MEDIAL-AXIS BIASED RAPIDLY-EXPLORING RANDOM TREES

A Senior Scholars Thesis

by

EVAN JOHN GRECO

Submitted to Honors and Undergraduate Research
Texas A&M University
in partial fulfillment of the requirements for the designation as

UNDERGRADUATE RESEARCH SCHOLAR

May 2012

Major: Computer Science

MEDIAL-AXIS BIASED RAPIDLY-EXPLORING RANDOM TREES

A Senior Scholars Thesis

by

EVAN JOHN GRECO

Submitted to Honors and Undergraduate Research
Texas A&M University
in partial fulfillment of the requirements for the designation as

UNDERGRADUATE RESEARCH SCHOLAR

Approved by:

Research Advisor:	Nancy M. Amato
Associate Director, Honors and Undergraduate Research:	Duncan MacKenzie

May 2012

Major: Computer Science

ABSTRACT

Medial Axis-Biased Rapidly-Exploring Random Trees. (May 2012)

Evan John Greco
Department of Computer Science and Engineering
Texas A&M University

Research Advisor: Dr. Nancy M. Amato
Department of Computer Science and Engineering

RRTs (Rapidly-Exploring Random Trees) have shown wide applications in robotics. RRTs are a type of sampling-based motion planners that expand to fill the space starting from one or more root configurations. RRTs are excellent at rapidly exploring open space in an environment, as well as finding configurations close to obstacles. PRMs (Probabilistic RoadMap methods) are another class of sampling-based motion planners. One particular planner, Medial Axis PRM (MAPRM), constructs roadmaps on the medial axis, leading to paths with high clearance. This work introduces a novel RRT variant, namely the Medial Axis RRT (MARRT) that constructs trees whose nodes and edges lie on (or near) the medial axis of the free configurations space. This is achieved through the use of MAPRM-like techniques to retract sampled configurations to the medial axis of the free space. We show MARRT successfully increases clearance along RRT paths for a broad spectrum of motion planning problems.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

Motion Planning is a known difficulty in robotics involving planning the paths of robots through many different types of environments. These environments may be a workspace for manufacturing robots, a disaster area in search and rescue, or even energy landscapes in protein folding. Applications of Motion Planning include virtual reality (VR), protein folding, multi-agent systems, manufacturing, prototyping, and computer aided design (CAD), among others.

More precisely, Motion Planning (MP) algorithms address the problem of finding a sequence of valid (collision-free) states (configurations) that take a moving object, referred to as a robot, from an initial configuration (C_{start}) to a goal configuration (C_{goal}). Deterministic calculation of the configuration space of an environment is known to be P-Space Hard [1] and generally infeasible, except for low DOF problems, e.g., $DOF < 5$.

The complexity of motion planning led to the introduction of sampling-based motion planning. Sampling-based planners sample the environment for valid configurations, and may bias these configurations based on certain parameters or algorithms [2][3]. Within this class of samplers lies tree-based and graph-based planners. Tree-based planners generally involve starting at some configuration C_{root} and incrementally growing towards some goal configuration C_{goal}, with each root growing it's own connected component (CC). One of the most common examples of a tree-based planner is the Rapidly-Exploring Random Tree (RRT) [4]. RRTs are useful for single-query

This thesis follows the style of the *IEEE Transactions on Robotics and Automation.*

problems in simple environments, where the goal is to explore free space efficiently to find a goal in real-time applications. There are several RRT-based methods to solving motion planning problems, including RRT-Connect [4] and OBRRT [5]. Graph-based planners, such as the Probabilistic Roadmap Method (PRM) [6], randomly sample the environment and create a graph that can be queried for paths from one location to another, usually using a shortest-path algorithm such as Dijkstra's Single Source Shortest Path Algorithm.

In general, sampling-based planners have difficulty creating roadmaps with high clearance. A PRM-based method that addresses this problem is the Medial-Axis PRM (MAPRM [3]). MAPRM randomly samples the environment and retracts the sampled configurations to the medial axis of the free space. This gives a roadmap with high clearance from obstacles, which as previously mentioned can be an important characteristic for paths. However, there is no such RRT-based method which fully grows on the medial-axis.

The primary contribution of this work is a novel RRT variant, the Medial-Axis Biased Rapidly-Exploring Random Tree (MARRT). MARRT retracts RRT nodes to the medial axis of the free space, along with the connections between them using the Medial Axis Local Planner (MALP [7]). In low dimensions, samples can be transformed to the Medial Axis at a low cost, while in higher dimensions we use an approximate method. In summation, the tree and its edges are all be on, or near, the medial axis, and grow in the fashion of an RRT. The goal of MARRT is not necessarily to provide the most efficient planning algorithm, but to establish the feasibility of a medial-axis biased RRT planner. A summary of the primary contributions of this paper are as follows:

- Introduction of a novel method, MARRT, that successfully grows RRTs on the medial axis of the free space.
- Detailed experimental evaluation in 2D and 3D environments with robots with DOF varying from 2 to 6.
- Analysis of MARRT roadmap performance and evaluation of clearance-related data compared to other common RRT-based planners such as RRT [8] and OBRRT [5]

In order to analyze the performance of MARRT, metrics such as path length, average clearance, node count, and others are detailed. The experiments are designed to measure general roadmap characterics (e.g., clearance) in the absence of a query and query-based scenarios, with both start and goal configurations being given. The environments themselves contain complicated narrow passages that many planners have trouble navigating, as well as open spaces that can also be effectively mapped using the medial axis. Qualitative analysis is done by utilizing visualizations of 2D experiments. The results show a clear advantage produced by MARRT in its ability to create paths that maximize clearance.

CHAPTER II

PRELIMINARIES

In this section, some basics of motion planning will be explained, along with the two algorithms that inspire MARRT: RRT and MAPRM.

