SR-RRT: Selective Retraction-based RRT Planner

Junghwan Lee[1] OSung Kwon[2] Liangjun Zhang[3] Sung-eui Yoon[4]

***Abstract*— We present a novel retraction-based planner, selective retraction-based RRT, for efficiently handling a wide variety of environments that have different characteristics. We first present a *bridge line-test* that can identify regions around narrow passages, and then perform an optimization-based retraction operation selectively only at those regions. We also propose a *non-colliding line-test*, a dual operator to the bridge line-test, as a culling method to avoid generating samples near wide-open free spaces and thus to generate more samples around narrow passages. These two tests are performed with a small computational overhead and are integrated with a retraction-based RRT. In order to demonstrate benefits of our method, we have tested our method with different benchmarks that have varying amounts of narrow passages. Our method achieves up to 21 times and 3.5 times performance improvements over a basic RRT and an optimization-based retraction RRT, respectively. Furthermore, our method consistently improves the performances of other tested methods across all the tested benchmarks that have or do not have narrow passages.**

I. INTRODUCTION

Many sampling-based motion planning algorithms have been designed and successfully used to compute collision-free paths for a variety of environments. Two of the most successful algorithms include Probabilistic Roadmap Method (PRM) [1] and Rapidly-exploring Random Tree (RRT) [2]. At a high level, as we randomly generate more samples, these techniques provide collision-free paths by capturing a larger portion of the connectivity of the free space.

One of the most challenging cases for sampling-based techniques is to efficiently handle narrow passages in the free space. In many motion planning applications, such as part disassembly simulation, tolerance verification, and protein folding [3], [4], a robot often needs to pass through narrow passages and the performance of sampling-based planning algorithms can degrade significantly. Many approaches have been proposed in different directions such as utilizing the workspace geometric information [5], filtering (or adaptive sampling) techniques towards more important regions [6], [7], [8], retracting samples, etc.

Recently, retraction-based techniques [9], [10], [11], [12], [13] have been actively studied, since they can pose samples near the boundary of the C-Obstacle, efficiently leading to explore important regions including narrow passages. However, these techniques have the computation overhead of retracting sampling near the boundary of the C-Obstacle and thus can run slower even to a basic RRT method, if the free space does not have such difficult regions [9].

[1]Junghwan Lee and [2]Osung Kwon are at Dept. of CS, KAIST, Daejeon, South Korea

[3]Liangjun Zhang is at Dept. of CS, Stanford University, USA

[4]Sung-eui Yoon is at Dept. of CS and Div. of Web Sci. and Tech., KAIST

Fig. 1. **Living Room Benchmark:** This figure shows three image shots of getting the sofa out through a door, which creates narrow passages in 6 dof configuration space. Our method achieves ten and two times improvements over a basic RRT and an optimization-based retraction method respectively.

Main results: In this paper we propose a novel retraction-based planner, *Selective Retraction-based RRT*, for a wide variety of environments that have or do not have narrow passages. Our method identifies the nearest neighbor node given a randomly generated sample, then computes a first in-contact configuration at the boundary of the C-Obstacle along the straight line from the node to the random sample. Instead of performing optimization-based retraction operations exhaustively, we selectively perform retraction operations, only if samples are deemed to be around narrow passages. To decide whether a node is on such a narrow passage, we propose a *bridge line-test*, an inexpensive filtering operation, which checks whether narrow passages exist along a line segment. In order to achieve a high accuracy even in high dimensional configuration spaces, we perform principal component analysis (PCA) and generate the line with a higher probability to cross narrow passages. Also, in order to generate more random samples near narrow passages, we present a *non-colliding line-test* that detects wide-open free spaces and cull samples near such spaces (Sec. IV).

We have implemented our SR-RRT method integrated with these two line-tests. In order to demonstrate benefits of our method, we have tested SR-RRT with various types of benchmarks that have different characteristics (Sec. V). Our method achieves on average 6.7 times and 2 times improvement over a basic RRT [2] and an optimization-based retraction method [9], respectively, because of the low computational overheads of our method and its accuracy of identifying narrow passages and wide-open free spaces. Moreover, while the basic RRT and the optimization-based retraction method show lower performance than the other tested methods in some benchmarks, our method consistently improves the performances across all the tested benchmarks that have or do not have narrow passages. As a result, we can conclude that our method is more robust and general than other tested methods.

II. RELATED WORK

In this section we discuss prior work on sampling-based motion planning, especially designed to efficiently handle

narrow passages.

A. Sampling-based Motion Planning

The sampling-based algorithms have been successfully used to solve various motion planning problems in practice. Among them Probabilistic Roadmap Method (PRM) [1] and Rapidly-exploring Random Tree (RRT) [2] are the most widely used approaches [14]. These techniques have been extensively studied and an excellent survey is available [15].

Our method is built upon the RRT methods, since they have been extended to solve a wide variety of practical problems in robotics [16], [17], [18], [19]. RRT methods have been improved in many different directions by considering workspace information [5], [20], biasing sampling strategies [21], [22], decomposing environments and focusing sampling on critical paths [23], etc.

B. Narrow Passages

One of the most difficult challenges for sampling-based methods is to efficiently handle narrow passages in the free space of a robot. Many strategies have been proposed to improve the performance of both PRM and RRT for narrow passages [15]. At a high level, they include the use of the workspace geometric information to guide the sampling [5], [24], filtering(or adaptive sampling) techniques towards important regions including narrow passages [6], [7], [8], dynamic sampling distributions [25], utilizing free space information [26], and retraction-based approaches.

