space and in the narrow passages. We analyze the performance of our planner using the Voronoi diagram of the nodes in the tree and highlight scenarios where our planner can work well.

We have implemented our planner and highlight its performance on difficult scenarios with narrow passages. As compared to the basic RRT algorithm, we observe significant improvement in the running time and the number of generated samples. Moreover, our algorithm is general and makes no assumption about input models, or their connectivity or topology.

Organization: The rest of the paper is organized as follows. In Section 2, we provide a brief survey of related work in sample-based planning. In Section 3, we present our optimization-based retraction algorithm, and describe our

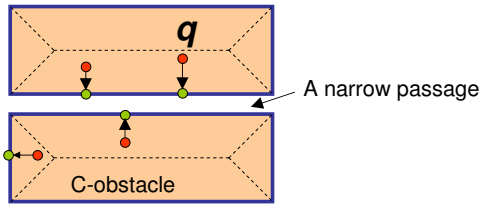


Fig. 1. **Retraction strategy for sampling narrow passage:** *The basic idea is to retract a randomly generated configuration to a more desirable region in the free space. A desirable location for retraction of an in-colliding configuration* $\mathbf{q}$ *is the closest boundary point in the free space. Intuitively, the given in-colliding configuration, e.g.* $\mathbf{q}$ *which is close to the boundary, is retracted into the narrow passage. This increases the number of samples in this narrow passage and is almost independent of the volume of the narrow passage [13], [16].*

enhanced RRT planner in Section 4. We analyze the performance of our planner using Voronoi diagrams in Section 5. We highlight its performance and compare our enhanced planner with other RRT planners in Section 6. We discuss some limitations of our approach in Section 7.

II. Related Work

In this section, we give a brief overview of prior work in sample-based motion planning and retraction-based methods.

A. Sample-based Planning

The sample-based approaches, such as probabilistic roadmaps (PRM) [1] and rapidly-exploring random trees (RRT) [2], [3], have been successfully used to solve high degree-of-freedom motion planning problems. To improve the performance of these planners in narrow passages, many sampling strategies have been proposed. See [4] for a recent survey. These include use of workspace information to guide the sampling [6], [7], filters to reject samples [8], [9], [10], adaption of the sampling distribution based on history [11], and retraction-based approaches.

B. Retraction-based Motion Planning

The retraction-based approaches have been widely used to improve the performance of sample-based planners in narrow passages [12], [13], [14], [15], [16], [17], [18]. The main idea is to retract a randomly generated configuration towards a more desirable region, e.g. to the closest point on the boundary of C-obstacle (Fig. 1) or the medial axis of the free space.

The main challenge in retraction-based approaches is that the retraction step may involve complicated or non-trivial computation. For example, computing the closest boundary point for an in-colliding configuration boils down to generalized penetration depth computation based on an appropriate distance metric. The computation of globally optimum penetration depth has high complexity [5]. As a result, most algorithms use heuristics to compute samples near the boundary of C-obstacle or in the contact space [12], [15], [16], [19]. Other approaches include dilation-based planning [13], and current practical solutions for them compute an approximate medial axes of the model [20], shrink the boundary using tetrahedral decompositions [21], or estimate the bound of the motion for the moving robot [22]. In practice, except [22], most of these techniques are limited to closed models and can be susceptible to robustness issues and degeneracies.

C. Contact Space Planning

Many retraction-based approaches tend to generate more samples near the *contact space*, the subset of the configuration space (C-space), which consists of the configurations when the robot touches one or more obstacles without any penetration. Contact space planning has been shown useful for handling narrow passages [14], along with manipulator planning and compliant motion planning. There is considerable work on contact modeling using the geometric or algebraic formulation, sampling, and local compliant planning [23], [24], [25], [26].

III. Optimization-based Retraction

In this section, we present our optimization-based retraction algorithm for sample generation. We use this algorithm to improve the performance of RRT planners in Section 4. Given an in-colliding sample, our algorithm retracts this sample to a more desirable location, i.e. the closest point on the boundary of C-obstacle or *contact space*. This idea is similar to other retraction-based sampling strategies such as the one used in OBPRM [12], which also tend to generate samples near the contact space to improve the performance of the planner in narrow passages. The main difference is that we perform the retraction step using iterative optimization.

A. The Retraction Step

As shown in Fig. 2, given an in-colliding sample $\mathbf{q}_r$, the *retraction step* is to compute its closest boundary point $\mathbf{q_m}$, which can be formally defined as:

$$\mathbf{q_m} = \arg\min_{\mathbf{q}} \delta(\mathbf{q}, \mathbf{q}_r), \mathbf{q} \in \mathcal{C}_{contact}, \tag{1}$$

where δ is a distance metric defined in the configuration space of the robot, and the configuration $\mathbf{q}$ lies in the contact space $\mathcal{C}_{contact}$.

An important issue in performing the retraction step is the choice of an appropriate distance metric δ in C-space. In practice, it is hard to define a distance metric that can meaningfully combine the translational and rotational components in C-space. In our formulation, we use *model-dependent distance metrics* such as the DISP distance metric, since it does not involve any weighting factor to combine the translational and rotational components [5].

