2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

An Adaptive Roadmap Guided Multi-RRTs Strategy for Single Query Path Planning

Wei Wang, Yan Li, Xin Xu, Simon X. Yang

***Abstract*—During the past decade, Rapidly-exploring Random Tree (RRT) and its variants are shown to be powerful sampling based single query path planning approaches for robots in high-dimensional configuration space. However, the performance of such tree-based planners that rely on uniform sampling strategy degrades significantly when narrow passages are contained in the configuration space. Given the assumption that computation resources should be allocated in proportion to the geometric complexity of local region, we present a novel single-query Multi-RRTs path planning framework that employs an improved Bridge Test algorithm to identify global important roadmaps in narrow passages. Multiple trees can be grown from these sampled roadmaps to explore sub-regions which are difficult to reach. The probability of selecting one particular tree for expansion and connection, which can be dynamically updated by on-line learning algorithm based on the historic results of exploration, guides the tree through narrow passage rapidly. Experimental results show that the proposed approach gives substantial improvement in planning efficiency over a wide range of single-query path planning problems.**

I. Introduction

ROBOT path planning has been one of the fundamental problems over the last couple of decades in such areas as robotics, artificial intelligence, as well as computer graphics. The original description of the problem is to plan a collision-free path for a robot made of an arbitrary number of polyhedral bodies among an arbitrary number of polyhedral obstacles between two collision-free queried positions of the robot, which has been shown to be PSPACE-complete by complex geometric analysis [1].

The well known complete motion planning algorithms, such as cell decomposition and visibility roadmaps, require explicit representation of robot configuration space. They are usually computationally intractable and hard to implement for practical applications [2]. "The curse of dimensionality" has lead to the development of randomized sampling-based motion planners, which can solve many previously considered hard problems successfully and quickly.

PRM [3] and RRT [4] are two typical randomized sampling-based motion planning methods. The PRM is a typical path planning algorithm for multi-query problems. It attempts to construct a graph structure by randomly sampled roadmaps so as to reduce the original planning problem to a graph traversing and searching problem. The PRM algorithm is recognized as one of the most successful randomized path planning algorithm for multi-dof robots. The RRT is initially proposed for single-query problems for non-holonomic robots [4]. The RRT and its variants are based on a tree-like data structure, whose nodes represent the sampled collision-free configurations and edges correspond to local paths connecting two nearby nodes. As the RRT planner incrementally grows a tree to explore unknown space, it may behave poor when there are narrow passages through which the planner has to pass to reach the goal configuration.

A narrow passage is a small region whose removal changes the connectivity of the component free space substantially. Most of the early works on PRM and RRT usually apply uniform randomized sampling strategy to sample collision-free configurations. The apparent drawback is that much computation time will be wasted in sampling wide open area before identifying a narrow passage, due to its small volume against the volume of the whole configuration space. Even two nearby nodes, which belong to two different components connected by one narrow passage, is hard to be connected by the strategy.

One solution is to develop non-uniform sampling technique that can efficiently allocate the computation resources according to the geometric complexity of local regions [5] [6]. Dalibard *et al*. [7] applied PCA method to recognize the favored direction of the narrow passage by dozens of collision-free samples already obtained.

The other solution is to create more than one tree to explore sub-regions simultaneously. Kuffner *et al*. [8] described a bi-directional RRT-Connect algorithm that grows two trees from the initial and goal configurations. Path planning based on expansions of multiple RRT trees can be found in [9] [10] [11].

The above multi-trees methods share the common idea that each tree has equal opportunity for expansion no matter whether it belongs to the restrained regions or not. The equiprobable expansion strategy confronts with the same drawback of uniform sampling strategy mentioned above. The fact that computation resources should be allocated in proportion to the geometric complexity of local regions motivates us to develop an adaptive multi-RRTs path planning algorithm in a single-query scenario.

Manuscript received September 15, 2009. This work was supported in part by National High Technology Research and Development Program of China (863 Program) under Grant 2006AA110114 and National Science Foundation of China (NSFC) under Grant 90820302 and Grant 60774076.

W. Wang, Y. Li and X. Xu are with the College of Mechatronics and Automation, National University of Defense Technology, Changsha, 410073, China (emails: wangmail200@yahoo.com.cn; spark99@263.net; xinxu@nudt.edu.cn).