Filtering techniques: Filtering techniques [6], [7], [8] aim to generate lots of samples at the robot's free space and filter out some of them that are not located near difficult regions such as narrow passages, leading to adaptive sampling. Boor et al. [6] proposed the Gaussian sampling strategy that distributes samples near the boundary of the free space. The bridge test proposed by Hsu et al. [8] uses three sampled configurations to boost the sampling density inside narrow passages. Shkolnik and Tedrake [27] proposed the Ball Tree algorithm that approximates the reachable free space in a similar way to our non-colliding line-test.

Retraction-based approach: The main idea of the retraction-based approach is to retract in-collision samples towards more useful regions such as the boundary of C-Obstacle or the medial axis of the free space. Some of retraction-based algorithms utilize the contact information for retraction [10], [12], [9] or dilate the free space [11], [13]. These techniques can efficiently handle narrow passages, but can run very slowly, if the free space does not have such difficult regions because of their computational overheads.

At a high level, our approach integrates these two different approaches, filtering and retraction-based techniques, in order to robustly handle various environments that do or do not have narrow passages. Hsu et al. [28] also considered to use different sampling methods and adaptively used them for a higher performance. On the other hand, this paper explores another direction of integrating different techniques; our method uses a cheap filtering method as a *culling technique* to decide whether it is desirable to perform an expensive retraction method or not.

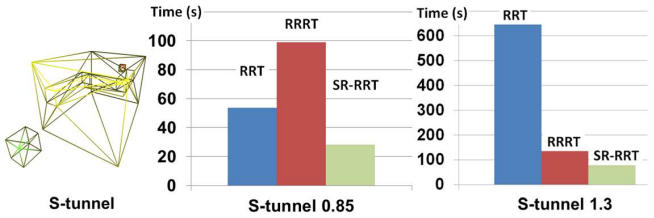


Fig. 2. **S-tunnel Benchmark:** The leftmost figure shows the S-tunnel benchmark. The right two figures show the average performance of two variations of the S-tunnel benchmark. 0.85 and 1.3 indicate the scaling factor of the cubic robot. 0.85 does not create any narrow passages, while 1.3 creates them.

III. BACKGROUNDS

In this section we give a brief review on the basic RRT and its variant based on an optimization-based retraction, which is designed for efficiently handling narrow passages. Then we discuss their issues that arise when we apply them to various environments, some portions of which contain narrow passages, but other portions of which do not have such difficult regions.

A. Reviews of RRT and Retraction

A basic RRT algorithm starts with a single or multiple random trees [2]. The basic RRT method generates a sample, q_r, in the configuration space and finds its nearest neighbor node, q_n, among nodes of random trees. Then we attempt to connect q_n with q_r. If q_r is in collision or there are any obstacles in between q_r and q_n, we compute a first in-contact configuration, q_c, that touches the boundary of C-Obstacle, along a straight line from q_r and q_n [14, pp. 189–192]. Then q_c is added to the tree and is connected with q_n (Fig. 4-(a)).

It has been known that the basic RRT explores the free space with a bias related to the Voronoi diagram [2]. Especially, a probability that a node of a random tree is chosen as the nearest neighbor node is proportional to the volume of the Voronoi region associated with the node.

This basic RRT or its variants, however, can take a prohibitively long planning time, when the free space of a robot contains narrow passages. This is because the regions associated with narrow passages are significantly smaller than other regions. As a result, the probability of exploring and getting out the narrow passage region is quite low.

In order to address the problem of the basic RRT, an optimization-based retraction technique [9] was proposed recently. Its main idea is to iteratively retract a sample near the boundary of the free space, i.e. contact space, based on a local contact analysis, while minimizing a particular objective function (e.g., a distance metric); for example, $q_{c'}$ is a newly retracted in-contact configuration from q_c in the right image of Fig. 4-(a). Even though this optimization technique behaves in a greedy manner, this technique has been demonstrated to work well in environments that have narrow passages. For example, this optimization-based retraction technique shows two times higher performance than the basic RRT in the wiper 1.0 benchmark (Fig. 3), which has narrow passages.

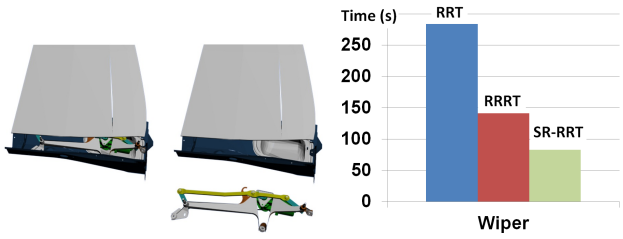


Fig. 3. **Wiper 1.0 Benchmark:** The left two figures show two animation sequences of getting a wiper out of the windscreen model. Since there are not much rooms between two models, this benchmark has narrow passages. The rightmost graph shows the average planning time of different methods performed in 100 times.

B. Issues with Optimization-based Retraction

The optimization-based retraction technique works quite well with scenes that have narrow passages. This result is achieved by performing extra operations to the basic RRT, such as local contact analysis, additional sampling, etc. These operations have relatively higher computational overheads, compared to other operations of the basic RRT method [9]. Also, while this technique explores and gets out of narrow passages efficiently by generating samples on the contact space, it covers all the contact spaces equally well. In other words, this technique tends to generate many samples near the contact space, even though the contact space is not on narrow passages.

