Evaluating Performance of Multiple RRTs

Matthew Clifton, Gavin Paul, Ngai Kwok, Dikai Liu, Da-Long Wang

Abstract—**This paper presents experimental results evaluating the performance of a new multiple Rapidly-exploring Random Tree (RRT) algorithm. RRTs are randomised planners especially adept at solving difficult, high-dimensional path planning problems. However, environments with low-connectivity due to the presence of obstacles can severely affect convergence. Multiple RRTs have been proposed as a means of addressing this issue, however, this approach can adversely affect computational efficiency. This paper introduces a new and simple method which takes advantage of the benefits of multiple trees, whilst ensuring the computational burden of maintaining them is minimised. Results indicate that multiple RRTs are able to reduce the logarithmic complexity of the search, most notably in environments with high obstacle densities.**

I. Introduction

Interest in Rapidly-exploring Random Trees (RRTs) has grown in recent years due to the increasing number of complex path and motion planning problems. The RRT search algorithm is a type of randomised path planner, especially suited to finding connected routes through complex environments. The RRT was first presented in [1] as a means of rapidly searching high-dimensional search spaces that have both algebraic and differential constraints [2]. It is primarily designed to act as a fast, single-query planner and has the key advantage that pre-processing of the environment is not required. The effectiveness of the RRT is demonstrated in [3] and has since been applied to many complex path and motion planning problems such as in [12, 13 and 15].

The RRT is a data-structure of connected states which grows in tree-like fashion to explore a search space. As randomly selected states are added to the tree it expands, thereby incrementally building a path from an initial state. The tree expands to explore the environment until a desired end state is reached. A key factor is that expansion of the tree is implicitly biased towards unexplored regions. Furthermore, the final path is always connected. Additional benefits include its simplicity to implement and few or no problem specific parameters [1] that require tuning. These desirable features make the RRT a popular path planning tool.

Manuscript received June 15, 2008. This work was supported in part by the Australian Research Council (ARC) Grant (LP0776312), the Roads and Traffic Authority, New South Wales, and the University of Technology, Sydney.

All of the researches listed are part of the ARC Centre of Excellence in Autonomous Systems (CAS), University of Technology, Sydney, P.O. Box 123, Broadway, New South Wales 2007, Australia.

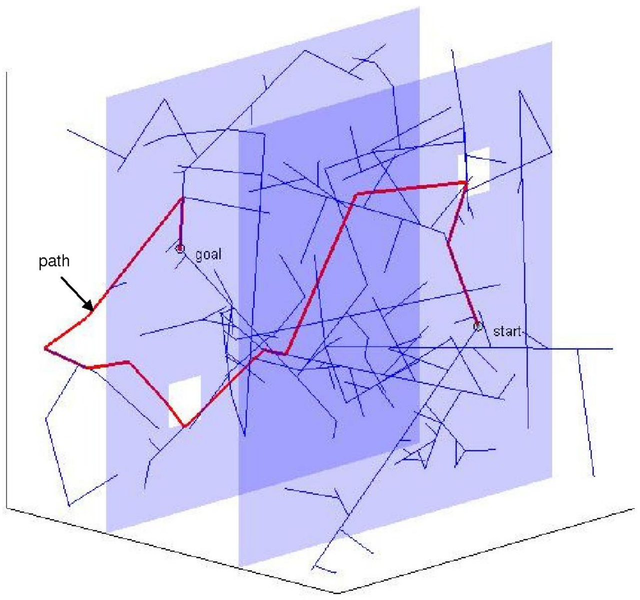


Figure 1. A Rapidly-Exploring Random Tree in 3-Dimensions

Normally the RRT is rooted at a starting point and expansion is guided by the random selection of new points from the search space. A nearest-neighbour calculation is performed to determine the closest node of the tree to the new point. An attempt is made to connect this point with the nearest node found. If no obstacles are encountered the new point is connected to the tree. However, if a collision occurs the point is discarded and the process begins again. As the algorithm progresses, a tree of connected states is constructed until a complete path to the goal state is found. These steps are summarised in Figure 2.

Ultimately, the distribution of the RRT vertices converges toward the sampling distribution, which is usually uniform [2]. This means that if a solution exists, ultimately the RRT will find it. However, it is difficult to characterise the rate of convergence as this property is heavily dependent on the shape and position of obstacles in the environment. Thus, convergence is a function of search space connectivity, which is often difficult to model.