S. X. Yang is with the ARIS Lab, School of Engineering, University of Guelph, Guelph, Canada (email: syang@uoguelph.ca).

The adaptive multi-RRTs algorithm proceeds in two stages: firstly, a non-uniform Bridge Test sampling algorithm [12] is improved to identify global important roadmaps in some crucial areas. Next, multiple trees are grown from the sampled roadmaps. The probability of selecting one particular tree for expansion, which can be dynamically updated by on-line learning algorithm based on the historic results of exploration, guides the tree through narrow passage rapidly. If two trees encounter in a collision-free configuration successfully, they are merged into a larger tree. The algorithm iterates to grow and merge trees so as to find a global collision-free solution path.

The adaptive selection model that forms the distinguished feature of our method is inspired by an n-armed bandit problem introduced in the book [13]. Since non-uniform sampling and exploring strategy is adopted in both stages, the adaptive Multi-RRTs algorithm can concentrate on exploring more complex sub-regions instead of wide open areas. Plenty of experimental results show that the proposed approach gives substantial improvement in planning efficiency over a wide range of single-query path planning problems.

In section II we present an improved Bridge Test algorithm that samples global roadmaps to identify narrow passages, section III describes the implementation details of the adaptive Multi-RRTs algorithm. In section IV we describe the experimental setup and the set of benchmarks used to test the performance of our planner. Section V draws conclusions.

II. NARROW PASSAGE SAMPLING

A. Related work

Difficulty posed by narrow passages and its importance were firstly issued in early works of PRM planners, as their successes highly depended on effective samples to capture the connectivity of free configuration space. As a result, plenty of non-uniform random sampling methods for PRM planners have been proposed to efficiently sample global roadmaps to identify critical regions.

It's possible to improve the sampling strategy in two directions. The first consideration is based on geometric analysis of the workspace information that derives what the important areas are [14] [15]. The other non-uniform sampling strategies are based on a series of collision detections [12] [16].

The so-called Bridge Test algorithm attempts to identify a narrow passage by using three samples along a line segment. Considering high efficiency and simplicity of the Bridge Test algorithm, we improve it to serve as a preprocessing stage to sample global important roadmaps in narrow passages.

B. Improved Bridge Test Algorithm

For clarity, we firstly give some definitions and terms. For a robot with n degrees of freedom, the n-dimensional topological space describing all possible positions and orientations of the robot is called configuration space, denoted by C. A configuration q is free if the rigid bodies of the robot placed at q don't collide with obstacles or with other bodies of the robot. The set of all free configurations in C are defined to be free space, denoted by $\mathcal{F}$. The obstacle space $\mathcal{B}$ is defined to be the complement of $\mathcal{F}$: $\mathcal{B}=C/\mathcal{F}$. Therefore, the path planning problem can be defined as follow: Given a pair of initial and goal configurations q_{init} and q_{goal}, find a continuous collision-free path τ: $[0, 1]\rightarrow\mathcal{F}$, such that $\tau(0)=q_{init}$, and $\tau(1)= q_{goal}$.

Intuitively, a narrow passage in C has at least one restricted direction, along which small perturbations will cause collision between robot and obstacle. It's easy to sample at random a short line segment through a collision-free configuration q such that the two endpoints of this line segment lie in $\mathcal{B}$ (figure 1). The line segment resembles a bridge across the narrow passage so that it's called Bridge Test. If we successfully obtain such a line segment, we say that the point $q\in\mathcal{F}$ passes the test. Clearly building short bridges is much easier in narrow passage than in wide open free space.

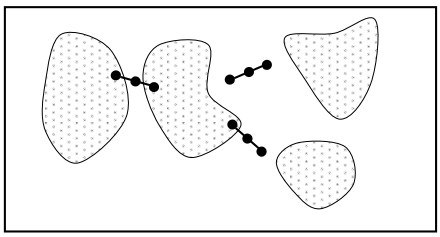

Fig. 1. The illustration of Bridge Test. Note that only the point lies in narrow passage can pass the Bridge Test.

In order to find a configuration that passes Bridge Test, two endpoints must be selected in advance which should be assigned inside the obstacles and close enough to ensure the mid-point of the bridge lying in a narrow passage. The original Bridge Test algorithm samples the first bridge endpoint q_f in C randomly, and then samples the second bridge endpoint q_s in the neighboring of q_f according to a specified probability density function λ_q, i.e. the product of the independent Gaussians on each axis of C. The means of each Gaussians are set to be the position of q_f in C. However it's a non-trivial work to obtain the standard deviation σ for each independent Gaussian distribution, as σ depends on the width of narrow passage that is always problem-specific. Instead, we exploit the intrinsic attribute in the sampling strategy to obtain the second bridge endpoint q_s.