A. Configuration space

Configuration Space (C_{space}) is the space that includes all poses and positions of a particular robot subject to environmental constraints. Each point in the C_{space} corresponds to a configuration of the robot. C_{space} is split into three primary subsets: C_{free}, C_{obs}, and $C_{contact}$. All valid configurations are $\in C_{free}$, while configurations with one or more dimensions partially or completely inside of an obstacle are $\in C_{obst}$. Contact configurations, particularly useful in cases such as when robotic manipulators make contact with objects (holding a glass, picking up a package, soldering points, etc.), occur where configurations and obstacles touch.

Sampling-based motion planning came to be after it was shown that explicitly calculating the C-Space of an environment is P-Space Hard[1]. To address this problem, sampling-based motion planning was developed.

B. Sampling-based motion planning

Sampling-based planners are particularly useful in motion planning. In sampling-based planners, different algorithms use different metrics and methods in order to bias samples in a way that facilitates the mapping of the workspace. In general, random configurations are sampled in the environment. Various methods have been developed to filter [9][10] or retract [2][3] samples to bias sampling towards different area of C_{free}.

As described in the introduction, the primary basis of MARRT is the Rapidly-Exploring Random Tree (RRT [8], Figure 1) Algorithm 1 describes the basic idea of how RRTs explore free space. For a given number of iterations, RRTs randomly sample a configuration in the workspace, and extend the tree rooted at q_{start} a distance δ towards the randomly sampled node. [8] describes the introduction of RRT-based algorithms to motion planning, as well as theoretical analysis that details the usefulness of applying RRTs to the realm of motion planning. The authors formulate theoretical and experimental results for 4 different types of motion planning problems: holonomic, non-holonomic, kinodynamic, and closed kinematic chains.

The primary function of the RRT is to incrementally randomly explore space. Depending on the δ value, RRTs can expand large distances efficiently in C_{free}. Particularly applicable to motion planning is RRT-Connect [4] which grows 2 trees: one from C_{start} and one from C_{goal}. However, we adjust MARRT from the basic RRT algorithm.

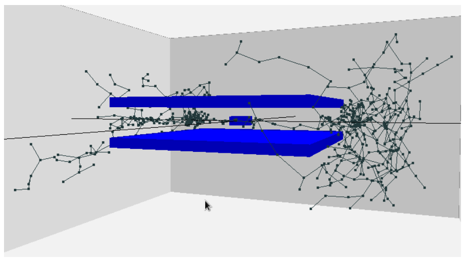
Fig. 1. RRT example.

Algorithm 1 Rapidly-Exploring Random Tree

```
Input: Environment $e$, Start Configuration $q_{start}$, Step Size $\delta$, Num Iterations $n$
Output: Roadmap $R$
  $R.insert(q_{start}$
  for $i = 1...n$ do
    $q_{rand} \leftarrow GetRandomCfg(e)$
    $q_{new} \leftarrow NearestNeighbor(R, q_{rand})$
    $ExtendTowardNode(q_{new}, q_{rand}, \delta)$
    $AddToRoadmap(q_{new})$
  end for
  return $R$
```

An additional study done by Kuffner et al. [11] introduces the concept of RRT-Connect. RRT-Connect is a bidirectional planner as discussed in [8], comprising of two trees, with one beginning at q_{start} and one at X_{goal}. Each tree grows towards each other using a greedy heuristic, with a connection between the two being attempted at each step. Once the two trees meet (i.e. a connection between a node in the q_{start} tree and a node in the q_{goal} tree is created), a path can be derived from the tree using a simple path finding algorithm. Additionaly, some analysis revealed that RRTs are indeed probabilistically complete, i.e. probability of finding a path approaches 1.

Another RRT-based sampler, OBRRT [5] exploits information gained about obstacles in order to bias the growth of the tree. Influenced by OBPRM[2], OBRRT incrementally chooses growth methods based on user-provided weights, and grows based on these methods. These methods include constructing vectors from randomly sampled configurations, or to randomly choose vectors based on workspace obstacles and then choosing to randomize orientation or position, among others.

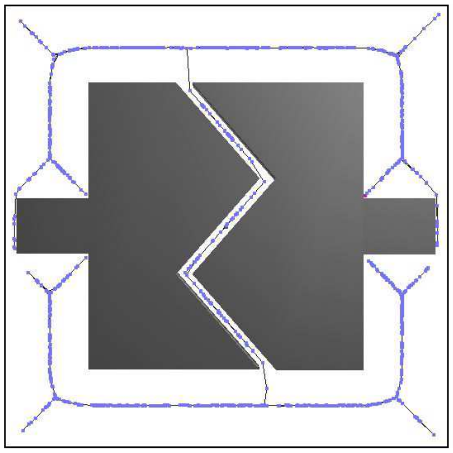

Fig. 2. MAPRM example.

Algorithm 2 Medial Axis PRM

Input: Environment e

Output: Roadmap R

```
  $done = false$
  **while** $!done$ **do**
    $q_{curr} \leftarrow GetRandomCfg(e)$
    $PushToMedialAxis(q_{curr})$
    $AddToRoadmap(q_{curr})$
    $Connect(R, q_{curr}, ...)$
  **end while**
  **return** $R$
```

There have been several techniques proposed that utilize the medial axis for motion planning. This can be desireable since the medial axis maximizes clearance from obstacles and hence can contain ‘safe’ paths. One of the first is the Medial-Axis PRM

(MAPRM) [3]. MAPRM allows nodes to be on the medial axis without its explicit computation. In particular, as shown in Algorithm 2 and Figure 2, in MAPRM, a random configuration is sampled, and then it is pushed to the medial axis of the free space. As with any PRM variant, MAPRM can create multiple connected components, and requires methods to connect them together if a fully-connected roadmap is desired. The expense of this algorithm is dominated by the $PushToMedialAxis()$ function, especially in higher dimensions where an approximate version is required. The expense of this function comes primarily due to the cost of collision detection calls, of which many are required in order to locate a point sufficiently close to the medial axis.