Empirically the RRT is observed to be very fast [2] in relatively open environments. However, when many obstacles are present, the time required to find a complete path can increase significantly. Addressing this problem is the objective of this paper. Multiple trees may lead to faster convergence and prove advantageous where narrow passages exist.

Rapidly-exploring Random Tree Pseudo-code:

```
do while path not found
    NEW_POINT
    NEAREST_NEIGHBOUR
    COLLISION_CHECK
    if no collision
        CONNECT
    end if
end do
```

Figure 2. The RRT algorithm.

This section presented the basic underpinnings of the RRT search algorithm. The rest of the paper is organised as follows. Section 2 focuses on particular areas where previous research has concentrated and outlines some of the most significant challenges currently faced by the RRT. Section 3 describes an experiment devised to test a new multiple RRT strategy designed to improve performance in search spaces with low connectivity. Section 4 presents the results of this experiment. Finally, Section 5 outlines the main conclusions and also describes avenues for further research.

II. Related Work

This section describes the development of the RRT as well as the major contributions made to the algorithm up until now. The first RRT algorithm extended the search tree an incremental distance towards each new point. One of the first innovations was RRT-Connect [2] and is now the most commonly used method. This approach connects the tree directly to the new point rather than extending a small distance towards it. RRT-Connect enables long paths to be constructed with only a single nearest-neighbour calculation [2]. In addition, it allows faster exploration of the search space. However, the ability to extend further can lead to higher chances of collisions with obstacles in some (notably highly cluttered) environments.

Collision checking is the most computationally expensive function in the RRT algorithm. Consequently, it is important to consider the number of times collision checking is performed during each search iteration. The cost to the search is greatest when many obstacles are present and thus, the direct connection method can adversely affect the amount of time required to find a solution. However, on the whole the RRT-Connect approach appears to provide a significant increase in performance [2], particularly when planning in large search spaces.

Connecting to the goal point is a crucial part of the algorithm. There are several methods that may be employed to make the final connection to the goal. When multiple solutions are acceptable, it is possible to define an acceptable goal region rather than a single point. However, when a precise end-point is required there is a very low probability that the goal point will be selected at random. Exact solutions may be required for example, when a manipulator must reach a specific end-effector position and orientation. In such cases, an additional step to connect each new tree node to the goal point may be included.

The use of a second tree rooted at the goal is another approach (proposed in [2]) which facilitates connecting the goal to the tree. This prevents the tree from failing to reach the goal through mischance as the two trees actively work towards connecting to each other. In [3] the advantages of bidirectional over single search are confirmed. Having two trees growing simultaneously can greatly reduce the time required to find a path. However, this introduces new problems to consider, such as when to attempt inter-tree connections.

Regardless of whether one or two RRTs are employed, the rate of convergence to a solution is largely determined by the shape of the environment. Figure 3 illustrates one such example. Despite the fact that the algorithm has quite effectively explored one particular region of the environment, it has great difficulty reaching the region separated by the obstacle. In this example, the connectivity of the search space is greatly affected by the shape and position of the obstacle. Each time a point is selected on the other side of the barrier, a collision is very likely to occur. This results in a large number of failed connection attempts. Such situations waste the opportunity to link to the new point, despite having incurred the computational cost of the nearest neighbour search and collision check.

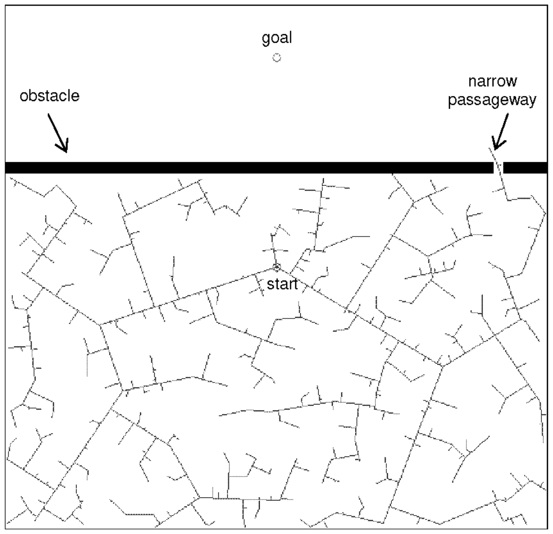


Figure 3. Environments exhibiting low-connectivity, such as that caused by some obstacles, can severely affect the convergence rate of the RRT. In the above example, the RRT is unable to extend into the top half of the search space until it has successfully navigated through the narrow passageway. The problem of stunted growth is exacerbated when multiple obstacles cause narrow passages to exist in serial.
Note: In order to clearly illustrate the problem, a simplified algorithm where only one tree, beginning at the starting point, has been permitted to grow. Even when growing from the goal is made possible, the RRT suffers similar problems when multiple obstacles exist.