B. Optimization-based Retraction Algorithm

We formulate the retraction computation (Eq. 1) as a constrained optimization problem. The objective function is based on the distance metric δ in C-space, and the constraint is that the resulting configuration needs to lie in $\mathcal{C}_{contact}$. As shown in Fig. 2, to retract a given in-colliding sample $\mathbf{q}_r$, our method starts with a *non-colliding* sample $\mathbf{q}_n$ (either collision-free or in the contact space) as the initial guess. The

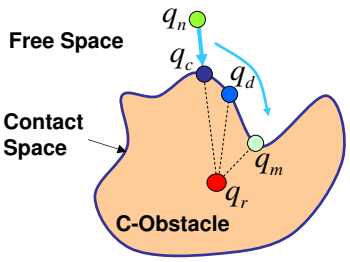


Fig. 2. **Optimization-based Retraction:** *Given an in-colliding sample* $\mathbf{q_r}$*, our algorithm retracts it to the locally closest point* $\mathbf{q_m}$ *on the boundary of C-obstacle by iterative optimization. In this case,* $\mathbf{q_n}$ *is the initial guess, while* $\mathbf{q_c}$ *and* $\mathbf{q_d}$ *are intermediate samples during the optimization.*

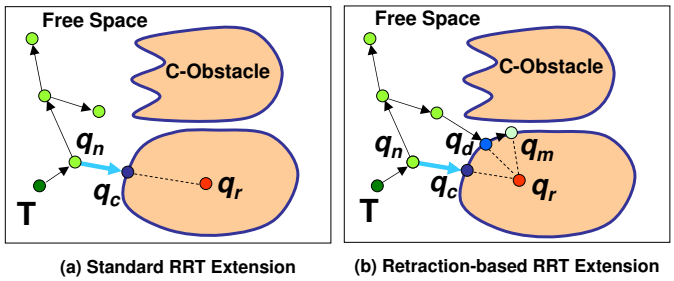
algorithm then performs the following steps in an iterative manner:

1) Project $\mathbf{q}_n$ into the contact space in order to generate a sample $\mathbf{q}_c$ in the contact space;
2) Perform a *contact query*, i.e. compute the closest feature pairs within a tolerance distance between the robot at $\mathbf{q}_c$ and the obstacles;
3) Searching over the local contact space formed by the closest feature pairs, compute a new non-colliding sample $\mathbf{q_d}$, which locally minimizes the distance to the sample $\mathbf{q}_r$ according to a distance metric δ;
4) Assign $\mathbf{q}_n = \mathbf{q_d}$, and go to Step 1.

These steps are iterated until the distance to $\mathbf{q}_r$ cannot be further reduced, which means a sample $\mathbf{q_m}$ realizing a local minima is found, or the maximum number of iterations has been reached. We use the symbol S to represent the sequence of samples $\mathbf{q}_n$, $\mathbf{q}_c$, and $\mathbf{q_d}$ generated by each iteration, excluding the duplicated or in-colliding samples. The distance of every sample in the sequence S to $\mathbf{q}_r$ strictly decreases, i.e. $\delta(\mathbf{q_i}, \mathbf{q}_r) > \delta(\mathbf{q_{i+1}}, \mathbf{q}_r)$, and it monotonically approaches the local minima $\delta(\mathbf{q_m}, \mathbf{q}_r)$. The generated samples in S can be used by any sample-based planner.

Our algorithm can efficiently optimize over the contact space and compute a local minima. We briefly describe some issues in implementing this algorithm. In Step 1, in order to compute the $\mathbf{q}_c$ in the contact space, we reduce the problem to computing the configuration when the robot barely touches the obstacles along the interpolating motion from $\mathbf{q}_n$ to $\mathbf{q}_r$. We use a binary search to compute $\mathbf{q}_c$ since $\mathbf{q}_n$ is a non-colliding configuration and $\mathbf{q}_r$ is an in-colliding one. In Section 6, we address the problem of computing the closest feature pairs in Step 2. In Step 3, we randomly generate samples over the local contact space formed by the closest feature pairs [5]. We discard in-colliding samples using collision checking and compute the non-colliding sample $\mathbf{q_d}$ that is closest to $\mathbf{q}_r$.

The optimization algorithm only needs to perform collision detection, along with *local contact analysis* to sample and search over the local contact space. Such computation can be implemented for any type of polygonal models, including polygon soup models. As a result, our algorithm is general and applicable to all general polygonal models.



(a) Standard RRT Extension (b) Retraction-based RRT Extension

Fig. 3. **RRT Extension:** *(a) Given a randomly generated configuration* $\mathbf{q_r}$*, the standard RRT extension scheme grows the tree - T from its nearest node* $\mathbf{q_n}$ *towards* $\mathbf{q_r}$*, stopping at the configuration* $\mathbf{q_c}$ *on the boundary. (b) In our retraction-based extension, we retract* $\mathbf{q_r}$ *to the free space by using optimization. The retraction step generates a sequence of non-collision configurations* $S = \{\mathbf{q_c}, \mathbf{q_d}, ..., \mathbf{q_m}\}$*, where* $\mathbf{q_m}$ *is a local minima point of the distance* δ *to* $\mathbf{q_r}$*. We then extend the tree to every configuration in* S *using the standard RRT extension. Therefore, the tree in our algorithm can grow towards* $\mathbf{q}_r$ *closely.*

IV. Retraction-based RRT Planner

In this section, we use the optimization-based retraction algorithm to improve the performance of the rapidly-exploring random tree planners (RRT). Prior retraction-based sampling strategies have mainly been applied to PRM planners and only retract the samples that lie in C-obstacle space. In our case, we retract many of the generated samples including the ones that belong to the free space.