[12] presents details on MAPRM for a three-dimensional free-flying rigid body, outlining that the algorithm is theoretically guaranteed to sample more nodes in a narrow passage than uniform random sampling. Experimentally, the work displays a large advantage over uniform sampling.

In both [3] and [12], sampling is limited to $\mathbb{R}^d$, $d \leq 3$ space, as exact computation of translating a random configuration to the medial axis is feasible. This is not true for higher dimensions. This led to the development of a general framework for MAPRM [13] allowing for an approximation of the medial axis for higher-dimension problems ($\mathbb{R}^d$, $d \geq 4$).

C. Local planners

In PRMs, roadmap edges correspond to, typically valid, trajectories connecting the start and the end configuration of the edge. These trajectories are typically validated using some simple, deterministic planner referred to as a local planner (LP). The most common local planner used in PRMs is the *straight line* local planner. In the straight-line LP, the intermediate configurations at some problem dependent resolution along the straight line in configuration space connecting the start and the end points of the edge are all tested for validity; if they are all valid then the edge is determined to be valid and an edge is added to the roadmap representing that connection. The straight-line LP is simple and convenient. However, it is not sufficient for our uses since we are interested in paths that lie on the medial axis.

In this work we will use the recently introduced medial axis LP (MALP) [7] in which as the edge itself is pushed to medial axis. MALP works by recursively splitting a straight line connection in half, and pushing the midpoint of each bisected straight line to the medial axis. The recursion stops when the vertices and edges are all within some user provided threshold destance of the medial axis. As shown in [7], the cost of MALP can vary greatly depending on the level of accuracy desired. From our experience, the desired accuracy is typically achieved within 4 or 5 levels of recursion.

CHAPTER III

MEDIAL AXIS RRT

In this section we describe Medial Axis RRT, or MARRT, which is a variant of the standard RRT algorithm that grows a tree on the medial axis from a specified initial configuration. There is also a variant MARRT-Connect that is analogous to the RRT-Connect algorithm which specifies both a start and a goal configuration and attempts to grow the tree from the start until it can be connected to the goal.

Algorithm 3 Medial Axis RRT

```
Input: Environment $e$, Local Planner $lp$, Start Configuration $q_{start}$, Step Size $\delta$,
  Max Iterations $n$
Output: Roadmap $R$
  $R.insert(q_{start})$
  for $i = 1...n$ do
    $q_{curr} \leftarrow GetRandomCfg(e)$
    $q_{nearest} \leftarrow GetNearestNeighbor(R)$
    $q_{curr} \leftarrow getRRTNode(e, R, \delta)$ //with respect to step size
    $PushToMedialAxis(q_{curr})$
    $lp.Connect(R, q_{curr}, e, ...)$
  end for
  return $R$
```

MARRT, in Algorithm 3, begins similarly to the other related algorithms – the start configuration q_{start} is added to the roadmap and will serve as the root of the tree. Inside the main loop, a random configuration is sampled. As with MAPRM, this node can be in q_{free} or q_{obst}.

After the node is sampled, the nearest neighbor in the tree must be found based on some distance metric (e.g. Euclidean). There are several variables in determining what distance metric is sufficient for which application [14]. This is the same as is done in standard RRT.

Next, the newly sampled node is moved to within a user-defined distance δ away from the closest neighbor in the tree rooted by q_{start}. In RRT, this node is required to be in q_{free} and is the final expansion step. In MARRT, the expansion is done in the same way, except that after the expansion the node is then pushed to the medial axis. The choice of δ is problem dependent. As discussed later, if the trajectory consists of long straight-line portions, then larger values of δ may be useful. Indeed, given the nature of the medial axis which maximizes clearance, MARRT may be able to use larger values of δ than other RRT variants.

In MARRT, as in MAPRM, $PushToMedialAxis()$ is the main step in creating a roadmap with nodes on the medial axis. As previously mentioned, $PushToMedialAxis()$ can be done in an exact fashion assuming the robot's DOF is less or equal to 3. After the configuration is pushed to the medial-axis of the free space, a connection will need to be made to it from the tree. This will be obtained by using a local planner. In the case of MARRT, the use of the Medial Axis LP [7] will be the default choice. MALP modifies the edges between nodes in the roadmap such that this path lies on the medial-axis. By having the nodes as well as the edges on the medial axis, every entity of the roadmap will have high clearance, as it is one of the primary goals of MARRT.

Algorithm 4 Medial Axis RRT-Connect

```
Input: Environment $e$, Local Planner $lp$, Start Configuration $q_{start}$, Goal Configuration $q_{goal}$, Step Size $\delta$
Output: Roadmap $R$
  $R.insert(C_{start})$
  $done = false$
  while !$done$ do
    $q_{curr} \leftarrow GetRandomCfg(e, robot)$
    $q_{nearest} \leftarrow GetNearestNeighbor(R)$
    $q_{curr} \leftarrow getRRTNode(e, R, \delta)$ //with respect to step size
    $PushToMedialAxis(q_{curr})$
    $lp.Connect(R, q_{curr}, e, ...)$
    if $lp.Connect(R, q_{goal}, e, ...)$ then
      $done = true$
    end if
  end while
  return $R$
```

We also define another version of the algorithm, MARRT-Connect (Algorithm 4), that is similar to RRT-Connect and is designed to grow the tree from the start configuration q_{start} to a goal configuration q_{goal}. In this case, we initially add both q_{start} and q_{goal} to the roadmap. Then, once the newly expanded node is added to the roadmap, a connection from the roadmap to q_{goal} is attempted in order to find a path from q_{start} to q_{goal}.