As a result, if an environment does not have narrow passages or have many obstacles that create the contact space without generating narrow passages, the optimization-based retraction technique can show even lower performance than the basic RRT, because of both the computational overhead and excessive sampling on the contact space. For example, it shows 84% lower performance than the basic RRT in the S-tunnel benchmark 0.85 (Fig. 2) that does not have narrow passages.

IV. SELECTIVE RETRACTION-BASED RRT

Our technique is based on the optimization-based retraction RRT method [9]. To efficiently handle various types of environments that have or do not have narrow passages, it is critical to identify regions that are likely to be narrow passages and to selectively perform the expensive retraction operation only on those regions.

In order to efficiently identify such narrow passages within our RRT-based planner, we present a *bridge line-test*, which is inspired by the bridge test [8], originally proposed for probabilistic roadmap techniques. Our bridge line-test uses a line passing through an in-contact configuration to test whether it is likely to have narrow passage around the configuration. We also propose a *non-colliding line-test*, a dual operator to the bridge line-test, to generate more samples near narrow passages.

A. Selective Retraction-based Extension

We first explain our extension method of SR-RRT, whose pseudo code is at Algorithm 1. Once a random sample q_r and its nearest neighbor node q_n are computed as mentioned in Sec. III-A, then we perform our extension method, **SRExtend** (q_n, q_r), shown in Algorithm 1. The first step of our extension method is exactly the same to the extension method of the basic RRT explained in Sec. III-A. Note that at this step, we may create an in-contact configuration q_c.

Algorithm 1 SRExtend(q_n, q_r)

```
if HaveCollisionFreePath (q_n q_r) then
    return RRTExtend(q_n, q_r)
end if
q_c ← the first in-contact configuration from q_n to q_r
if BridgeLineTest (q_c) then
    OptimizedRetraction (q_c)
end if
return q_c
```

As the second step of our extension method, we perform our bridge line-test to decide whether we need to perform the retraction operation. If the bridge line-test indicates that there is a narrow passage around the in-contact configuration q_c, we perform the retraction operation and generate another in-contact configuration $q_{c'}$ in a way that we can reduce the distance between q_r and the newly generated in-contact configuration $q_{c'}$ [9]. This operation is represented as *OptimizedRetraction* $(\cdot)$ in the pseudo code of our extension method (Algorithm 1).

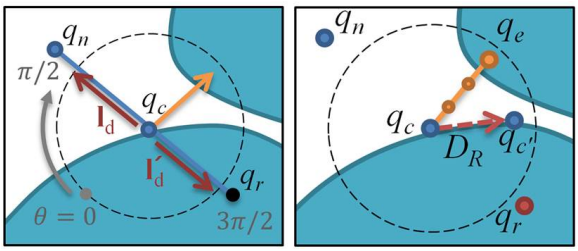
B. Bridge Line-Test

The bridge line-test determines whether a narrow passage exists around an in-contact configuration q_c. Its pseudo code is shown in Algorithm 2. To perform the bridge line-test, we first generate a line whose one end is located at q_c. The line can have an arbitrary line direction. Instead of generating the line direction in a uniform manner, we take into account available local information around q_c to make the line direction to cross narrow passages with higher probability.

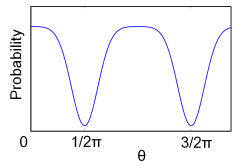
Note that it has been identified that a collision-free path exists from q_c to q_n (see the left image of Fig. 4-(a)). As a result, we can assume that there are no narrow passages along that particular direction, $\vec{l}_d$, which is $q_n - q_c$. On the contrary, a line segment between q_c and q_r is in C-Obstacle. As a result, we can also assume that there are no narrow passage along a line direction, $\vec{l}'_d = (q_r - q_c)$, from q_c. This local information leads us to generate an arbitrary line direction with a higher probability in these intervals defined between these two line directions, to make the randomly generated line to cross potentially-existing narrow passages.

In particular, we use a line generating function, $p_l(\cdot)$, a mixture of two reflected Gaussians, each of which has near zero probability at these two line directions $\vec{l}_d$ and $\vec{l}'_d$, while they peak at a plane whose normal is either $\vec{l}_d$ or $\vec{l}'_d$; Fig. 4-(b) shows the 2D example of this distribution function.

Once we generate a line direction starting from q_c, then we compute the other end point, q_e of the line by computing a random distance, d, according to a Gaussian function, $p_d(\cdot)$. We set the mean of the Gaussian function to be the average



(a) Procedure of the bridge line-test



(b) PDF for the line direction

Fig. 4. The upper two figures show the procedure of generating and performing a bridge line-test, while the bottom figure shows the probability distribution function (PDF) for the line direction parameterized with θ.

Algorithm 2 BridgeLineTest(q_c)

Construct a random line $\overline{q_c q_e}$ based on $p_l(\cdot)$, $p_d(\cdot)$, and PCA on the local free space of q_c
if (! HaveCollisionFreePath ($\overline{q_c q_e}$)) **then**
  return true // identified as a narrow passage.
end if
return false

distance between successive in-contact configurations computed by the retraction operation (e.g., D_R in Fig. 4-(a)). The main reason of choosing the mean in such a way is that since we can go deeper on average in the amount of D_R in a narrow passage with a retraction operation, we aim to identify narrow passages with that amount of width. We set the standard deviation of the Gaussian function to be small, but to have a meaningful probability (e.g., $D_R/2$) to identify very small narrow passages; we make sure that $p_d(0)$ has a non-zero probability to detect narrow passages with infinitely small width, when we generate infinite number of samples.