A. RRT Planner

The RRT algorithm explores the free space by randomly sampling and building a tree (Fig. 3-(a)). RRT's are used to search high-dimensional spaces with both algebraic constraints (arising from obstacles) and differential constraints (e.g. the non-holonomic constraints). The basic RRT algorithm is as follows. Starting with a tree T with a root node, the algorithm iteratively adds more nodes to the tree. During each iteration, a configuration $\mathbf{q}_r$ is randomly generated, and the basic RRT algorithm attempts to connect the nearest node $\mathbf{q}_n$ in the tree T to $\mathbf{q}_r$ by a straight line in the configuration space (Fig. 3-(a)). If the configuration $\mathbf{q}_r$ and $\mathbf{q}_n$ can be connected via a collision-free path, the tree is extended from $\mathbf{q}_r$ to $\mathbf{q}_n$ and grows. Otherwise, the planner computes $\mathbf{q}_c$, the first *in-contact* configuration (a configuration in the contact space) on the straight line from $\mathbf{q}_n$ to $\mathbf{q}_r$. The tree then extends to $\mathbf{q}_c$. We refer to this way of growing the tree as the *standard RRT extension*.

One of the challenges for RRT planners is to generate samples in narrow passages of the free space. Moreover, though the basic RRT planner can perform a biased search towards regions not yet visited, such bias does not take into account the obstacles in the environment. Therefore, the basic RRT planner can have difficulty growing out of narrow passages in cluttered environments.

B. Retraction-based RRT

We use the retraction-based algorithm to improve the performance of RRT planners, especially in narrow passages. Our modified RRT algorithm (Alg. 1) proceeds as follows:

Algorithm 1: Retraction-based RRT Extension

Input: $T = \{V, E\}$ - an RRT
$\mathbf{q}_r$ - a randomly generated configuration in C-space
Output: T, an extended RRT
begin
  $\mathbf{q}_n \longleftarrow$ the nearest neighbor of $\mathbf{q}_r$ in T
  if $(\mathbf{q}_n, \mathbf{q}_r)$ *is a collision-free path* **then**
    $T.AddVertex(\mathbf{q}_r), T.AddEdge(\mathbf{q}_n, \mathbf{q}_r)$
  else
    // To get a set of non-colliding configurations using
    // the retraction step; using $\mathbf{q}_n$ as the initial guess
    $S \longleftarrow Optimize(\mathbf{q}_r, \mathbf{q}_n)$
    for $\mathbf{q}_i \in S$ **do**
      $Standard\ RRT\ Extension(T, \mathbf{q}_i)$
  return T
end

1) Given the randomly generated configuration $\mathbf{q}_r$, free or in-colliding, we compute the nearest node $\mathbf{q}_n$ in the tree;
2) Check whether $\mathbf{q}_r$ and $\mathbf{q}_n$ can be connected via a collision-free path. If there is such a path, grow the tree from $\mathbf{q}_n$ to $\mathbf{q}_r$.
3) Otherwise, we retract the sample $\mathbf{q}_r$, as shown in Fig. 3-(b):
   a) Using $\mathbf{q}_n$ as the initial guess, we apply the optimization-based retraction step presented in Section 3. The retraction step generates a sequence S of non-collision configurations, approaching the closest boundary point of $\mathbf{q}_r$;
   b) For every configuration in S, our algorithm performs the standard RRT extension.

We refer to our scheme of growing the tree as *retraction-based extension*. There are several benefits of our enhanced RRT planner using this new extension scheme. First, the sampled configurations that are close to the narrow passages are likely to be retracted into the narrow passages. Consequently, our extended planner generates more samples in narrow passages. Secondly, the tree grows closely towards any randomly generated configuration, significantly improving the bias of the growth of the tree towards regions not yet visited. Finally, we perform the retraction step on free as well as in-colliding configurations. Overall, our retraction-based RRT can explore or capture the connectivity of narrow passages quickly. We further analyze the behavior of our retraction-based RRT algorithm in Section 5 and demonstrate these benefits on many challenging benchmarks in Section 6.

V. Analysis

In this section, we analyze the behavior of our retraction-based RRT planner. We use the Voronoi diagram defined over the nodes of the RRT in the configuration space to analyze the performance of our enhanced RRT planner. Based on this analysis, we identify the planning scenarios with narrow passages where our retraction-based RRT planner can be quite effective.

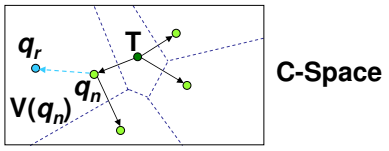


Fig. 4. **Analyzing RRT using Voronoi Diagrams:** *Given a randomly generated configuration $\mathbf{q}_r$, the step of finding its nearest node $\mathbf{q}_n$ in the tree T for extension is equivalent to locating the Voronoi region that contains $\mathbf{q}_r$ [3].*

A. Voronoi Diagrams

The behavior of RRT algorithms can be understood using Voronoi diagrams [2], [3], [27]. Specifically, the Voronoi diagram for a set of points S in a metric space is the partition of this space which associates a region $V(\mathbf{q})$ with each point $\mathbf{q}$ from S in such a way that all points in $V(\mathbf{q})$ are closer to $\mathbf{q}$ than to any other point in S. Given a tree built by an RRT algorithm, we consider the Voronoi diagram over the set of nodes of the tree (Fig. 4) in the configuration space associated with a distance metric δ.