As the distance deviation between bridge end-points q_f and q_s should not be too far, sampling the second end-point q_s randomly in C is unreasonable. Instead, we scale the sampled configuration to the neighborhood of the first end-point q_f by the strategy described as follow: Suppose q_{min} denotes the lower limits for each coordinate in the configuration space. A candidate configuration point q_c is sampled at random without collision detection. The difference of (q_c-q_{min}) is multiplied by a small scale $1/l$ to obtain the offset:

$$dq = (q_c - q_{min})/l \quad (1)$$

q_s is then evaluated by:

$$q_s = q_f + (-1)^n \cdot dq \quad (2)$$

In which, n is a random integer, implying that q_s may lie either in front of or behind q_f.

Algorithm 1 BRIDGE_TEST()

```
1   nplist.clear();
2   for k = 1 to K do
3     q_f ← RANDOM_POINT();
4     if not COLLISIONFREE(q_f) then
5       q_c ← RANDOM_POINT();
6       d_q ← (q_c - q_min)/l;
7       q_s ← q_f + (-1)^n * d_q;
8       if not (COLLISIONFREE(q_s) and q_s ∈ C then
9         q_m ← (q_f + q_s)/2;
10        if COLLISIONFREE(q_m) then
11          nplist.push_back(q_m);
12        if nplist.size() > maxSize then
13          return nplist;
14  return nplist;
```

Fig. 2. Improved Bridge Test algorithm.

The pseudocode of the improved Bridge Test algorithm is shown in figure 2. In line 4, 8, and 10, collision detection function COLLISIONFREE() is called to test whether a configuration in C is collision-free or not. When middle point q_m passes Bridge Test, it's pushed into a sampled roadmap list *nplist* that will be used to grow new additional trees in the successive planning procedure. The algorithm will be terminated when either the maximum allowed number of roadmaps *maxSize*, or the maximum number of iterations has been reached.

Even though there's only one parameter l to perform Bridge Test for finding a verified roadmap, it's not easy to determine an optimal value for general problems. We typically set l in the range of 10-30 through a wide variety of independent experiments, which is common enough to capture the generic property of Bridge Test.

The parameter *maxSize* is also relative to specific problem. Too small value can't identify all of the narrow passages across the entire narrow passages, while too large value may cost much computation time, as collision detection subroutine might be called frequently. Empirically we set *maxSize* in the range 5~15. We also provide user interface to adjust the parameters on the fly.

III. Adaptive Multi-RRTs Algorithm

A. RRT-Connect algorithm

Starting at a given initial configuration, the classic RRT prefers to grow a tree-based data structure to incrementally explore the unknown configuration space. A node in tree corresponds to a configuration in C, while the edge between nodes indicates the connecting path between two distinct configurations. The probability that a configuration is selected for extension is proportional to the area of its Voronoi region, so the RRT tends to rapidly grow in the unknown regions of $\mathcal{F}$. It was shown in [8] that in the limit, the coverage of the configuration space is uniform when the number of nodes in the tree tends to infinity. Thus, the original RRTs algorithm is a no-biased planner.

Instead of extending one step toward the sampled target configuration each time, a more greedy strategy is to extend several steps till either collision with obstacle is detected, or the target is reached. The strategy is called RRT-Connect, and the pseudocode is shown in figure 3. The algorithm attempts to sample configurations along the line connecting the random target q_{target} and its nearest neighboring configuration q_{near} in $\mathcal{T}$. if a new collision-free configuration q_{new} is generated, q_{new} is added to $\mathcal{T}$ as a new node, and an edge is added to connect q_{near} and q_{new}. The algorithm will return *Reached* if q_{new} is close to q_{target}. If several steps toward q_{target} have been taken, it will return *Advanced*. Otherwise, it will return *Trapped*, which means no step is extended forward.

Algorithm 3 CONNECT($\mathcal{T}$, q_{target})

```
1   S ← Trapped;
2   q_near ← NEAREST_NEIGHBOR(T, q_target);
3   do
4     if NEW_CONFIG (q_near q_target, q_new) then
5       T.add_vertex(q_new);
6       T.add_edge(q_near, q_new);
7       if DIST(q_new , q_target) < δ then
8         return Reached;
9       else
10        S ← Advanced;
11        q_near ← q_new;
12    else
13      return S;
```

Fig. 3. The original RRT-Connect algorithm.