After computing a line whose two end points are q_c and q_e, i.e. $\overline{q_c q_e}$, we check whether there are any collisions in the line. For detecting collisions on the line, we can use either continuous or discrete collision detection methods. For discrete collision detection methods, we check a fixed number of intermediate configurations on the line. If there are any collisions at configurations other than the in-contact configuration q_c, we treat the region around these two configurations of q_c and q_e as they are in a narrow passage, and thus perform the retraction operation.

1) Re-testing: Our bridge line-test can fail with a low probability to identify a region to be in a narrow passage, even though the region is in a narrow passage. This failure is mainly because our bridge line-test considers only one dimensional line in multiple dimensional configuration spaces to check whether a node is in a narrow passage. Instead of performing the bridge line-test only one time for each node, we perform the bridge line-test, whenever an in-contact

Algorithm 3 Re-test(q_n)

if q_n is in-contact configuration and did not pass earlier bridge line-tests **then**
  if BridgeLineTest (q_n) **then**
    OptimizedRetraction (q_n)
  end if
end if

configuration that was not identified as narrow passages with earlier bridge line-tests is chosen as a nearest neighbor node of a random sample. This re-testing is defined as **Re-test(q_n)** shown in Algorithm 3.

Note that it is quite reasonable to re-test the node with the bridge line-test, since the node selected as a nearest node to others many times implies that the tree expansion may be stuck there by potentially-existing narrow passages around the node. Moreover, by allowing multiple re-testing, the bridge line-test can correctly identify the existence of narrow passages in a probabilistic manner with many random samples.

C. High-Dimensional Configuration Spaces

Our bridge line-test is very efficient, since the line-test considers whether there are collisions between the line of robot configurations and the environment. However, its accuracy degrades as the dimension of the configuration spaces goes higher because of its one dimensional nature of checking collisions. In order to ameliorate this problem, we consider how local free spaces grow, and generate random lines of bridge line-tests more frequently in dimensions that may contain narrow passages.

Inspired by a recent dimension reduction technique developed for random motion planning approaches [26], we perform the principal component analysis (PCA) to see how local free space is distributed, and treat a region to be in a narrow passage when the local free space is not uniformly distributed in the configuration space. For example, if the free space is located in a narrow passage, we can assume that the free space is not uniformly distributed, but is severely constrained and thus has an elongated shape (see the left image of Fig. 5).

The PCA is a well-known statistical procedure that computes a covariance matrix out of a set of points and provides eigenvectors and their corresponding eigenvalues of the matrix [29]. The PCA is often used for dimensionality reduction, and the computed eigenvectors are treated as principal components of input points.

We apply the PCA in a similar manner as used in the PCA-RRT [26]. When we need to generate a random line from q_c, we first collect k nearest neighbors from q_c by a breadth-first search, and then perform the PCA on those nearest neighbors. We assume that the principal component with the maximum eigenvalue, i.e. variance, aligns with the longest axis of the elongated-shaped narrow passage. Our goal of choosing the random line direction is to increase a probability that the generated direction crosses the longest axis of the narrow passage.

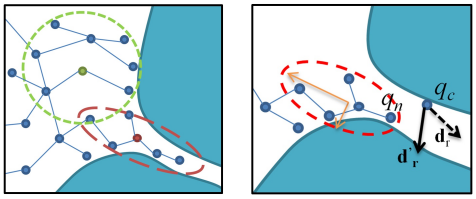


Fig. 5. The left figure shows shapes of local free spaces (green and red ones) computed from two nodes. The red one located in a narrow passage has elongated shape, while the green one located in wide-open free space is uniformly distributed in the space. The right figure shows the transformed line direction $\vec{d}'_r$ from an initially generated direction $\vec{d}_r$. Note that orange vectors are eigenvectors scaled by the corresponding eigenvalues computed from the local region of q_n.

To meet our goal, we transform $\vec{d}_r$, computed in the line generating PDF $p_l(\cdot)$ in Sec. IV-B, into a new line direction $\vec{d}'_r$ based on the following equation:

$$\vec{d}'_r = \sum_{i=1}^{n} (\frac{1}{\lambda_i} \vec{d}_r \cdot U_i) U_i,$$

where U_i is i-th eigenvector and λ_i is its corresponding eigenvalue. This equation transforms $\vec{d}_r$ to follow principal components more that have lower variances, leading to crossing a potentially-existing narrow passage, since $1/\lambda_i$, a transforming weight, becomes bigger as we have lower variances.

The accuracy of the PCA-based technique for transforming the line direction depends on how well our assumption is valid given a local configuration. Moreover, this technique makes a bias for generating line directions for our bridge line-test. In order to mitigate the negative effects of our PCA-based technique, we adopt the transformed line direction $\vec{d}'_r$ with a probability $p_l(\vec{d}'_r)$, the line generating PDF. If $\vec{d}'_r$ is rejected, we generate a line direction according to $p_l(\cdot)$ as mentioned in Sec. IV-B. As a result, the PCA-based bias has $(1-p_l(\vec{d}'_r))p_l(\vec{d}'_r)$ higher probability over our prior line generating function $p_l(\cdot)$, given the PCA-based computed line direction $\vec{d}'_r$. Note that we use our bridge line-test combined with the PCA technique to efficiently identify narrow passages before performing our retraction method, while the PCA-RRT [26] used the PCA to control random sampling.