The problems caused by obstacles are not only restricted to the narrow passageways they may create. When the RRT shown in Figure 3 is indeed able to navigate through the small gap, it will not be able to successfully connect to many of the points in the newly discovered region. Despite successfully finding a path through the narrow passageway, the growth is still perturbed by the presence of the obstacle. Randomly selected points are more likely to be closer to nodes on the opposite side of the obstacle rather than the node that has penetrated the gap. Thus, these points will suffer from collisions when they attempt to connect to the tree. This is yet another example in which an ordinary RRT search can be adversely affected by its environment.

Is it possible to make changes to the RRT to overcome these problems? It may seem worthwhile to allow the algorithm to continue with its nearest neighbour searching until it discovers a node with which it is indeed able to make a connection. However, modifying the algorithm in this way requires the added computational expense of a potentially large number of additional nearest neighbour searches. Subsequently, it may lead to many expensive collision checks for each search iteration.

Failed attempts to link nearest neighbours with random points are a waste of computational resources and should be avoided. In [11] it is suggested that a search visibility region or boundary domain (a sphere of certain radius around nodes) is defined to reduce the size of the search space. This was found to improve performance in general but did lead to a trade-off with the additional computational time required to define the search sub-space. Incorporating knowledge of failures as proposed in [8] is an interesting approach which leads to the notion of intelligent sampling.

Non-uniform sampling is an area of research which has much potential to improve the performance of the RRT. The search algorithm can explore very slowly when the sampling domain is not well adapted to the problem [11]. Normally, a point is selected from the search space based upon a uniformly random distribution. However, as shown previously, this approach can be ineffective where narrow passageways exist and the search space has low connectivity.

Rather than uniform random sampling, many researchers have focussed on other sampling methods, some even adaptive. The elegance of the RRT search algorithm is the manner in which randomisation implicitly computes the Voronoi regions created by each node of the tree [7]. The largest regions are more likely to be selected and so the tree is pulled from its root out into the search space [1, 7]. In [10], the Voronoi regions are computed explicitly which results in a more predicable search but at a higher computational cost. By making the probability non-uniform, such as increasing it around the goal as in [13], the rate of convergence of the RRT can be increased. Without biasing, the tree can come very close to the goal but still fail to connect to it. However, such biasing can also lead the RRT to become trapped in local minima during the search. Thus, a trade-off exists as making the algorithm domain specific (by definition) reduces its generality.

Another problem is that RRTs typically do not consider factors such as path length or smoothness during the search. Thus, path quality is usually sub-optimal. In [7] and [14] a path cost is taken into account which biases the tree growth to produce shorter paths while exploring its environment. In [14], after running an initial search, the algorithm recommences using previous tree information to find better solutions. This approach incrementally improves the quality of the solution while time permits. This work is extended in [15] where the performance of RRTs is analysed in dynamic environments. RRTs are shown to work well for three mobile robots in partially unknown environments. In cases of such complexity, and where solutions are required in real-time, the efficiency of algorithms is of even greater importance.

Computational efficiency of search algorithms is generally of significant interest to researchers. Due to their relatively recent introduction, there is still much work being carried out to analyse and improve RRTs. As discussed previously, central problems involve algorithm efficiency (which is heavily correlated with the frequency of nearest-neighbour and collision checking functions), and the problems associated with obstacles, narrow passages and low search space connectivity.