CHAPTER IV

RESULTS

In this section, experimental setup, results, and analysis are provided.

A. Setup

MARRT was implemented in C++ in the Probabilistic Motion Planning Library (PMPL), developed in the Parasol Lab at Texas A&M University. PQP [15] is used for Collision Detection. The experiments themselves ran on a cluster consisting of 24 IBM x335 servers with 4GB RAM and (2) 2.4GHz Intel Xeon CPUs each. The operating system for the cluster is CentOS 5.4, with Rocks 5.0 as the clustering software. Visualization is done with Vizmo [16], an in-house developed tool developed by Parasol Lab that displays sampled configurations, bounding boxes, paths, and other useful information pertaining to motion planning. Results were averaged over 10 random seeds.

When considering the effectiveness of MARRT, it is important to note that the clearance of the resulting roadmap is the main benefit of the algorithm. With this in mind, there are several clearance-based metrics of note:

- **Minimum Path Clearance:** When navigating environments that may be dynamic or approximated, having a small chance of an unexpected collision is important. RRT paths should have low path clearance, while MARRT should have high path clearance.

- **Average Roadmap/Path Clearance:** In addition to knowing the minimum clearance, knowing the average is a good way to see just how safe the roadmap is as a whole in terms of avoiding collisions. As a whole we compare the entire tree's ability to retain clearance for MARRT.

In addition to these metrics, standard motion planning performance metrics including collision detection (CD) call totals and roadmap node count are reported for problems involving queries.

B. Environments

The experiments are split into 2 main sections: 2D and 3D environments. The 2D environments are meant to provide a qualitative analysis on how the different RRTs grow, and how they perform when presented with narrow passages. In the 2D analysis, all methods are given a δ value of 5% of the environment's resolution, based on a given distance metric (in all cases in this paper, standard euclidean). In the 3D environments, an emphasis is given on the RRTs solving queries. To enable better performance in all planners, a maximum δ value is used (that is, at each iteration, the algorithms are allowed to be as greedy as possible when attempting to expand).

The 2D environments include S-Tunnel (Figure 3), 2D Maze (Figure 8), and 2D Z-Tunnel (Figure 9). S-Tunnel presents a 2-DOF robot that needs to navigate a winding narrow passage in order to complete the query. The 2D Maze presents a relatively complicated maze that has 2 solutions. The beginning node starts in the very middle of the maze, which allows for an interesting view into how the various RRT-based methods perform in such an environment over a set number of iterations. The 2D Z-Tunnel environment provides an in-between in difficulty compared to the other two 2D environments, in that it has a z-shaped narrow passage while also forcing the methods to navigate additional corners in order to solve the query.

The 3 3D environments are called 3D Z-Tunnel (Figure 11), 3D Maze (Figure 13), and Flange (Figure 14). The 3D Z-Tunnel presents a 3-Dimensional Z-shaped narrow passage. In order to see how the RRTs are able to navigate while inside the narrow passage, the starting configuration is placed in the center, with the goal configuration at one of the ends of the passage. The 3D Maze environment includes a series of tubes that must be navigated in order to reach the goal. As with Z-tunnel, the starting configuration is placed in the middle of the environment. Finally, the Flange environment simply requires the methods to plan the motion of a tube that is stuck in a constricting obstacle. This environment requires subtle translations and rotations of the tube before it can be extracted from its constricting obstacle. The 3D Maze environment is also used for the 9 DOF serial robot experiment.

C. Results and analysis

The experimental results show a pervasive and consistent increase in overall clearance for MARRT. First, we show and analyze the qualitative results for the 2D experiments as shown in Figure 3, Figure 8, and Figure 9.

1. 2D Environments

The initial experiment involves a qualitative analysis of the coverage of MARRT versus RRT and OBRRT. The experiment consists of 3 separate starting positions with no queries, as we are interested in the manner of the growth of the RRTs. The first position, as shown in Figure 3, is in the bottom left of the environment, while the other two starting positions are in the center and top right of the environment. 2,000 nodes are sampled.

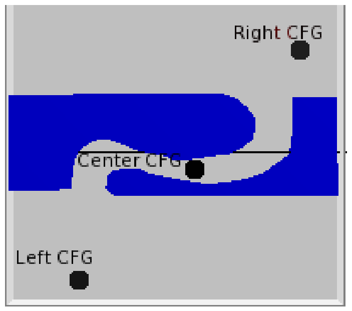

Fig. 3. S-Tunnel with starting CFG points.

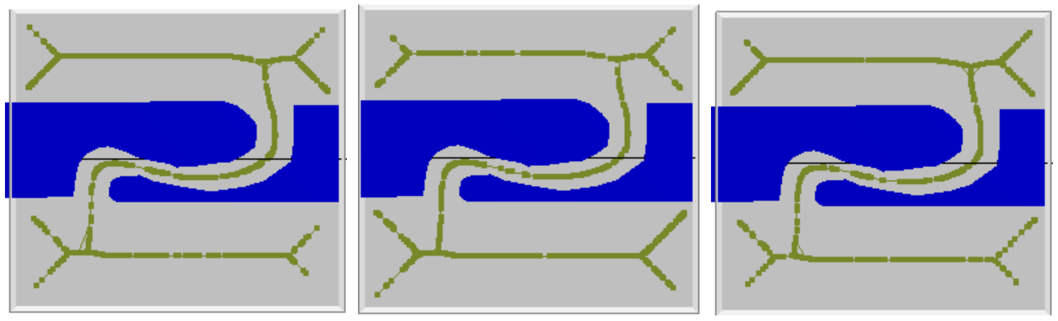

Fig. 4. S-Tunnel MARRT examples: left, center, and right starting positions.