D. Non-Colliding Line-Test

We can generate more samples near narrow passages by using both the bridge line-test and the optimization-based retraction operator. However, these retraction operations can be performed, once a random sample is located closely to such regions. In order to increase the probability to generate samples near such regions in an efficient manner, we propose a sampling bias technique using the *non-colliding line-test*, which is a dual operator to our bridge line-test.

Our main idea is that once we identify wide-open free spaces in the configuration space, it is more desirable to discard samples generated within such free spaces and to generate more samples outside those areas and potentially towards narrow passages.

It is, unfortunately, challenging to exactly define wide-open free spaces in high-dimensional configuration spaces [30]. Instead of constructing them deterministically, we define them in a probabilistic manner. At a high level, we generate a line segment from a node (similar probabilistic manner used in the bridge line-test) and check whether the line is a collision-free path. If so, we treat the region around the node as a wide-open free space.

Algorithm 4 NonCollidingLineTest(q_n)

```
Compute a hypersphere with the center of q_n and radius of d_NN
Generate a random point q_e within the hypersphere
if HaveCollisionFreePath (q_e q_n) then
    return true
end if
return false
```

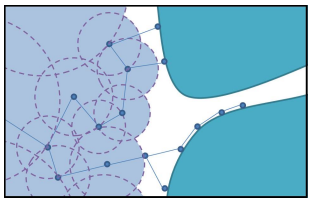


Fig. 6. This figure shows free hyperspheres that approximate wide-open free spaces.

Let us first define a *free hypersphere* in the configuration space to be a hypersphere, whose contained configurations do not have any collisions against the environment. Especially, we define the center of each hypersphere to be located at a node of the random tree, except for in-contact configurations; in-contact configurations have contacts by the definition and thus we do not consider them as wide-open free spaces. Also, we set the radius of each hypersphere to be the distance computed between the center node of the hypersphere and its nearest neighbor node (Fig. 6). Note that when we add a new node to the random tree, we compute the distance, d_{NN}, between the new node and its nearest neighbor node, and use the distance for the radius of each hypersphere associated with the new node and its nearest neighbor node.

In order to determine whether a hypersphere of a node is free hypersphere or not, we use the non-colliding line test. The non-colliding line-test generates a random line starting from the center node, q_n, of the hypersphere. If there are no collisions in the random line, we treat the hypersphere to be free hypersphere. Note that the non-colliding line-test acts as an efficient probability function in terms of identifying whether a region can be classified as a wide-open free space; even though performing the non-colliding line-test one time may incorrectly identify a hypersphere as a free hypersphere, performing it whenever a node is chosen as a nearest neighbor node can identify a region correctly in a probabilistic sense.

To generate a line used for a non-colliding test, we uniformly generate a random direction for the line segment starting from the center node q_n of a hypersphere. Then we compute another end point, q_e, of the line along the chosen line direction. q_e is computed by a random distance generated by a Gaussian distribution function, whose mean

Algorithm 5 SR-RRT Planner

Require: tree T
  T.AddVertex (q_{init})
  repeat
    $q_r \leftarrow$ RandomState(); $q_n \leftarrow$ NearestNeighbor(q_r, T)
    if q_n is not an in-contact configuration AND distance(q_n, q_r) $< q_n.d_{NN}$ **then**
      if NonCollidingLineTest(q_n) **then**
        CONTINUE
      end if
    end if
    Re-test(q_n)
    $q_{new} \leftarrow$ SRExtend(q_n, q_r)
    if $q_n.d_{NN} >$ distance(q_n, q_{new}) **then**
      $q_n.d_{NN} \leftarrow$ distance(q_n, q_{new})
    end if
    $q_{new}.d_{NN} \leftarrow$ distance(q_n, q_{new})
    T.AddVertex (q_{new})
    T.AddEdge (q_n, q_{new})
  until a collision-free path between q_{init} and q_{goal} is found

and standard deviations are set to be the half of the radius of the hypersphere associated with q_n.

Once we have a set of approximate, free hyperspheres, we discard a random sample if the random sample is within the free hypersphere of its nearest neighbor node. A pseudo code of our non-colliding line-test is shown at Algorithm 4.

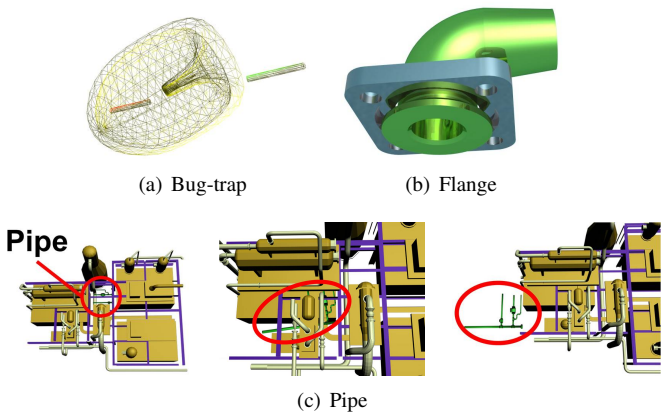
E. Overall Algorithm

A pseudo code of the overall algorithm of our SR-RRT planner is shown at Algorithm 5. We generate a random sample q_r and find its nearest neighbor node q_n. Then we discard it if the sample q_r is inside the wide-open free spaces probabilistically defined by performing the non-colliding list-test. Otherwise, we perform our extension algorithm after performing the re-testing process. As the last step of our method, we update the nearest neighbor distance, d_{NN}, and the tree, T. We iteratively perform these steps until we find a collision-free path between the initial and goal configurations.