The most expensive steps in the search algorithm are the nearest neighbour and collision checking functions. The cost of nearest neighbour calls is one of the major bottle-necks in the performance of sampling-based motion-planning algorithms such as the RRT. Therefore, it is crucial to develop efficient techniques for nearest neighbour searching [6]. Normally, nearest neighbour searches involve calculating the squared distance between two or more different points and comparing these measurements to find the minimum. Despite this being a straight-forward calculation, when a very large number of nodes must be searched, this process can take a significant amount of time. Inexact methods may be considered in cases where time or computational resources are restricted. However, it is foreseeable that approximate nearest neighbour searches, whilst providing greater efficiency, may lead to poorer solutions or not finding paths at all.

Multiple RRTs is a solution that has potential to provide greater robustness across environments of varying complexity. Some researchers have considered this idea such as in [5] and [9]. Maintaining a number of RRTs makes possible the use of previous learned knowledge [5], which is normally wasted if a collision occurs. Multiple RRTs are created in [5] to form a forest of trees which can be merged, split and pruned. The focus of this work was how to manage the forest of RRTs. By contrast, this paper shows how multiple trees may lead to faster convergence and prove advantageous where narrow passages exist. Multiple RRTs are also studied in [9] where many important issues are identified. Factors to consider include; when to create a local tree, how often the local trees should be allowed to grow, and when to look for inter-tree connections. Consideration of these issues and ways to address them has prompted the experiments carried out in this paper.

This section covered many of the major contributions made towards the RRT up to now. The above discussion also highlights many of the problems faced when implementing RRTs. The following section will describe the experimental procedure used to evaluate the performance of multiple RRTs as a means of improving computational efficiency in search spaces of high obstacle density.

III. Experimental Procedure

This section describes the experiments carried out to measure the efficacy of multiple RRTs. The work presented in this paper proposes a connection policy which ensures high computational efficiency is maintained despite the presence of numerous trees. A simple but effective strategy is implemented where new trees are permitted to grow whenever a point is not able to be connected to an existing tree. The pseudo-code of the proposed Multi-RRT algorithm is shown in Figure 4. Whenever a collision occurs a new tree is initialised. Inter-tree connections are attempted to all trees at each step, however only the nearest neighbour of each tree is considered.

Multi-RRT Pseudo-code:

```
do while path not found
    NEW_POINT
    for each tree
        NEAREST_NEIGHBOUR
        COLLISION_CHECK
        if no collision
            CONNECT
        else
            NEW_TREE
        end if
    end for
end do
```

Figure 4. The proposed new Multi-RRT algorithm.

In order to keep computational efficiency close to optimal, only the nearest neighbour (rather than every node) of each individual tree is considered for connection. This is important because attempts to connect to a greater number of nodes would lead to a large increase in the amount of computationally expensive collision checks required.

Another key advantage of the proposed Multi-RRT algorithm is that new trees are only created as required. The number of trees employed is adjusted dynamically and is determined automatically depending on the connectivity of the environment. Search spaces that have no or few obstacles will have fewer collisions and thus a lower number of trees will be created during the search. By contrast, additional trees will be generated when collisions occur in environments exhibiting high obstacle densities or when regions are isolated by narrow passageways.

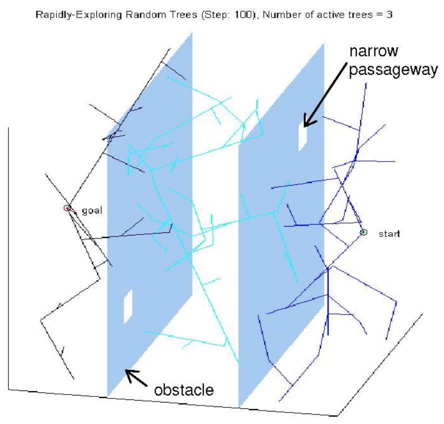
While a limit on the maximum number of trees could be imposed, the search environment does this implicitly. For example, Figure 5 depicts an environment with two large obstacles. The search space is effectively divided into three possible exploration regions, which are connected together by only two small windows. After 100 search iterations it can be seen in Figure 5 (a) that trees have begun to grow in each of the sub-sections.

Each time a new point is selected, the respective nearest neighbours belonging to each tree in existence will be determined. At this stage of the search, every new point is able to connect to at least one of the three trees and there is no need for any additional trees. Thus, although no explicit restriction is placed on the algorithm with respect to the maximum number of trees possible, no more than what is necessary is created. In this way, multiple RRTs can be implemented without introducing additional parameters which may require tuning; ensuring the generality of the algorithm and simplicity of its application are preserved.