B. The adaptive tree selecting strategy

The Bridge Test algorithm is served as a pre-processing step in our adaptive Multi-RRT framework to analyze and extract information in narrow passages. Once multiple trees are grown from the sampled roadmaps in the second step, the bidirectional RRT-Connect algorithm can be adopted as a local path planner to connect a pair of candidate trees due to its powerful planning capability. Problem arises when considering which pair of trees should be selected to extend and connect. We design the strategy as follows.

Assume that there are n trees in our framework, including trees rooted at the queried configurations and additional trees rooted at intermediate sampled roadmaps. These trees are assigned with a selecting probability p_i (i = 1,2,⋯,n). We build the probabilistic distribution model based on the consideration that the trees in the restricted regions should be allocated more computation time to capture the connectivity. It seems difficult to choose the optimal values for each p_i in advance. So let's treat them as the dynamic value that change over time.

At the beginning, $p_1, p_2, \cdots, p_n$ for each tree is set to be the same. To update them, we maintain a weight w_i (i = 1,2,⋯,n)

and adjust w_i by assigning a reward r after each expansion performed by a particular selected tree. For a randomly sampled target q_{target}, if the selected tree is hard to extend one step toward q_{target}, it's implied that the tree may lie in the restricted region of the free configuration space. On the contrary, if it's easy to reach the δ neighboring region of q_{target} by RRT-Connect extension, it's implied that the tree may lie in the open wide area. As a result, the output of the RRT-Connect algorithm in each time step t can be used to verify whether a selected tree is easy to explore or not. Initially we set $w_i = 0$ for all i, and we update w_i based on three conditions of exploring result:

1) For a selected tree, if the RRT-Connect algorithm returns *Reached*, which means the tree can reach the δ neighboring region of q_{target} successfully, implying that the tree may lie in the open wide area, We set the reward r = -1. The exploration of this tree in the next round is restrained (figure 4-(a)).
2) For the circumstance that the RRT-Connect algorithm returns *Advanced*, which means the tree can extend several steps, yet not close enough to reach the target point, we set the reward r = 0. The exploration of this tree in the next round is neither encouraged nor restrained (figure 4-(b)).
3) If the RRT-Connect algorithm returns *Trapped*, which means no step is extended forward. It's reasonable to assume that the tree lies in the cluttered area. We set the reward r = 1. The exploration of this tree in the next round is encouraged (figure 4-(c)).

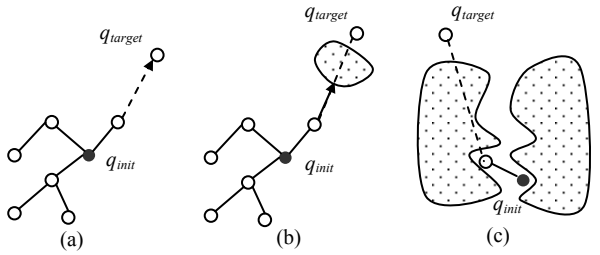

Fig. 4. Illustration of reward assignment. (a) Open area: r = -1; (b) Area with a few obstacles: r = 0; (c) Cluttered area: r = 1.

The weight of w_i is updated by:

$$w_i(t+1) = w_i(t) + (r_i(t+1) - w_i(t))/(k_i(t)+1) \quad (3)$$

Where $k_i(t)$ is the updating count of w_i. If a tree is not selected in the current step, its weight remains the same as before. Based on the weight assignment in each time step t, we choose the candidate expansion tree with the probability:

$$p_i(t) = \exp(w_i(t)/\tau) / \sum_{j=1}^{n} \exp(w_j(t)/\tau) \quad (4)$$

The above formula is called Gibbs, or Boltzmann distribution, where τ is a positive parameter called the *temperature*. High temperatures cause the actions of selecting trees to be nearly all equiprobable no matter how weights update. The lower the temperature is, the more difference the selection probability is. In our implementation, τ is set to be 0.8.