V. Results and Discussions

We have implemented SR-RRT for three dimensional rigid body robots on an Intel i7 desktop machine that has 3.33GHz CPU. Our method is built upon a basic RRT integrated with an efficient optimized-based retraction method [9]. We use PQP library [31] for collision detection. For the local planning, we use a linearly interpolated motion between two configurations and check whether we have collisions on a fixed number of intermediate configurations on the linearly interpolated motion.

Benchmarks: We test our method against ten different benchmarks that have different characteristics. We classify our benchmarks as three types: environments with a high portion of narrow passages (NP), environments that do not have such narrow passages (Non-NP), and in-between



(a) Bug-trap (b) Flange

(c) Pipe

Fig. 7. This figure shows three different benchmarks that have narrow passages. Also, three image shots that show the planning result are shown for moving the pipe out of an industrial environment.

environments (Mixed). Benchmarks with the type of Non-NP include S-tunnel 0.85 (Fig. 2) and S-tunnel 1.0. The S-tunnel model has a S-shape of tunnel and we change its characteristics by scaling a cubic-shaped robot; S-tunnel x uses a scaling factor of x for the robot. Benchmarks with the type of NP include S-tunnel 1.15 and 1.3, flange (Fig. 7-(b)), bug-trap (Fig. 7-(a)), wiper 1.0 (Fig. 3), pipe (Fig. 7-(c)), and living room (Fig. 1). Finally, wiper 0.9 belongs the class of Mixed; we scale down the size of the wiper. The dimensions of the configuration spaces in all of these benchmarks are six. The benchmark information is summarized in Table I.

A. Results and Comparisons

We run our method, SR-RRT, with each of our benchmarks in 100 times and report the average running time in Table I. In order to demonstrate the relative benefits of our method, we have also implemented the basic RRT, called RRT, and the optimization-based retraction RRT, called RRRT [9], which are described in Sec. III-A. We use the same values for parameters (e.g., the collision checking frequency used in a local planner) that are shared between different versions of RRT methods.

For all the benchmarks, our method computes collision-free paths in less than three minutes. In the pipe and living room benchmarks, our method spends over one minute on average to compute a collision-free path, since these models consist of more than 40 K triangles and have narrow passages.

For NP-typed benchmarks, our method shows higher (e.g., 72% higher on average) performance over RRRT and much higher (e.g., 7.7 times higher on average) over RRT. Since RRT does not bias its sampling towards narrow passages, it runs quite slowly in NP-typed benchmarks. RRRT shows higher performance than RRT. However, because of its higher computational overheads caused by performing retractions on all the contact spaces, RRRT runs slower than our method.

For Non-NP-typed benchmarks, RRRT shows lower (e.g., about 50%) over RRT, because RRRT excessively generates many in-contact configurations, most of which do not capture a new connectivity of free spaces, but pose the

					SR-RRT			
Model	#. Tri	Type	RRT	RRRT	BL-test	+NC-test	+PCA	Rep. Img.
S-tunnel 0.85	64	I	53.68	98.83	35.45	28.83	28.14	Fig. 2
S-tunnel 1.0	64	I	96.21	141.22	78.56	70.01	65.57	See below
S-tunnel 1.15	64	III	406	129.5	53.66	53.26	52.08	See below
S-tunnel 1.3	64	III	646	134.72	81.08	80.89	78.9	See below
Bug-trap	2.7k	III	145	39.72	30.47	26.4	22.17	Fig. 7-(a)
Flange	6.3k	III	589.24	33.12	32.32	30.06	27.99	Fig. 7-(b)
Wiper 0.9	26.7k	II	107.57	107.75	50.87	53.51	48.05	See below
Wiper 1.0	26.7k	III	283.56	141.5	90.64	86.15	82.91	Fig. 3
pipe	48.4k	III	639.99	225.79	176.42	170.6	166.08	Fig. 7-(c)
Livingroom	137k	III	776.54	140.94	94.15	84.17	77.2	Fig. 1

TABLE I

MODEL COMPLEXITY AND PERFORMANCE RESULTS; SEE SEC. V FOR THE DETAILED INFORMATION. TYPE I, II AND III REPRESENT MODEL TYPES OF NON-NP, MIXED AND NP, RESPECTIVELY.