Given a randomly generated configuration $\mathbf{q}_r$, the step of computing the nearest node in the RRT algorithm is equivalent to locating the Voronoi region that contains $\mathbf{q}_r$ (Fig. 4). Therefore, the probability of a node $\mathbf{q}$ in the tree being chosen for extension is proportional to the ratio ρ of the volume of its Voronoi region $V(\mathbf{q})$ to the volume of the sampling space [28] or the entire configuration space $\mathcal{C}$:

$$\rho(\mathbf{q}) = \frac{Volume(V(\mathbf{q}))}{Volume(\mathcal{C})}. \tag{2}$$

We refer to this ratio as *extension ratio* of a node. If a node has a higher value of extension ratio ρ, this node has a higher likelihood of being chosen for extension as compared to other nodes in the tree [3]. Therefore, RRT planners can bias the growth of the tree towards regions not yet visited [3].

B. Analysis of Retraction-based RRT

We analyze the behavior of our retraction-based RRT planner (Fig. 5). The tree in our planner is biased towards the contact space, and many nodes of the tree are either close to it or in the contact space (e.g. $\mathbf{q}_{n1}$ in Fig. 5-a). This is due to the retraction algorithm, which iteratively optimizes over the contact space. We classify the nodes in the tree, near or in the contact space, according to whether a node is:

- Type 1: far away from any narrow passage (e.g. $\mathbf{q}_{n0}$ in Fig. 5-(a)),
- Type 2: lying in a narrow passage, or
- Type 3: close to the entrance of a narrow passage (e.g. q_{n1} in Fig. 5-(a)).

Among these nodes, the nodes of Type 2 or Type 3 are important for planning as they are associated with narrow passages. Our retraction algorithm utilizes them in a manner such that many samples are generated in close proximity of Type 2 and Type 3 nodes or in the associated narrow passages. This is due to our retraction computation, and Fig. 5-(b) shows such an example. For a randomly generated sample $\mathbf{q}_{r_1}$, its nearest neighbor $\mathbf{q}_{n_1}$ with Type 3 is chosen

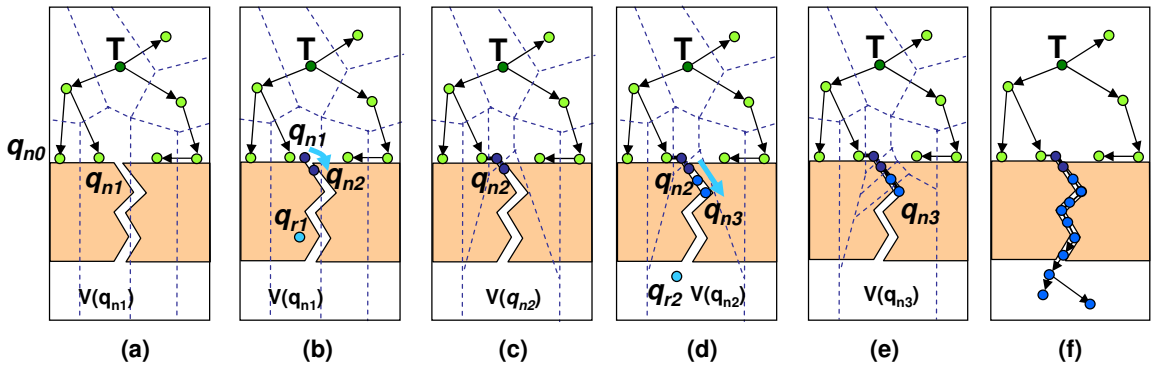


Fig. 5. **Analysis of Retraction-based RRT planner:** *(a) A tree - T needs to grow through a zigzag narrow passage. The Voronoi diagram is defined over the nodes of the tree in the configuration space. The Voronoi region of the node $\mathbf{q}_{n_1}$ is denoted as $V(\mathbf{q}_{n_1})$. (b) For a randomly generated in-colliding configuration $\mathbf{q}_{r_1}$, our RRT planner uses its nearest neighbor $\mathbf{q}_{n_1}$ as the initial guess and performs the optimization-based retraction. Since $\mathbf{q}_{n_1}$ is close to the entrance of the narrow passage, it is very likely that the optimization algorithm generates configurations, e.g. $\mathbf{q}_{n_2}$, in the narrow passage. (c) Though the node $\mathbf{q}_{n_2}$ in the tree T lies in the narrow passage, it has a high value of extension ratio ρ due to the large volume of its Voronoi region $V(\mathbf{q}_{n_2})$. Therefore, the node $\mathbf{q}_{n_2}$ has a high likelihood of being chosen for extension. (d) Given a randomly generated collision-free configuration $\mathbf{q}_{r_2}$, the node $\mathbf{q}_{n_2}$ is chosen for extension and more samples in the narrow passage are generated. (e) With a high value of the extension ratio, the node $\mathbf{q}_{n_3}$ in the narrow passage has a high likelihood of being chosen for extension. Therefore, more nodes in the narrow passage can be generated. (f) Finally, the tree grows through the narrow passage.*

for extension. Stating from $\mathbf{q}_{n_1}$, our retraction algorithm iteratively optimizes over the contact space and generates more samples towards or along narrow passages.

A key factor that governs the effectiveness of our retraction-based RRT planner is the probability with which the nodes of Type 2 and Type 3 are chosen for RRT extension. In general, the performance of sample-based approaches degrades in narrow passages since the ratios of the volumes of narrow passages to the volume of the sampling space are typically small. As a result, prior randomized sampling methods may not compute sufficient number of samples in narrow passages to find collision-free paths. However, there are many planning scenarios with narrow passages where the extension ratios, ρ, for the tree nodes of Type 2 and Type 3 are much larger as compared to the ratios of the volumes of their associated narrow passages to the volume of the sampling space. Fig. 5 illustrates such cases. Our retraction algorithm can generate more samples in the narrow passages and thereby improve the performance of the planner.