The MARRT graphs, as shown in Figure 4, show a relatively even coverage of the space while maintaining high clearance. The starting position of the tree makes little difference in the trees that are grown, which is in contrast to the performance of the other two methods. This may be explained when considering the high connectivity of nodes on the medial axis, especially in low DOF environments (in this case, only two).

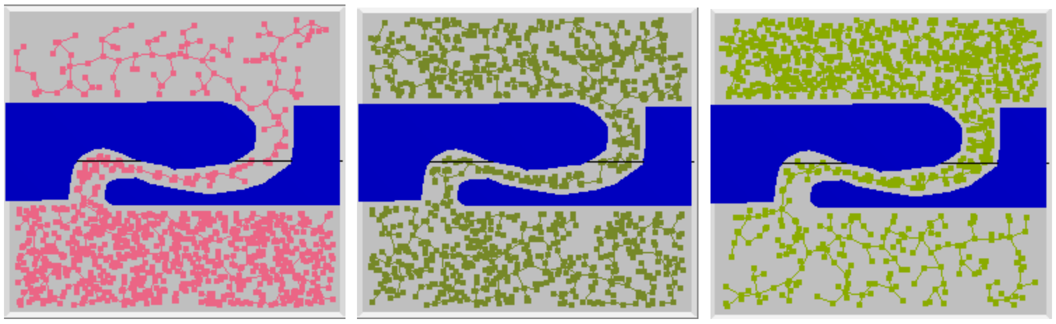

Fig. 5. S-Tunnel RRT examples: left, center, and right starting positions.

RRT, as seen in Figure 5, performance inconsistently when compared to MARRT. RRT grows evenly in the center starting position, with an even distribution of nodes throughout the environment. However, imbalances are visible when the starting configurations are moved to the top right or bottom left of the environment.

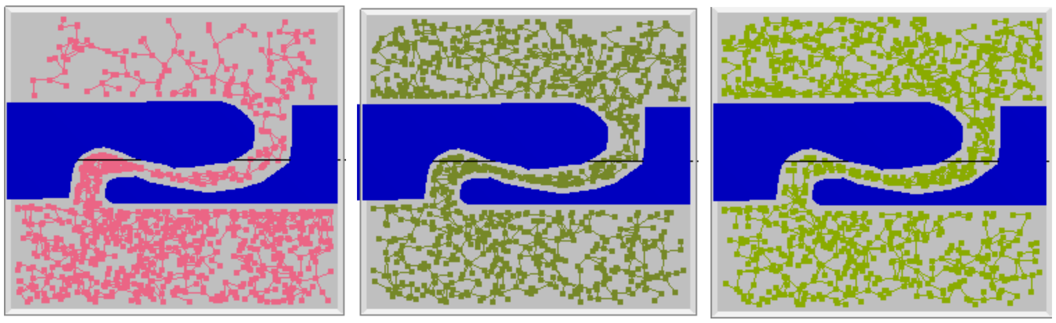

Fig. 6. S-Tunnel OBRRT examples: left, center, and right starting positions.

Similarly to RRT, OBRRT displays an uneven distribution of nodes based on the location of the root configuration, as seen in Figure 6. While not as drastic as RRT, OBRRT fails to achieve the symmetric coverage of MARRT.

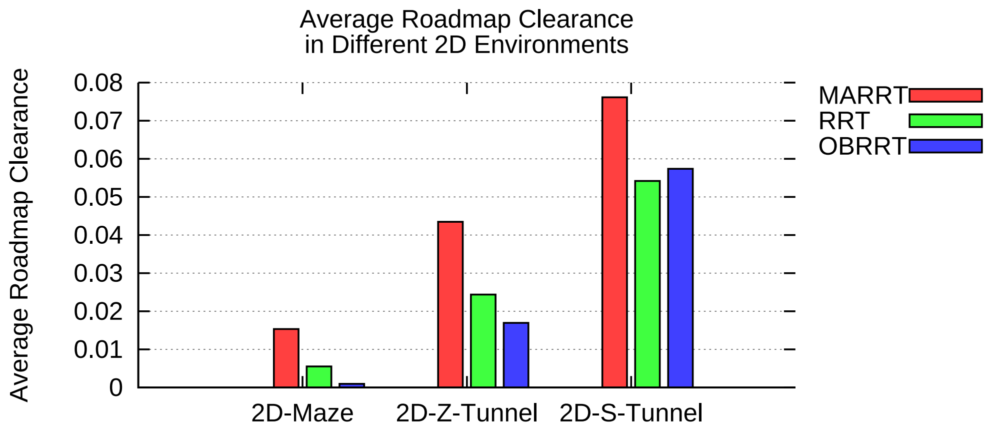

Fig. 7. Average clearance in 2D environments.

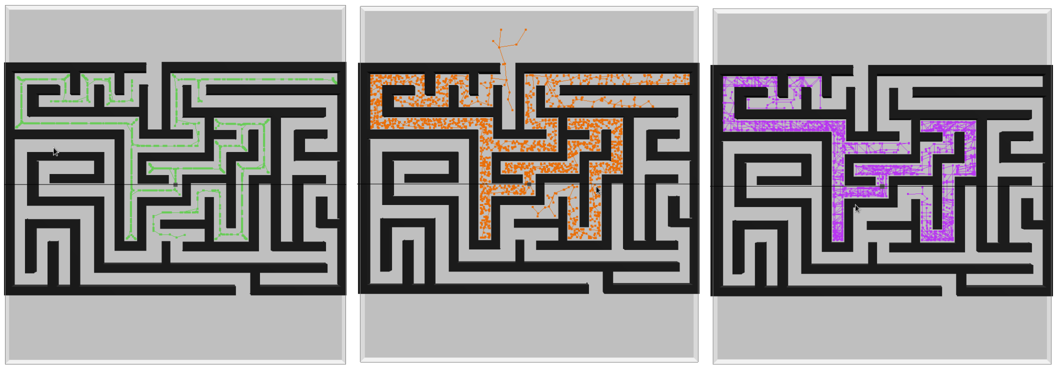

Fig. 8. 2D Maze examples: MARRT, RRT, and OBRRT.