Model (scale)	Flange	S-tunnel 1.0	S-tunnel 1.3	Bugtrap	Wiper 1.0	Pipe	Room
Retraction	76%	24%	52%	51%	57%	50%	31%
BLTest	6%	13%	14%	28%	12%	6%	14%
NCTest	1%	3%	3%	9%	2%	1%	6%
RRT	17%	60%	31%	12%	29%	43%	49%
Cull % of BLTest	35%	98%	83%	51%	72%	74%	83%
Cull % of NCTest	87%	86%	84%	78%	97%	90%	89%

TABLE II

BREAKDOWN OF THE RUNNING TIME OF SR-RRT. CULL % REFERS TO THE CULLING RATIOS OF TWO TYPES OF LINE-TESTS.

computational overheads. On the other hand, our method still shows 68% improvements on average over RRT. Even though its improvement over RRT is weaker than improvements made with NP-typed benchmarks, our method shows improvements over RRT even in these benchmarks that do not have narrow passages. Furthermore, our method shows about three times improvements over RRRT, since our method selectively performs retraction operations. These results indicate that the overheads of our line-tests are small and demonstrate the robustness of our methods.

For the Mixed benchmark (e.g., wiper 0.9 benchmark), RRT and RRRT show the similar running time, while our method still shows 2.24 times improvement over them. Therefore, we can conclude that our method works more robustly than RRT and RRRT for a wide variety of environments that have or do not have narrow passages.

We measure how much improvement we make with each component of our contributions. By enabling bridge line-tests we observe 74.6% improvement over RRRT. We achieve 10.4% further improvement by additionally enabling the non-colliding tests. Also, by adopting the PCA-transformed line directions, we observe additional 14.6% improvement. The SR-RRT column in Table I shows incremental effects from each component on each tested benchmark.

B. Discussions

Table II shows a breakdown of the running time of our method in different components including the retraction, two line-tests, and other parts (e.g., computing in-contact configurations, connecting nodes, etc. related to the basic RRT); time spent for performing PCA is included in the bridge line-test. Depending on benchmarks, components take different portions of the overall running time. In most of the benchmarks, two line-tests take about 7% to 17% in the overall running time. For the bug-trap benchmark, these two tests take 37% of the overall running time, since many in-contact samples are chosen as nearest neighbors for the tree expansion. Still retractions and basic RRT operations take much larger portions than our two tests.

The culling ratios of samples due to the non-colliding line-tests are quite high (e.g., 78% to 97%) across all the benchmarks. On the contrary, the culling ratios of retraction operations due to bridge line-tests relatively vary a lot. The flange benchmark shows the lowest culling ratio (e.g., 35%) because there are lots of narrow passages while getting the flange out of the curved pipe. In the S-tunnel 1.0 benchmark, we achieve up to 98% culling ratio, since the benchmark does not have any narrow passages.

Our two line-tests check for collisions in a fixed number of intermediate configurations on a line. For the frequency of checking collisions in our line-tests, we simply use the same resolution of discrete collisions employed in the local planner. Therefore, their costs per each line-test vary depending on the frequency that the local planner uses. Among our benchmarks, each bridge and non-colliding line-test takes 3.4 ms and 1 ms respectively on average. Since we perform bridge line-tests near narrow passages, the employed collision algorithm needs to traverse a bounding volume hierarchy deeper in order to localize collisions [31]. As a result, the bridge line-test requires about three times higher running time than the non-colliding line-test.

We have tested S-tunnel models with varying scaling factors for the cubic-shaped robot (Table I). As we increase the scaling factor, the benchmark poses a more challenging narrow passage problem. In this setting, our method achieves noticeably higher performance improvements over RRT, as we have more challenging narrow passage problems (e.g., 1.9, 7.8, and 8.19 times improvements over S-tunnel 0.85, 1.15, and 1.3 respectively). On the other hand, our method shows slowly diminishing improvements over RRRT as we increase the scaling factor. This is mainly because there are more narrow passages and thus culling ratio of our line-tests decrease. For example, the flange benchmark has narrow passages in most of its free space. As a result, our method shows almost similar, but still higher performance to that of RRRT. These results also demonstrate both the low computational overheads and robustness of our method.

Limitations: Our method works quite well with all the tested benchmarks. Even though the proposed line-tests with PCA computations can be performed without much overheads, their accuracy in terms of identifying narrow passages and wide-open areas may not be high in other scenes. This is mainly because we check a fixed number of configurations on a line in the configuration space. Also, even though there are no narrow passages, our bridge line-tests may treat sharp corners as narrow passages. Our method does not guarantee to always improve the performance over the basic RRT and the optimization-based retraction RRT, because of these properties. Also, even though we

identify narrow passages, they may not contribute to the final solution. Nonetheless, among all the tested benchmarks, our method shows improvements over other tested RRT methods, because of its low computational overheads and probabilistically high accuracy.

VI. CONCLUSIONS AND FUTURE WORKS

We have presented a novel retraction-based planner, SR-RRT, which selectively performs the retraction operations only near narrow passages. To perform such adaptive retraction method, we proposed a bridge line-test that can efficiently identify whether a region contains narrow passages or not. Especially, we performe PCA with local free spaces and generate lines that can cross potentially-existing narrow passages with a high probability. Also, in order to generate samples near such narrow passages, we presented a non-colliding line-test that can identify whether a region is a wide-open free space or not. These line-tests have minor computational overheads and can work for high dimensional configuration spaces. Moreover, our method has been demonstrated to show 6.7 times and 2 times improvements on average compared to a basic RRT and an optimization based RRT methods, respectively. More importantly, our method shows the highest performance among all the tested methods with all the tested benchmarks, while the performance of other methods depend on the type of environments. This result demonstrates higher robustness and generality of our method.