VI. Implementation and Results

In this section, we present experimental results of our retraction-based RRT planner, RRRT, on 3D rigid robots. We first address some implementation issues. Next, we highlight the performance of our planner on a set of benchmarks (Figs. 6, 7, 8 and 9). In each benchmark, a rigid robot needs to plan through some narrow passages in the 3D environment. All the timings reported here were taken on a Windows PC with 2.8GHz of CPU and 2GB of memory.

A. Implementation

We have implemented RRRT on 3D rigid robots. Our implementation consists of two parts: the implementation of the retraction step and the integration with an RRT planner. In the first part, we use the DISP distance metric defined in SE(3) and extend the generalized penetration depth computation algorithm in [5] to perform the retraction step. We set the maximum iteration in each retraction step as 5. We use PQP [29] for collision detection. We further extend this library to perform contact query by computing the closest features [30]. We implement a basic RRT planner. For simplicity, we perform the local planning by using a linear interpolation motion and checking for collisions on a finite number of intermediate configurations of the motion.

We integrate our retraction algorithm into the basic RRT planner. During each retraction step, a sequence of configurations, close to or in the contact space, are generated. Our planner RRRT then attempts to extend the tree to each configuration using the standard RRT extension scheme. In our implementation, we observe the difficulty of connecting two nearby samples when both of them are close to the contact space. Currently we use an enhanced local planning scheme - *vertex enhancement* that can generate additional samples around them [1]. To deal with this issue, other local planning schemes can also be employed [31].

B. Results

We test our retraction-based RRT planner on a set of benchmarks. In our experiment, we run every benchmark 10 times and compute the average running time. The timing is summarized in Table I. The geometric complexity of the benchmarks is highlighted in Table II. RRRT can handle general polygonal models, including polygon soup models.

In the notch benchmark (Fig. 6), there are three narrow passages since the width of the corridor within each notch-shaped model is 1 and it is slightly larger than the 'thickness' of the g-shaped robot, 0.95. The environment also possesses an interesting property. The widths of the two gaps formed by the three notch-shaped models are 0.9, resulting in two potentially false passages. Therefore, dilation-based planners may not work well on this benchmark. In our experiment, the basic RRT planner was unable to find a solution within $1,232.2$s. On the other hand, our planner can find a collision-

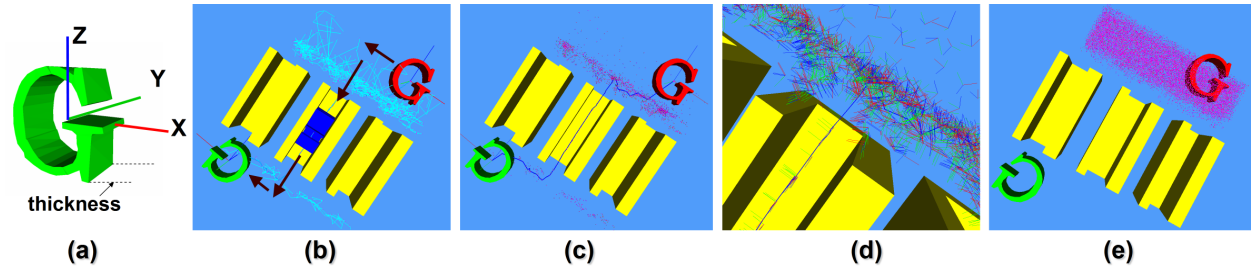


Fig. 6. **Notch Benchmark:** *(a) A G-shaped robot. (b) The environment is consisted of three notch-shaped obstacles, and the robot needs to move from one side to the other. Due to the narrow passages, the basic RRT planner was unable to find a solution after* $500,000$ *iterations within* $1,232.2$*s. On the other hand, RRRT can find a collision-free path within* 25.4*s. In (b), (c), the collision-free path, the tree and its nodes are highlighted after being projected from the 6D C-space into 3D Euclidean. Many nodes are biased towards the contact space as well as the narrow passage. (d) is a magnified version of the result. Here, a node in the tree is visualized by a 3D point to indicate the position of the robot, together with a 3-dimensional orthogonal frame to indicate the orientation. (e) shows the nodes generated by the basic RRT planner, where no node lies in the narrow passages.*

	Notch	Torus	Flange	Alpha Puzzle
RRRT: t_{all} (s)	25.4	44.9	25.0	4,130.5
RRRT: nodes	1,401	1,471	119	103,121
Basic RRT: t_{all} (s)	$>$ 1,232.2$\star$	4,920.9	680.1	$>$ 40,747.1 $\star$
Basic RRT: nodes	$>$ 105,987$\star$	43,512	95	$>$ 28,219 $\star$

TABLE I

Performance: THE TABLE COMPARES THE PERFORMANCE OF OUR PLANNER - RRRT WITH THE BASIC RRT PLANNER ON DIFFERENT BENCHMARKS. THE TABLE INCLUDES THE PLANNING TIME t_{all} AND THE NUMBER OF NODES IN THE RESULTING TREE. $\star$: THE BASIC RRT PLANNER CANNOT FIND A PATH WITHIN A LARGE MOUNT OF TIME.