In the 2D Maze environment, we see that, in Figure 8, the general paths are similar. However, the quality of these paths in terms of clearance vary greatly. For MARRT, high clearance is obtained throughout the entire map, while both OBRRT and RRT construct maps that go near the walls of the narrow passages. Overall distance that can be traversed in the workspace favors RRT, but MARRT can traverse near as far, while retaining a high clearance roadmap.

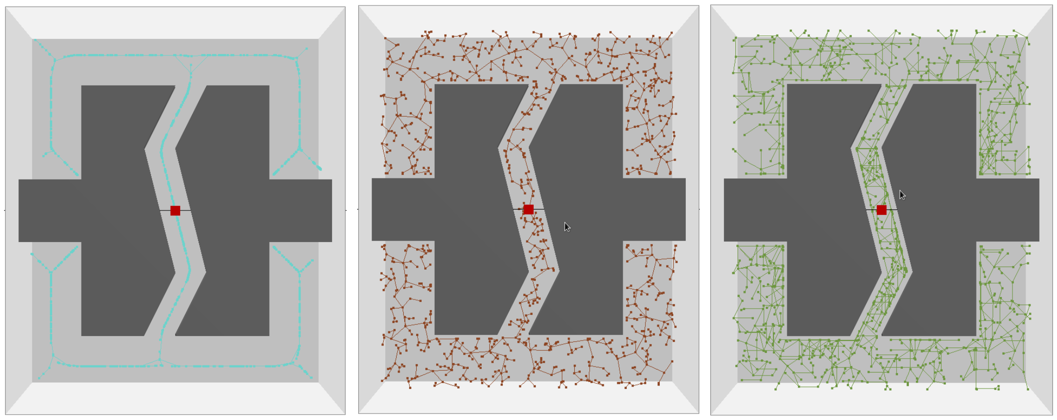

Fig. 9. 2D Z-Tunnel examples: MARRT (left), RRT (center), and OBRRT (right).

In Figure 9, we can see that all 3 methods managed to map the overall workspace. The main difference is in the clearance of the overall roadmaps. MARRT maps the entire available free space, with nodes that have high clearance. RRT and OBRRT have a larger coverage of the free space, but the clearance values are low, as shown in Figure 7

2. *3D (6 DOF) Experiments*

We now move to analyzing the results for the 3D environments. For these environments, a quantitative approach is taken. For each environment, queries are assigned (q_{start} and q_{goal}), and tree generation halts when the trees are able to connect to the goal configuration. The results of the experiments can be viewed in Figure 10 as well as Table I.

In terms of clearance data, MARRT produces higher clearance roadmaps and paths, which is the primary objective of the method.

In the 3D Z-Tunnel environment (Figure 11), the largest discrepancy is in the path length of each method. MARRT produces a path that is several times larger than both RRT and OBRRT. This may be explained by a type of back-tracking that is possible with a medial-axis based algorithm, especially when producing a tree. When sampling in a tight narrow passage, it is possible that the nearest neighbor can change from an out-most branch to an inner branch, resulting in a suction effect that can limit outward growth. One way to address this artifact of the method would be to iteratively smooth and then re-push the path to the medial axis. Another reason the MARRT paths are longer is because paths on the medial axis are inherently longer than those that are allowed to cut corners, which is a natural trade-off between path length and path quality.

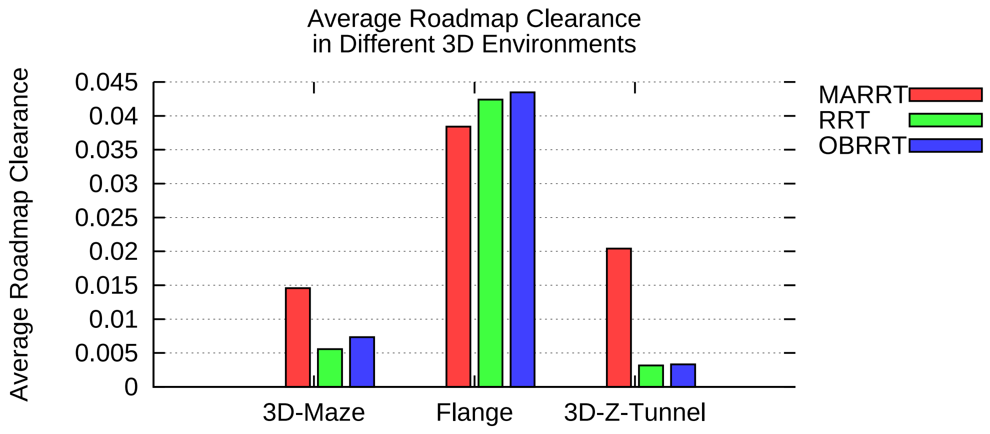

Fig. 10. 3D average clearance data.