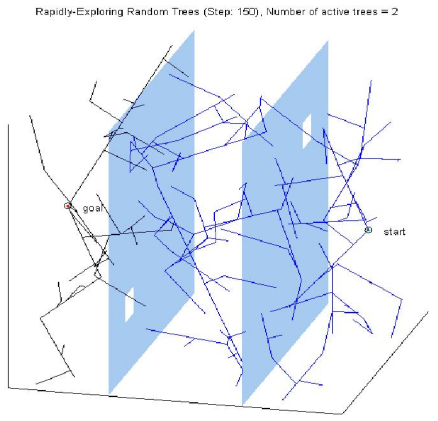
Additionally, when a connection between two or more trees is found, the trees are merged, helping to manage the total number of trees. This event occurs at step 150 in Figure 5 (b) where the two trees on the left become one. Similarly, in Figure 5 (c) the final two trees are united to form one tree, and hence a complete path from the starting point to the goal is found.

The performance of the proposed Multi-RRT strategy is evaluated with respect to the standard RRT-Connect algorithm. RRT-Connect is limited to a maximum of two trees, one rooted at the start and one at the goal. This is considered as the benchmark against which the proposed Multi-RRT is compared. The Multi-RRT algorithm has no limit to the number of trees it may create and is free to multiply depending on its environment.

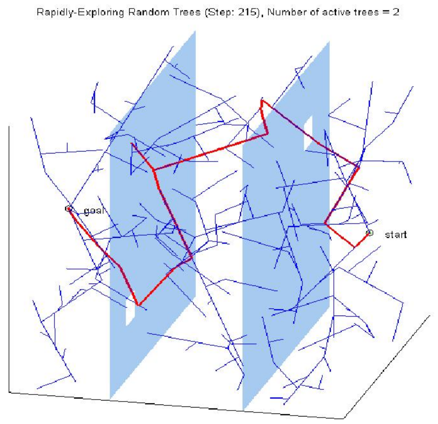
An experiment is conducted to compare the efficiency of the two algorithms in search spaces of increasing complexity. As the number of obstacles is increased, each search space exhibits progressively decreasing connectivity. The number of obstacles ranges from two to fifteen, providing fourteen different testing environments. One obstacle is not implemented since both algorithms would simply grow two trees and the results of each method would be the same. In the first environment, two walls are positioned to divide the search space into three equal sections. Each subsequent search space has a wall added up to the maximum of fifteen. For each environment the obstacles are positioned such that the search space is equally divided.


Rapidly-Exploring Random Trees (Step: 100), Number of active trees = 3
narrow passageway
goal
start
obstacle

(a)


Rapidly-Exploring Random Trees (Step: 150), Number of active trees = 2
goal
start

(b)


Rapidly-Exploring Random Trees (Step: 215), Number of active trees = 2
goal
start

(c)

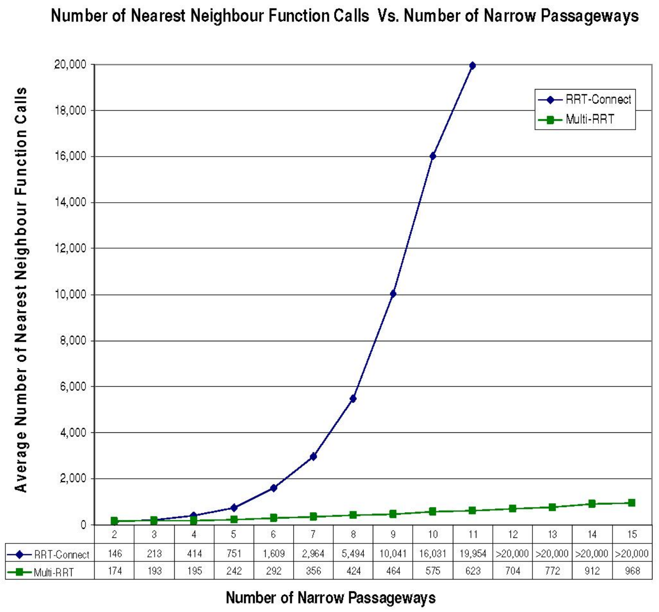
Figure 5. An example of multiple RRTs path planning in the presence of two obstacles and two narrow passageways. In (a), three trees are exploring the environment. In (b), two of them have connected with each other. In (c), a completed path from the start point to the goal has been discovered.