	Notch	Torus	Flange	Alpha Puzzle
# Tri of robot	28	20	3,525	1,044
# Tri of obstacles	756	5,760	5,306	1,044
# of obstacles	3	2	1	1

TABLE II

Model Complexity: THE TABLE SUMMARIZES THE GEOMETRIC COMPLEXITY OF EACH BENCHMARK.

free path within 25.4s. Figs. 7,8 show two additional benchmarks where the models are more complex. Our planner can compute a collision-free path through a narrow passage within 44.9s and 25.0s, respectively. Fig. 9 shows Alpha Puzzle - a well-known challenging benchmark for motion planning algorithms [12]. Our planner can solve it within $4,130.5$s, generating $103,121$ nodes.

Table III highlights two ways to break down the running time for RRRT. One way is to measure the $t_{retraction}$, the time on the retraction step and $t_{linking}$, the time on connecting the samples. The other way is to measure t_{cd} the time on collision detection and $t_{contact}$, the time on contact query. Overall, the function for collision detection takes around 70% to 80% of the total time.

C. Comparison with RRT planners

We compare the performance of our retraction-based RRT planner with other RRT planners. We first compare with the basic RRT planner, which uses the *standard RRT extension*. Table I shows for all four benchmarks, RRRT is much more efficient than the basic RRT planner. We also test both planners on different versions of the notch benchmark, i.e. scaling the G-shaped robot from 0.8 to 0.95. Tab. IV shows that RRRT drastically improves the performance for

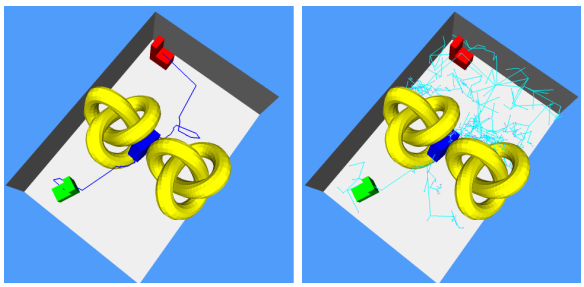


Fig. 7. **Torus Benchmark:** *Left: A L-shaped robot needs to move from one side (red) in the environment, consisting of two torus knot shaped obstacles, to the other (green). Our planner can compute a collision-free path within* 44.9*s. Right: the computed tree is highlighted.*

each version of the problem. Figs. 6-(c),(e) compare the distribution of the nodes generated by RRRT and the basic RRT planners for the 0.95 version. The RRRT planner generates more samples in the contact space and narrow passages, while the basic RRT cannot generate samples in narrow passages.

There are variants of RRT-based planners such as [15], [32] to improve the performance on narrow passages. We quantitatively compare our planner with the RRT-based planner presented in [15] by using the flange benchmark (Fig. 8). Our planner takes 25.0s for this task and is much more efficient than the planner in [15], which takes 227.1s on a similar PC. Compared with another RRT planner in [32], one difference is that we perform the retraction on both in-colliding as well as collision-free configurations, while their method can only bias the growth of the tree using the collision-free configurations. Finally, another RRT variant [27] also takes into account C-obstacle into the RRT bias as ours. However, their work mainly characterizes the issue when the sampling domain is not well adapted to the problem, while our method focuses on improving the performance in narrow passages.

We have not performed an extensive comparison with other retraction-based methods [20], [21]. However, our planner has many distinct features as compared to them. The small-step retraction-based methods [20], [21] identify the in-colliding configurations near the free space by shrinking the models of the robot and the obstacles. These methods are only applicable to closed models. Moreover, it is difficult

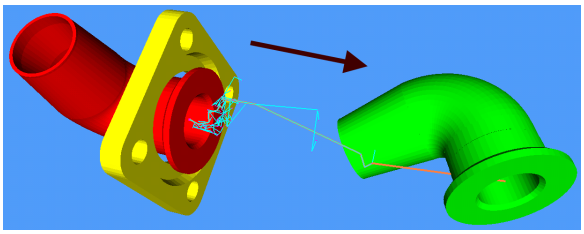


Fig. 8. **Flange Benchmark:** *The CAD model - 'elbow' needs to slide out of the hole of the CAD model - 'flange'. Our planner takes* 25.0*s for this task, while a variant of RRT presented in [15] takes* 227.1*s on a similar PC.*

	Notch	Torus	Flange	Alpha Puzzle
t_{all}(s)	25.4	44.9	25.0	4,130.5
$t_{retraction}$(s)	11.0	22.1	16.1	2,655.8
$\#_{retraction}$	625	1,203	232	61,650
t_{per_retra}(ms)	17.600	18.371	69.397	43.079
$t_{retraction}/t_{all}$	43.3%	49.2%	64.4%	64.3%
$t_{linking}$ (s)	14.2	18.9	6.6	1,343.1
$\#_{linking}$	1,587	2,920	303	164,242
$t_{per\_linking}$ (ms)	8.948	6.473	21.782	8.178
$t_{linking}/t_{all}$	55.9%	42.1%	26.4%	32.5%
t_{cd}(s)	18.5	32.1	19.3	3,273
$\#_{cd}$	91,958	133,580	20,794	14,453,046
t_{per_cd}(ms)	0.201	0.240	0.928	0.226
t_{cd}/t_{all}	72.8%	71.5%	77.0%	79.2%
$t_{contact}$(s)	0.9	2.0	2.9	479.2
$\#_{contact}$	1,548	2,831	252	132,837
$t_{per\_contact}$(ms)	0.581	0.706	11.508	3.607
$t_{contact}/t_{all}$	3.4%	4.4%	11.5%	11.6%