3D Maze						
Method	Nodes	Collision Detection Calls	Roadmap Clearance	Path Length	Path Clearance	Clearance Variance
MARRT	1167.1	133099	0.0109656	264.5	0.0145652	2.24012e-05
RRT	1386.6	17881	0.00245215	77.7	0.00557842	1.41861e-05
OBRRT	4104.8	35966.7	0.0015525	64.6	0.0073481	4.94818e-06
Flange						
Method	Nodes	Collision Detection Calls	Roadmap Clearance	Path Length	Path Clearance	Clearance Variance
MARRT	146.8	289510	0.0268746	81.7	0.0384039	0.000697416
RRT	1605	6484.14	0.0522617	56.8571	0.0423998	0.00161294
OBRRT	1115.11	6343	0.031549	53.556	0.0434707	0.0011004
3D Z-Tunnel						
Method	Nodes	Collision Detection Calls	Roadmap Clearance	Path Length	Path Clearance	Clearance Variance
MARRT	647	71617.7	0.00821397	540.1	0.0203978	2.78616e-05
RRT	332.1	6731	0.00133686	108.1	0.00316082	1.18155e-06
OBRRT	92.1	83217.3	0.00132918	138.9	0.00331836	2.40437e-06

Table I

3D environment experimental data. Clearance values are averaged. Clearance variance is the average variance of each roadmap's clearance values.

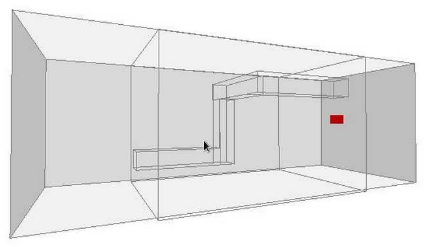

Fig. 11. 3D Z-Tunnel environment.

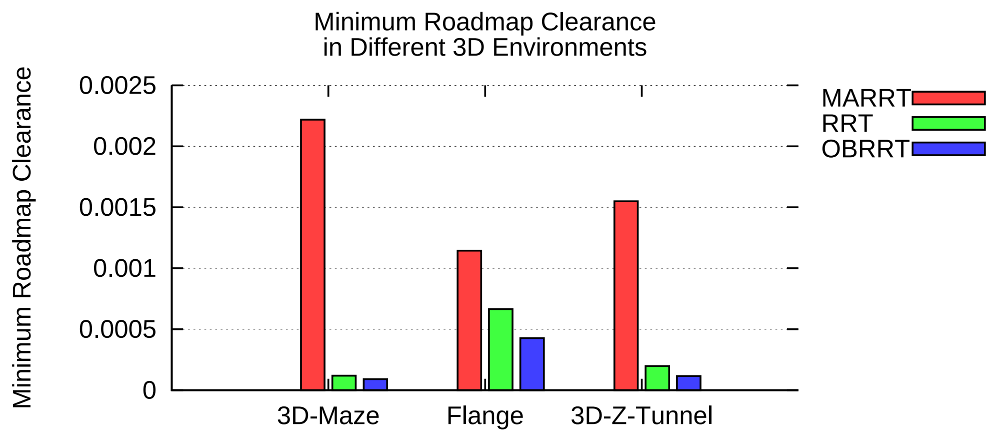

Fig. 12. Minimum clearance for 3D environments.

In the 3D Maze environment (Figure 13, Figure 12), similar results to 3D Z-Tunnel are obtained. In total, the clearance values clearly favor MARRT, while path length follows a similar pattern that was shown in the 3D Z-Tunnel results. MARRT solved the query with fewer nodes than RRT and OBRRT as well, because it took longer steps toward the goal.

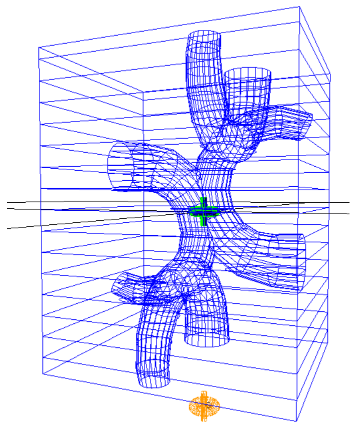

Fig. 13. 3D Maze environment.

The Flange environment (Figure 14) presents a different challenge than the other environments, in that it requires a large object to be removed from a constraining obstacle. This is also the only environment where not all methods solved the problem with 100% efficiency. MARRT performed the best, solving 100% query attempts. OBRRT performed with 90% efficiency, while RRT solved the query with only 70% efficiency. MARRT was also far more efficient in terms of node count with 146.8 nodes on average being required.

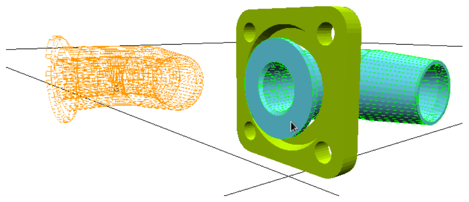

Fig. 14. Flange environment.

3. 3D Maze Serial

3D Maze			
Method	Collision Detection Calls	Roadmap Clearance	Clearance Variance
MARRT	7.86312e06	0.00319565	1.08712e-05
RRT	54503.4	0.00214199	1.52141e-05
OBRRT	69656.2	0.00154734	7.2189e-06

Table II

3D environment serial experimental data. Clearance values are averaged. Clearance variance is the average variance of each roadmap's clearance values.

In order to show the generic nature of MARRT, a 9 DOF experiment was run on the 3D Maze environment. The experiment consists of a robot with 3 rotational joints, 3 rotational degrees of freedom, and 3 translational degrees of freedom, all of which combine to a 9 DOF robot. The experiment does not have a query; the roadmap consists of 5,000 nodes of free growth from the starting configuration, which is located in the center of the environment. The experiments show a qualitative advantage for MARRT, as MARRT is able to navigate through the narrow passage to the open areas above and below. Displayed in Figure 15, MARRT is the only method of the 3 that is able to move out of the medial axis. With the extra degrees of freedom, utilizing the medial axis is especially helpful. In the case of RRT and OBRRT, randomly sampling configurations to expand to becomes more difficult, as the randomization of additional parameters lowers the ability that a configuration in C_{free} may be sampled. While MARRT comes out ahead qualitatively, its largest hindrance is the number of collision detection calls. This is due to the use of MALP in higher degrees of freedom. The experiments were run with 5 random rays per medial axis calculation, which, when combined with MALP and a higher degree of freedom, leads to very high numbers of collision detection calls, visible in Table II. However, the average roadmap clearance is also higher for MARRT, which when combined with the qualitative advantage, gives MARRT an edge.