It's observed that the update rule (3) has the following property:

$$\sum_{t=1}^{\infty} \frac{1}{k_i(t)+1} = \infty \text{, and } \sum_{t=1}^{\infty} \left(\frac{1}{k_i(t)+1}\right)^2 < \infty \quad (5)$$

The first condition guarantees that the steps are large enough to eventually overcome any initial conditions or random fluctuations. The second condition guarantees that eventually the steps become small enough to assure convergence according to the law of large numbers [13]. Therefore, the combinatorial tree exploration strategy will be converged to the best strategy with probability one according to the law of large numbers. The result can be interpreted that the ensemble trees performs almost as well as the best component tree, though which component tree is the best is not known in advance.

C. The adaptive Multi-RRTs algorithm

Our adaptive Multi-RRTs algorithm firstly grows multiple trees from the roadmaps sampled by the improved Bridge Test algorithm. Based on the on-line update rule described above, the algorithm repeatedly selects the most probable tree to explore according to the latest probability p_i. The detailed introduction is given in figure 5. The sub-routines called by the Multi-RRTs algorithm are explained and listed below:

Algorithm 4 MULTI-RRTs(q_{init}, q_{goal})

```
1   nplist = BRIDGE_TEST(q_init, q_goal);
2   trlist = TREE_BUILD(q_init, nplist, q_goal);
3   INITIALIZE(w_i);
4   for k = 1 to K do
5       p_i = UPDATE_PROB(w_i);
6       Tc = PICK(trlist, p_i);
7       q_target = RANDOM_CONFIG();
8       state = CONNECT(Tc, q_target);
9       if state = = Reached then
10          r = -1;
11      else if state = = Advanced then
12          r = 0;
13      else if state = = Trapped then
14          r = 1;
15      (Tn, q_near) = NEAREST_NEIGHBOR(nplist, q_new);
16      if CONNECT(Tn, q_near) = = Reached then
17          Tc.MERGE(Tn);
18          trlist.REMOVE(Tn);
19      UPDATE_WEIGHTS(w_i, r);
20      if CONTAIN(Tc, q_init) and CONTAIN(Tc, q_goal) then
21          return PATH(Tc);
22  return Failure;
```

Fig. 5. The adaptive Mulit-RRTs algorithm.

- TREE_BUILD(): Building trees from the intermediate sampled roadmaps, as well as the initial and goal configurations, respectively;
- INITIALIZE(): Initializing weights $w_i = 0$ for all i;
- UPDATEP_ROB(): Updating probability p_i in each time step t;
- PICK(): Picking up one tree from the tree list *trlist* according to the probability distribution p_i;
- RANDOM_CONFIG(): sampling a configuration in C

at random;

- NEAREST_NEIGHBOR(): Searching for a tree from the tree list *trlist* that has a node q_{near} in the neighboring region of q_{new};
- UPDATE_WEIGHTS(): Updating weights w_i according to reward assignment r;
- MERGE(): Merging one tree to the other;
- CONTAIN(): Checking whether a configuration is contained in a specified tree or not;

If the improved Bridge Test algorithm can not find such a list of the sampled roadmaps to identify narrow passages, the adaptive Multi-RRTs algorithm will degrade to an adaptive version of the well-known bidirectional RRT-Connect algorithm. As a result, Multi-RRTs algorithm generally performs better than RRT-Connect algorithm. Though full theoretical proof is not given in this paper, our adaptive Multi-RRTs algorithm is expected to hold the property of *probabilistic completeness*. For more details about the proof of *probabilistic completeness* for bidirectional RRT-Connect algorithm, the reader is referred to [8].

As trees in narrow passage may be allocated more computation resources, the adaptive Multi-RRTs algorithm tends to bias the direction of exploration toward difficult regions, whereas RRT-Connect algorithm biases the exploration toward goal tree. Compared to previous RRT-based algorithms, extra computation time may be required in the proposed algorithm due to the manipulation of multiple trees. For the open wide environments where double trees are enough to rapidly explore and connect, the performance of our algorithm might be decreased slightly. However, we think the compensation is worthwhile compared to the performance improvement in the complex environment with plenty of narrow passages scattered over the entire configuration space.

IV. Experimental Results

We implemented the adaptive Multi-RRTs algorithm in MS-Windows platform based on Linux version of Motion Strategy Library (http://msl.cs.uiuc.edu/msl/) developed by Steven M. LaValle from University of Illinois at Urbana Champaign. As a typical extension of single-query path planning method, the performances of our adaptive Multi-RRTs algorithm is compared to the well-known bidirectional RRT-Connect algorithm and equiprobable Multi-RRTs algorithm (The selection probability for each tree is set to be the same all through the procedure of RRT expansion). For each of the experiments, we show the running times (Times), the total number of nodes in explored trees (Nodes) and the number of collision detection calls (CD Calls) in each stage averaged over 30 runs. All of the experiments are performed on a 3.0G Hz Pentium PC with 1.0G memory.