TABLE III

Breakdown of Running Time: THE TABLE SUMMARIZES TWO WAYS TO BREAK DOWN THE RUNNING TIME FOR MAIN FUNCTIONS IN RRRT. ONE WAY IS TO MEASURE $t_{retraction}$, THE TIME ON THE RETRACTION STEP AND $t_{linking}$, THE TIME ON CONNECTING THE GENERATED SAMPLES. ANOTHER WAY IS MEASURE t_{cd}, THE TIME ON COLLISION DETECTION AND $t_{contact}$, THE TIME ON CONTACT QUERY. $\#_{retraction}$ AND t_{per_retra} DENOTE THE NUMBER OF RETRACTION STEPS AND THE AVERAGE TIME OF EACH RETRACTION STEP, RESPECTIVELY.

to perform the shrinking step on complex models, and the topology of the dilated free space may be different. On the other hand, our algorithm is directly applied to general polygonal models. Furthermore, based on the shrinking of geometric models, these methods implicitly use the formulation of *growth distance* for quantifying the amount of interpenetration among the models [33]. This formulation is not as rigorous as our underlying formulation of generalized penetration depth computation, which is based on a proper distance metric that meaningfully combines the translational and rotational motion of the robot [5]. Fig. 10 shows such an example.

VII. LIMITATIONS

There are several limitations of our approach. Our optimization-based retraction searches over the contact space and computes a local minima. As a result, it can generate many configurations that lie in the contact space but not in the narrow passages. This can affect the overall performance of the planner. Furthermore, the optimization-based

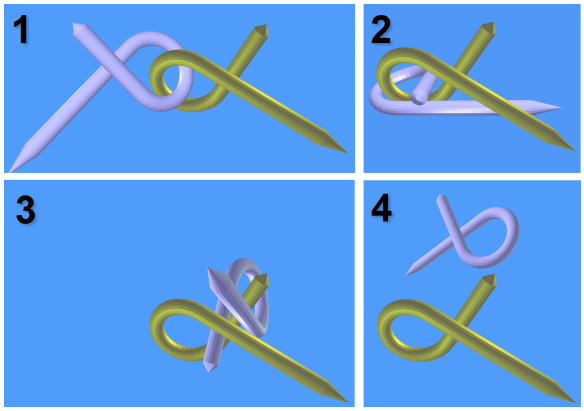


Fig. 9. **Alpha Puzzle Benchmark:** *RRRT can solve this challenging problem - to separate the two interlocked alpha-shaped models within* $4,130.5s$.

retraction step has additional overhead. If the configuration has no narrow passages, our enhanced planner may take longer time as compared to the basic planner. However, in the notch benchmark where the robot is scaled down to 0.8 so that the narrow passages become wider, our planner still performs as well as the basic RRT. Finally, our algorithm is restricted to rigid models, and performing the retraction step on articulated models can be more expensive.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we present an optimization-based retraction algorithm to improve the performance of RRT planners by retracting the samples so that they can be more likely to be connected to the tree. The resulting tree can grow closely towards every randomly generated sample, including in-colliding as well as free configurations. We analyze the behavior of our planner using Voronoi diagrams of the configurations in the tree and highlight the scenarios where our planner can handle narrow passages well. We have implemented this algorithm and applied it for rigid robots in challenging planning scenarios. Our experimental results show that our algorithm generates more samples near the contact space or in the narrow passages than prior RRT planners and is able to explore more difficult regions in the configuration space. In practice, we observe significant speedups over prior RRT planners.

There are many avenues for future work. We are interested in applying our retraction algorithm to PRM planners and

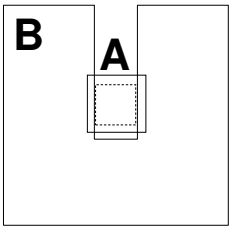


Fig. 10. **Growth Distance vs. Penetration Depth:** *When the square* A *is slightly shrunk, it will be disjoint from the notch* B*. Therefore, the formulation of growth distance treats this case as shallow penetration. However, the formulation of generalized penetration depth characterizes it as deep penetration. The latter characterization is more accurate in the context of path planning.*

Robot Scale	0.80	0.85	0.90	0.91	0.92	0.93	0.94	0.95
Basic RRT: t_{all}(s)	11.6	31.2	120.8	143.4	184.7	292.7	306.3	$>$ 1,232.2 $\star$
RRRT: t_{all}(s)	6.6	7.0	9.4	14.0	8.1	26.1	23.6	25.4
Basic RRT: nodes	4,226	8,843	22,755	23,085	28,417	43,200	42,530	$>$ 105,987 $\star$
RRRT: nodes	398	467	524	786	510	1,519	1,460	1,401

TABLE IV

Comparison: THE TABLE COMPARES THE PERFORMANCE OF RETRACTION-BASED RRT WITH THE BASIC RRT PLANNER ON THE NOTCH BENCHMARK. THE ROBOT IS SCALED FROM 0.8 TO 0.95. OUR PLANNER SIGNIFICANTLY IMPROVES THE PERFORMANCE ON EACH VERSION OF THE PROBLEM. $\star$ DENOTES THE MOST DIFFICULT VERSION - 0.95. THE BASIC RRT PLANNER CANNOT SOLVE IT AFTER RUNNING 500, 000 ITERATIONS WITHIN 1.232.2S, WHILE OUR RRRT CAN COMPUTE A PATH WITHIN 25.4S.

performing the retraction computation for articulated robots. Moreover, it would be interesting to apply our algorithm to CAD part disassembly [30]. Finally, we would like to extend our RRT planner for motion planning under differential constraints, e.g. non-holonomic motion planning.