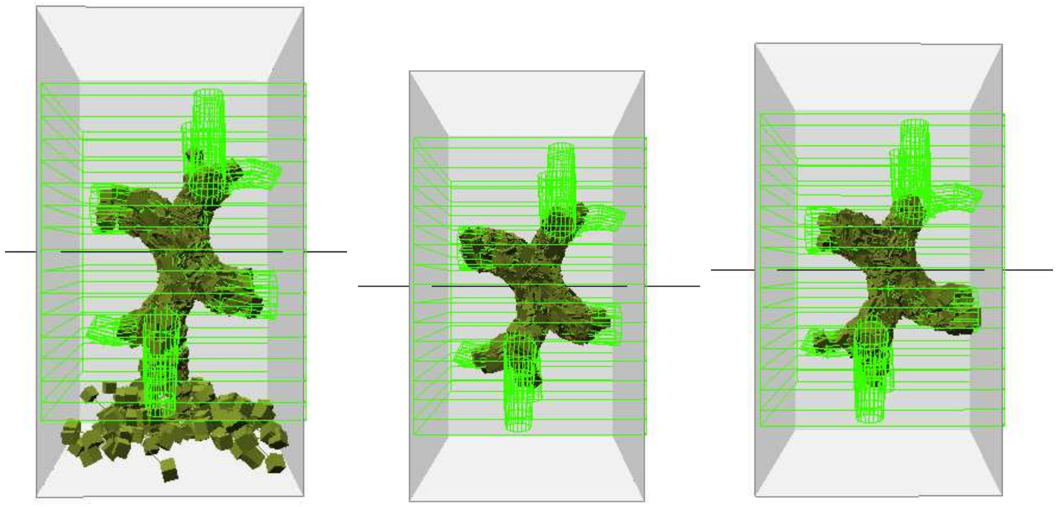

Fig. 15. 3D Maze Serial (9 DOF) examples: MARRT (left), RRT (center), and OBRRT (right).

CHAPTER V

CONCLUSION

In conclusion, we introduced a novel algorithm, Medial Axis RRT (MARRT), which succesfully grows RRTs with high clearance. When compared to RRT and OBRRT, MARRT provides attractive roadmaps. These roadmaps would be safer to navigate for a robot under uncertainty. For future work an exploration into ways to focus the expand step to efficiently bias the roadmap towards a goal configuration could be taken, a high DOF analysis, and potentially how to limit the collision detection calls required by the PushToMedialAxis operation.

REFERENCES

[1] J. H. Reif, Complexity of the mover's problem and generalizations. In *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, pp. 421–427, 1979.

[2] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo, OBPRM: An obstacle-based PRM for 3D workspaces. In *Robotics: The Algorithmic Perspective*, Natick, MA, pp. 155–168, A.K. Peters, Proc. Third Workshop on Algorithmic Foundations of Robotics (WAFR), Houston, TX, 1998, 1998.

[3] 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. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2, pp. 1024–1031, 1999.

[4] J. J. Kuffner and S. M. LaValle, RRT-Connect: An efficient approach to single-query path planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 995–1001, 2000.

[5] S. Rodriguez, X. Tang, J. M. Lien, and N. M. Amato, An obstacle-based rapidly-exploring random tree. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 895–900, 2006.

[6] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, August 1996.

[7] K. Manavi, S. Thomas, and N. Amato, Enhanced local planning for medial axis roadmaps. Tech. Rep., Texas A&M University, 2011.

[8] S.M. LaValle and J.J. Kuffner Jr., Rapidly-exploring random trees: Progress and prospects. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2000.

[9] V. Boor, M. H. Overmars, and A. F. van der Stappen, The Gaussian sampling strategy for probabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2, pp. 1018–1023, May 1999.

[10] D. Hsu, T. Jiang, J.H. Reif, and Z. Sun, Bridge test for sampling narrow passages with probabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 4420–4426, 2003.

[11] J.J. Kuffner Jr. and S.M. Lavalle, Rrt-connect: An efficient approach to single-query path planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2, pp. 995–1001, 2000.

[12] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, Motion planning for a rigid body using random networks on the medial axis of the free space. In *Proc. Ass. for Computing Machinery (ACM)*, pp. 173–180, 1999.

[13] J. M. Lien, S.L. Thomas, and N. M. Amato, A general framework for sampling on the medial axis of the free space. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 4439–4444, 2003.

[14] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, Choosing good distance metrics and local planners for probabilistic roadmap methods. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 630–637, 1998.

[15] S. Gottschalk, M.C. Lin, and D. Manocha, Obbtree: A hierarchical structure for rapid interference detection. In *Proc. Ass. for Computing Machinery (ACM)*,

pp. 171–180, 1996.

[16] A. Vargas E., J.M. Lien, and N.M. Amato, Vizmo++: a visualization, authoring, and educational tool for motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 727–732, 2006.

CONTACT INFORMATION

Name:	Evan John Greco
Address:	c/o Dr. Nancy M. Amato Department of Computer Science and Engineering 3112 TAMU Texas A&M University College Station, TX, 77843
Email Address:	egreco@neo.tamu.edu
Education:	B.S., Computer Science, Texas A&M University, Dec 2012 Undergraduate Research Scholar