There are many avenues for future research directions. We would like to design more accurate, yet efficient filtering methods. Also, we would like to identify collision-free paths in a multiresolution approach [23], [24] in order to more efficiently find such paths. It will be very challenging to design a multiresolution technique for environments that contain narrow passages shown in this paper. Finally, the optimization-based retraction method has been extended to articulated robots. We would also like to test our method to such cases.

ACKNOWLEDGMENTS

We would like to thank anonymous reviewers and members of SGLab for helpful feedbacks. We also would like to thank Parasol Lab for their benchmarks and Kavraki Lab for their software. This research was supported in part by MKE/KEIT [KI001810035261], MSRA, MKE/MCST/IITA [2008-F-033-02], BK, DAPA/ADD (UD080042AD), MEST/NRF/WCU (R31-2010-000-30007-0), KMCC, MCST/KOCCA/CT/R&D 2011, MEST/NRF (2011-0030822).

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.M. LaValle, "Rrt-connect: An efficient approach to single-query path planning", in *IEEE Int. Conf. on Robotics and Automation*, 2000, pp. 995–1001.

[3] H. Chang and T. Li, "Assembly maintainability study with motion planning", in *IEEE Int. Conf. on Robotics and Automation*, 1995.

[4] J.C. Latombe, "Motion planning: A journey of robots, molecules, digital actors, and other artifacts", *Int. Journal of Robotics Research*, pp. 1119–1128, 1999.

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

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

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

[8] D. Hsu, Tingting Jiang, J. Reif, and Zheng Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners", in *IEEE ICRA*, 2003, vol. 3, pp. 4420 – 4426 vol.3.

[9] L. Zhang and D. Manocha, "An efficient retraction-based rrt planner", in *IEEE Int. Conf. on Robotics and Automation*, 2008, pp. 3743 –3750.

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

[11] D. Hsu, G. Sanchez-Ante, Ho lun Cheng, and J.-C. Latombe, "Multi-level free-space dilation for sampling narrow passages in prm planning", in *IEEE ICRA*, 2006, pp. 1255 –1260.

[12] S. Redon and M.C. Lin, "Practical local planning in the contact space", in *IEEE Int. Conf. on Robotics and Automation*, 2005, pp. 4200 – 4205.

[13] M. Saha and J.-C. Latombe, "Finding narrow passages with probabilistic roadmaps: the small step retraction method", in *IEEE/RSJ Int. Conf. on IROS*, 2005, pp. 622 – 627.

[14] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006.

[15] D. Hsu, J.C. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning", *IEEE Transactions on Robotics*, vol. 25, no. 7, pp. 627 – 643, 2006.

[16] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation", in *IEEE/RSJ Int. Conf. on IROS*, 2002, pp. 2383–2388.

[17] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with rrts", in *IEEE Int. Conf. on Robotics and Automation*, 2006, pp. 1243–1248.

[18] M.S. Branicky, M.M. Curtiss, J. Levine, and S. Morgan, "Sampling-based planning, control and verification of hybrid systems", *IEE Proceedings Control Theory and Applications*, pp. 575 – 590, 2006.

[19] J. Cortes, L. Jaillet, and T. Simeon, "Molecular disassembly with rrt-like algorithms", in *IEEE Int. Conf. on Robotics and Automation*, 2007, pp. 3301 –3306.

[20] B. Burns and O. Brock, "Single-query motion planning with utility-guided random trees", in *IEEE Int. Conf. on Robotics and Automation*, 2007, pp. 3307 –3312.

[21] A. Yershova, L. Jaillet, T. Simeon, and S.M. LaValle, "Dynamic-domain rrts: Efficient exploration by controlling the sampling domain", in *IEEE ICRA*, 2005, pp. 3856 – 3861.

[22] S.R. Lindemann and S.M. LaValle, "Incrementally reducing dispersion by increasing voronoi bias in rrts", in *IEEE Int. Conf. on Robotics and Automation*, 2004, pp. 3251–3257.

[23] J. Guitton, J.-L. Farges, and R. Chatila, "Cell-rrt: Decomposing the environment for better plan", in *IEEE/RSJ Int. Conf. on IROS*, 2009, pp. 5776–5781.

[24] S. Rodriguez, S. Thomas, R. Pearce, and N. Amato, "Resampl: A region-sensitive adaptive motion planner", in *Int. Workshop on the Algorithmic Foundations of Robotics*, 2006.

[25] 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*, 2005, vol. 17, pp. 361–376.

[26] S Dalibard and J.P. Laumond, "Linear dimensionality reduction in random motion planning", *Int. Journal of Robotics Research*, 2011.

[27] A. Shkolnik and R. Tedrake, "Sample-based planning with volumes in configuration space", *CoRR*, vol. abs/1109.3145, 2011.

[28] D. Hsu, G. Sanchez-Ante, and Z. Sun, "Hybrid prm sampling with a cost-sensitive adaptive strategy", in *IEEE Int. Conf. on Robotics and Automation*, 2005, pp. 3874 – 3880.

[29] I. Jolliffe, "Principle component analysis", in *Springer-Veriag*, 1986.

[30] M. Sharir, "Algorithmic motion planning", in *Handbook of Discrete and Computational Geometry*, J. E. Goodman and J. O'Rourke, Eds., chapter 40, pp. 733–754. CRC Press LLC, Boca Raton, FL, 1997.

[31] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha, "Distance queries with rectangular swept sphere volumes", *Proc. of IEEE Int. Conference on Robotics and Automation*, pp. 3719–3726, 2000.