The first experiment (figure 6) is a 2-D maze problem for a point robot. Even though this problem is only two dimensional, RRT-Connect algorithm has trouble because of the many narrow passages. From Figure 6 it is seen that the intermediate trees grown from sampled roadmaps are rapidly merged to the initial or the goal tree, and the selecting probability curves tend to be stable after 1000 iterations. It's observed that the proposed algorithm generates much less nodes than those of RRT-Connect algorithm, implying higher performance. The result data in table 1 verify that the adaptive Muti-RRTs algorithm performs much better than the original RRT-Connect algorithm.

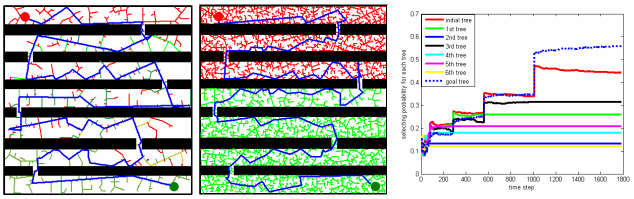

Fig. 6. Experiment 1. The initial/goal configuration is highlighted by solid red/green circle. Both the expanded trees and a solution path found by the proposed algorithm (left) and RRT-Connect algorithm (middle) are sketched, respectively. The change of the selecting probability for each tree over a period of time in the proposed algorithm is plotted in the right figure.

The second experiment (figure 7) is designed for two L-shaped 6-dof robots that have to switch the position by passing through a small hole in the center of a wall. The parameter *maxSize* in this experiment is set to five, so the total number of trees is seven. The reader is referred to the accompanying video for detailed animation. The number of dof in this experiment adds up to twelve. Note that the rigid robot must check for collisions with the other robot and with the obstacle, resulting in additional difficulty in finding a collision-free solution path. The result data shown in table 1 indicate that the proposed algorithm gives more than three times improvement in the running time over the equiprobable Multi-RRTs algorithm. The performance comparison between the proposed algorithm and bidirectional RRT-Connect algorithm is more significant. The improvement reaches as much as nine times.

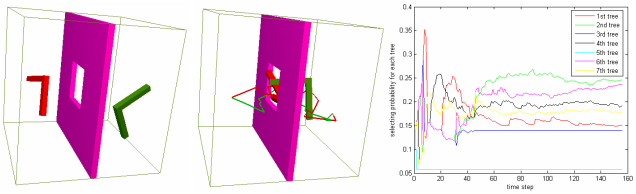

Fig. 7. Experiment 2. Left: The goal configuration. Middle: A solution path found by the proposed algorithm. Right: The change of the selecting probability for each tree over a period of time in the proposed algorithm.

The third experiment is for a 6-dof rigid stick robot (bug). The objective is to take the bug robot outside of the trap obstacle through the tiny opening (figure 8). The reader is referred to the accompanying video for detailed animation. The 3-D model is obtained from the motion planning benchmark maintained by *Parasol Lab*, Texas A&M University [17]. The problem is a special challenge for randomized sampling-based planners due to its extremely narrow corridor against the large space. As the difficult regions are effectively identified by the Bridge Test algorithm in the preprocessing step, our adaptive Multi-RRTs algorithm

provides a substantial improvement over the original RRT-Connect algorithm (Table 1). We maintain seven RRT trees by setting *maxSize* to five in this experiment as well.

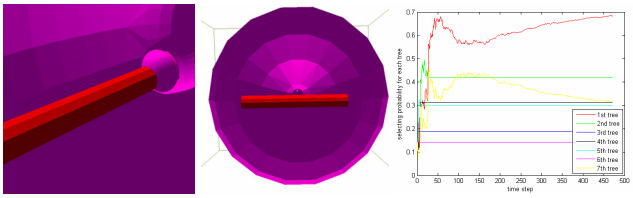

Fig. 8. Experiment 3. Left: The initial configuration. Middle: The goal configuration. Right: The change of the selecting probability for each tree over a period of time in the proposed algorithm.