Acknowledgements: This research was supported in part by ARO Contracts DAAD19-02-1-0390 and W911NF-04-1-0088, NSF awards 0400134, 0429583 and 0404088, DARPA/RDECOM Contract N61339-04-C-0043 and Intel.

REFERENCES

[1] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, pp. 12(4):566–580, 1996.

[2] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. IEEE International Conference on Robotics and Automation*, 2000.

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

[4] D. Hsu, J. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *Int. J. Robotics Research*, vol. 25, no. 7, pp. 627–643, 2006.

[5] L. Zhang, Y. Kim, and D. Manocha, "A fast and practical algorithm for generalized penetration depth computation," in *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[6] H. Kurniawati and D. Hsu, "Workspace-based connectivity oracle: An adaptive sampling strategy for prm planning," in *Proc. of 7th International Workshop on the Algorithmic Foundations of Robotics*, 2006.

[7] J. van den Berg and M. Overmars, "Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners," *The International Journal of Robotics Research*, vol. 24, no. 12, pp. 1055–1071, Jan 2005.

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

[9] T. Simeon, J. P. Laumond, and C. Nissoux, "Visibility based probabilistic roadmaps for motion planning," *Advanced Robotics Journal*, vol. 14, no. 6, 2000.

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

[11] M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. M. Amato, "C-space subdivision and integration in feature-sensitive motion planning," in *Proc. IEEE International Conference on Robotics and Automation*, 2005.

[12] N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo, "Obprm: An obstacle-based prm for 3d workspaces," *Proceedings of WAFR*, pp. 197–204, 1998.

[13] D. Hsu, L. Kavraki, J. Latombe, R. Motwani, and S. Sorkin, "On finding narrow passages with probabilistic roadmap planners," in *Robotics: The Algorithmic Perspective—Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 1998, pp. 141–154.

[14] S. Redon and M. Lin, "A fast method for local penetration depth computation," *Journal of Graphics Tools*, vol. 11, no. 2, pp. 37–50, 2006.

[15] S. Rodriguez, X. Tang, J. Lien, and N. Amato, "An obstacle-based rapidly-exploring random tree," in *Proc. IEEE International Conference on Robotics and Automation*, 2006, pp. 895–900.

[16] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "Motion planning for a rigid body using random networks on the medial axis of the free space," in *Symposium on Computational Geometry*, 1999, pp. 173–180.

[17] M. Foskey, M. Garber, M. Lin, and D. Manocha, "A voronoi-based hybrid planner," *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2001.

[18] K. Hoff, T. Culver, J. Keyser, M. Lin, and D. Manocha, "Interactive motion planning using hardware accelerated computation of generalized voronoi diagrams," *Proceedings of IEEE Conference of Robotics and Automation*, 2000.

[19] S. Redon and M. Lin, "Practical local planning in the contact space," *ICRA*, 2005.

[20] M. Saha, J. Latombe, Y. Chang, Lin, and F. Prinz, "Finding narrow passages with probabilistic roadmaps: the small step retraction method," *Intelligent Robots and Systems*, vol. 19, no. 3, pp. 301–319, Dec 2005.

[21] H.-L. Cheng, D. Hsu, J.-C. Latombe, and G. Sánchez-Ante, "Multi-level free-space dilation for sampling narrow passages in PRM planning," in *Proc. IEEE Int. Conf. on Robotics & Automation*, 2006, pp. 1255–1260.

[22] E. Ferr and J.-P. Laumond, "An iterative diffusion algorithm for part disassembly," in *Proc. IEEE International Conference on Robotics and Automation*, April 2004, pp. 3149–3154.

[23] B. R. Donald, "A search algorithm for motion planning with six degrees of freedom," *Artif. Intell.*, vol. 31, no. 3, pp. 295–353, 1987.

[24] H. Hirukawa, Y. Papegay, and H. Tsukune, "A motion planning algorithm of polyhedra in contact for mechanicalassembly," in *20th International Conference on Industrial Electronics, Control and Instrumentation*, vol. 2, 1994, pp. 924–929.

[25] X. Ji and J. Xiao, "Planning motion compliant to complex contact states," *International Journal of Robotics Research*, vol. 20, no. 6, pp. 446–465, 2001.

[26] J. Xiao and X. Ji, "On automatic generation of high-level contact state space," *International Journal of Robotics Research*, vol. 20, no. 7, pp. 584–606, July 2001.

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

[28] C. Pisula, K. Hoff, M. C. Lin, and D. Manocha, "Randomized path planning for a rigid body based on hardware accelerated voronoi sampling," *Proc. of International Workshop on Algorithmic Foundation of Robotics*, 2000.

[29] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes," Department of Computer Science, University of North Carolina, Tech. Rep. TR99-018, 1999.

[30] L. Zhang, X. Huang, Y. Kim, and D. Manocha, "D-plan: Efficient collision-free path computation for part removal and disassembly," Department of Computer Science, University of North Carolina at Chapel Hill, Tech. Rep. 08-002, 2008.

[31] L. Zhang and D. Manocha, "Motion interpolation with distance constraints," Department of Computer Science, University of North Carolina at Chapel Hill, Tech. Rep. 08-001, 2008.

[32] M. Strandberg, "Augmenting RRT-planners with local trees," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 4, April 2004, pp. 3258–3262.

[33] C. J. Ong and E. Huang, "An incremental version of growth distance," in *Proc. of Int'l Conf. on Robotics and Automation*, 1998.