Small windows exist through which the planner must pass in order to connect the start and goal points. Although obstacle data is necessary for the algorithm to perform collision checks, no attempt is made to process this data to extract information. This reduces the computational expense of the search and makes the RRT appropriate for use as a single-query planner.

In this experiment, one hundred trial runs of each of the two algorithms, in each of the fourteen different search spaces are performed. Several statistics are measured in order to compare the Multi-RRT and RRT-Connect approaches. The respective number of collision checks, nearest-neighbour searches, and search iterations are taken as proxies for evaluating the efficiency of the two algorithms. The number of search iterations is limited to a maximum of 20,000 due to time constraints. The following section presents the findings of these experiments.

IV. Results

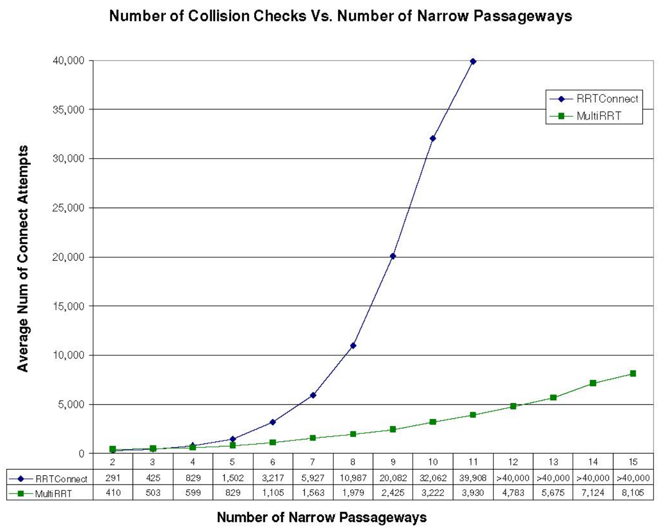
The experimental results confirm the hypothesis that multiple RRTs prove advantageous in environments that have many narrow passageways. The results of the average number of nearest neighbour function calls versus the number of narrow passageways present in the search space for both algorithms are shown in Figure 6.



	2	3	4	5	6	7	8	9	10	11	12	13	14	15
◆ RRT-Connect	146	213	414	751	1,609	2,964	5,494	10,041	16,031	19,954	>20,000	>20,000	>20,000	>20,000
■ Multi-RRT	174	193	195	242	292	356	424	464	575	623	704	772	912	968

Figure 6. Graph showing the average number of nearest neighbour function calls versus the number of narrow passageways. The number of function calls made by RRT-Connect grows exponentially as additional obstacles are added while the performance of the Multi-RRT is linear.

It is clear from the graph that the Multi-RRT method results in significantly fewer nearest neighbour calculations. When the number of narrow passageways is greater than 12 the RRT-Connect method fails to find a path within the maximum number of search iterations provided. In stark contrast, the Multi-RRT is consistently able to find solutions irrespective of the number of obstacles. The presence of many narrow passageways does not pose a considerable hindrance to multiple RRTs. The number of search iterations is also equal to the number of nearest neighbour checks. It is evident that fewer search iterations are required to find paths.



	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RRTConnect	291	425	829	1,502	3,217	5,927	10,987	20,082	32,062	39,908	>40,000	>40,000	>40,000	>40,000
MultiRRT	410	503	599	829	1,105	1,563	1,979	2,425	3,222	3,930	4,783	5,675	7,124	8,105

Figure 7. Graph showing the average number of collision checks versus the number of narrow passageways. Once again, the number of function calls made by RRT-Connect grows exponentially as connectivity of the search space is reduced while the performance of the Multi-RRT is superior.

From Figure 7 it is obvious that as the number of narrow passageways is raised, the performance of the multiple RRT becomes increasingly superior to the RRT-Connect method on the basis of computational efficiency. More significant however, is the ability of the Multi-RRT algorithm to find paths when RRT-Connect cannot. A key outcome of these experiments is that not only is the Multi-RRT method more efficient, it is actually better at finding solutions, especially in environments containing many obstacles. Since the Multi-RRT is completing fewer search iterations, and the average number of collision checks is also reduced, the convergence rate is improved.