TABLE I
PERFORMANCE COMPARISON FOR DIFFERENT PLANNERS

Experiment \ Planner			RRT Connect	Equiprobable Multi-RRTs	Adaptive Multi-RRTs
#1	Times (Sec)	Bridge Test	-	0.26	0.28
		Tree expansion	16.5	0.52	0.55
	Nodes		6665	1769	1804
	CD Calls	Bridge Test	-	2341	2385
		Tree expansion	39825	17168	17352
#2	Times (Sec)	Bridge Test	-	0.10	0.12
		Tree expansion	22.73	7.84	2.36
	Nodes		2519	934	387
	CD Calls	Bridge Test	-	526	559
		Tree expansion	92276	31865	9642
#3	Times (Sec)	Bridge Test	-	8.36	8.28
		Tree expansion	3532	14.97	1.72
	Nodes		856741	2478	359
	CD Calls	Bridge Test	-	96837	96253
		Tree expansion	26932429	65624	8624

V. CONCLUSIONS

We have proposed an adaptive Multi-RRTs strategy to address the single query narrow passage path planning problems. The improved Bridge Test algorithm is served as the preprocessing step to identify global important landmarks in narrow passages. Once the multiple trees are grown from the sampled roadmaps, the on-line learning strategy will adaptively update the selecting probability for each tree according to the historic results of tree explorations. The approach is simple to implement, and it's general enough for a wide class of multi-dof robot path planning problems. Experimental results show that the proposed algorithm achieves high performance over classic RRT-Connect algorithm.

We observe that several nearby samples in the same narrow passage are returned by the improved Bridge Test algorithm which will grow redundant trees. We attempt to filter the redundant roadmaps in our future research so that one sampled roadmap exactly corresponds to one narrow passage.

ACKNOWLEDGMENT

W. Wang thanks Steven M. LaValle for providing the source code of the Motion Planning Library. We thank anonymous reviewers for their constructive comments and suggestions that improved the paper.

REFERENCES

[1] J. Canny, "Some algebraic and geometric computations in PSPACE. *Annual ACM Symposium on Theory of Computing*, ACM Press, Chicago, 1988, pp. 460-469.

[2] S. M. Lavalle, *Planning Algorithms*. Cambridge: Cambridge University Press, 2006. Available at: http://planning.cs.uiuc.edu/

[3] L. E. Kavraki, P. Švetska, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration space," *IEEE Trans. on Robotics & Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[4] S. M. LaValle, "Rapidly-exploring Random Trees: A new tool for path planning," Tech. Rep., Computer Science Dept., Iowa State University, 1998.

[5] L. Jaillet, A. Yershova, S. M. LaValle, and T. Siméon, "Adaptive tuning of the sampling domain for dynamic-domain RRTs," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.

[6] B. Burns and O. Brock, "Single-query motion planning with utility guided random trees," *IEEE Intl. Conf. on Robotics and Automation*, Rome, 2007.

[7] S. Dalibard and J. P. Laumond, "Control of probabilistic diffusion in motion planning," *International Workshop on Algorithmic Foundations of Robotics*, 2008.

[8] J. J. Kuffner and S. M. LaValle," RRT-connect: An efficient approach to single-query path planning," *IEEE Intl. Conf. on Robotics and Automation*, San Francisco, 2000, pp. 995–1001.

[9] T. Y. Li and Y.C. Shie, "An incremental learning approach to motion planning with roadmap management," *IEEE Intl. Conf. on Robotics and Automation*, Washington, D.C., 2002, pp. 3411–3416.

[10] M. Strandberg, "Augmenting RRT-planners with local trees," *IEEE Intl. Conf. on Robotics and Automation*, New Orleans, 2004, pp. 3258–3262.

[11] E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd, and L. E Kavraki, "Sampling based roadmap of trees for parallel motion planning," *IEEE Trans. on Robotics*, vol. 21, no. 4. 597–608, 2005.

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

[13] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge: MIT Press, 1998.

[14] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspaces," *International Workshop on Algorithmic Foundations of Robotics*, Houston, 1998, pp. 155–168.

[15] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space," *IEEE Intl. Conf. on Robotics and Automation*, Detroit, 1999, pp. 1024–1031.

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

[17] N. M. Amato: Motion Planning Benchmark. Texas A&M University. Available at: http://parasol-www.cs.tamu.edu/dsmft/benchmarks/mp/