V. Conclusions and Future Work

The connection policy presented in this paper facilitates the use of multiple Rapidly-exploring Random Trees in path planning. The approach is simple to implement and in environments with no obstacles the algorithm functions as a normal RRT. In the presence of obstacles and when new points are unable to connect to an existing tree, a new search tree will be invoked. Thus, this method ensures the highly desirable parameter-free nature of the RRT is preserved.

These experiments demonstrate that the rate of convergence can be greatly improved using multiple RRTs. Furthermore, when compared to the standard RRT algorithm, the Multi-RRT approach exhibits greater computational efficiency, most notably in environments with high obstacle densities.

It would be worthwhile extending these experiments by including a greater variety of environments. In addition, another advantage of multiple RRTs is there ability to explore without being restricted to regions nearest the start and goal points. This may prove especially beneficial when path planning involves multiple goals.

Acknowledgment

This work is supported in part by the ARC Centre of Excellence program, funded by the Australian Research Council (ARC) and the New South Wales State Government, Australia. The authors also wish to acknowledge the support of the University of Technology, Sydney and the RTA of NSW.

References

[1] LaValle, S. M., 'Rapidly-Exploring Random Trees: A New Tool for Path Planning', TR 98-11, Computer Science Department, Iowa State University, http://janowiec.cs.iastate.edu/papers/rrt.ps, Oct. 1998.
[2] Kuffner, J. J., LaValle, S. M., 'RRT-Connect: An Efficient Approach to Single-Query Path Planning', Proceedings IEEE International Conference on Robotics and Automation (ICRA 2000), San Francisco, CA, 2000.
[3] LaValle, S. M., Kuffner, J. J., 'Rapidly-Exploring Random Trees: Progress and Prospects', In Proceedings Workshop on the Algorithmic Functions of Robotics, 2000.
[4] LaValle, S. M., Kuffner, J. J., 'Randomized Kinodynamic Planning', Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on, Pages 473-479 Volume 1, 1999.
[5] Li, T., Shie, Y., 'An Incremental Learning Approach to Motion Planning with Roadmap Management', Proceedings. ICRA 2002. IEEE International Conference on Robotics and Automation, Washington, DC, Pages 3411-3416, Volume 4, 2002.
[6] Yershova, A., LaValle, S. M., 'Improving Motion-Planning Algorithms by Efficient Nearest-Neighbor Searching', Robotics, IEEE Transactions on, Pages 151-157, Volume 23, No. 1, February 2007.
[7] Urmson, C., Simmons, R., 'Approaches for heuristically biasing RRT growth', Intelligent Robots and Systems. Proceedings. 2003 IEEE/RSJ International Conference on, Pages 1178-1183, Volume 2, 2003.
[8] Kavralu, L. E., Svestka P., Latombe J. C., Overmars, M. H., 'Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces', Robotics and Automation, IEEE Transactions on, Pages 556-580, Volume 12, No. 4, August 1996.
[9] Strandberg, M., 'Augmenting RRT-planners with local trees', Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on, Pages 3258-3262, Volume 4, 2004.
[10] Lindemann, S. R., LaValle, S. M., 'Incrementally Reducing Dispersion by Increasing Voronoi Bias in RRTs', Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on, Pages 3251-3257 Volume 4, 2004.
[11] Yershova, A., Jaillet, L., Simeon, T., LaValle, S.M., 'Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain', Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, Pages 3856-3861, 2005.
[12] Miyazawa, K., Maeda, Y., Arai, T., 'Planning of Graspless Manipulation Based on Rapidly-Exploring Random Trees', Assembly and Task Planning: From Nano to Macro Assembly and Manufacturing, 2005. (ISATP 2005). The 6th IEEE International Symposium on, Pages 7-12, 2005
[13] Bertram, D., Kuffner, J., Dillmann, R., Asfour, T., 'An Integrated Approach to Inverse Kinematics and Path Planning for Redundant Manipulators', Robotics and Automation, 2006. Proceedings 2006 IEEE International Conference on, Pages1874-1879, 2006.
[14] Ferguson, D., Stentz, A., 'Anytime RRTs', Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, Pages 5369-5375, 2006.
[15] Ferguson, D., Stentz, A., 'Anytime, Dynamic Planning in High-dimensional Search Spaces', Robotics and Automation, 2007 IEEE International Conference on, Pages 1310-1315, 2007.