# MARRT: Medial Axis Biased Rapidly-Exploring Random Trees

Jory Denny, Evan Greco, Shawna Thomas, and Nancy M. Amato

***Abstract*— Motion planning is a difficult and widely studied problem in robotics. Current research aims not only to find feasible paths, but to ensure paths have certain properties, e.g., shortest or safest paths. This is difficult for current state-of-the-art sampling-based techniques as they typically focus on simply finding any path. Despite this difficulty, sampling-based techniques have shown great success in planning for a wide range of applications. Among such planners, Rapidly-Exploring Random Trees (RRTs) search the planning space by biasing exploration toward unexplored regions. This paper introduces a novel RRT variant, Medial Axis RRT (MARRT), which biases tree exploration to the medial axis of free space by pushing all configurations from expansion steps towards the medial axis. We prove that this biasing increases the tree's clearance from obstacles. Improving obstacle clearance is useful where path safety is important, e.g., path planning for robots performing tasks in close proximity to the elderly. Finally, we experimentally analyze MARRT, emphasizing its ability to effectively map difficult passages while increasing obstacle clearance, and compare it to contemporary RRT techniques.**

## I. INTRODUCTION

Planning the path of a robot through complex environments is notoriously difficult. Despite the difficulty, many applications require solving motion planning tasks such as robotics, virtual reality [11], bioinformatics [18], and computer-aided design [1], among others.

Motion planning algorithms find a sequence of valid (e.g., collision-free) states that take a movable object from some initial position to some final position. However, deterministically finding a path is thought to be intractable [15].

Sampling-based motion planning has proven to be a great success in overcoming this complexity [6], [9]. One such method, Rapidly-exploring Random Trees (RRTs) [9], starts at some state $q_{root}$ and incrementally expands, filling the planning space. RRTs are useful for single-query scenarios and non-holonomic systems. There are several RRT variants aimed at enhancing various properties of the algorithm, such as improving the speed of single-query planning [7] or planning in narrow passages [16], [21]. To our knowledge, there has been little prior work on tailoring RRT growth to produce paths that attempt to maximize clearance, the distance from obstacles. Thus, current RRT approaches are not suited for applications where path clearance is important, such as planning in close proximity to the elderly or in industrial settings.

However, one such graph-based approach, Medial Axis PRM (MAPRM) [3], [12], [20], is able to create roadmaps with nodes that have high obstacle clearance. Graph-based planners, e.g., Probabilistic RoadMaps (PRMs) [6], randomly sample all of the planning space and connect neighboring samples to create a graph representing the connectivity of the planning space. MAPRM differs from PRM in that the nodes of the graph are pushed to the medial axis. MAPRM has the additional benefit of efficiently mapping narrow passages.

In this paper, we present a novel RRT variant, Medial Axis RRT(MARRT), that grows a tree on the medial axis of the free space. Like RRT, MARRT begins by iteratively extending the tree towards a randomly sampled configuration. However, MARRT differs from RRT by pushing each extension of the tree to the medial axis of the free space. In low dimensions, samples can be transformed to the medial axis at a low cost using an exact clearance calculation to nearby obstacles. The specific contributions of the paper are as follows:

- Introduction of a novel method, MARRT, that grows an RRT $\epsilon$-close to the medial axis of the free space.
- Prove $\epsilon$-closeness of the tree to the medial axis and probabilistic completeness under mild assumptions about the planning space, clearance computations, and algorithmic inputs.
- A detailed experimental evaluation in 2D and 3D environments including a demonstration that MARRT yields greater tree and path clearance with minimal computation time overhead compared with other common RRT-based planners such as RRT [9], RRT$^*$ [5], and OBRRT [16].

We note that the goal of MARRT is not necessarily to provide the most efficient planning algorithm but to establish the feasibility of growing an RRT along the medial axis, thereby creating paths with high obstacle clearance.

## II. PRELIMINARIES AND RELATED WORK

In this section, we present some basics of motion planning, review previous algorithms for single-query motion planning, and survey sampling-based planning on the medial axis of the free space.

**Motion planning preliminaries.** A robot is a movable object whose position and orientation can be described by $n$ parameters, or *degrees of freedom* (DOFs), each corresponding to an axis of movement (e.g., positions, orientations, joint

This research supported in part by NSF awards CNS-0551685, CCF-0833199, CCF-0830753, IIS-0916053, IIS-0917266, EFRI-1240483, RI-1217991, by NIH NCI R25 CA090301-11, by Chevron, IBM, Intel, Oracle/Sun and by Award KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST). J. Denny supported in part by an NSF Graduate Research Fellowship.

Jory Denny, Shawna Thomas, and Nancy M. Amato are with the Parasol Lab, Department of Computer Science and Engineering, Texas A&M University, College Station, TX, USA, {jdenny, sthomas, amato}@cse.tamu.edu.

E. Greco contributed to this work as an undergraduate at Texas A&M University. He is currently employed by National Instruments.

angles, etc.). Hence, a robot's placement, or configuration, can be described by a unique point $q = \langle x_1, x_2, ..., x_n \rangle$ in an $n$-dimensional space (where $x_i$ is the $i$th DOF). This space, consisting of all possible robot configurations (feasible or not), is called *configuration space* ($\mathcal{C}_{space}$) [13]. The subset of all feasible configurations is the *free space* ($\mathcal{C}_{free}$), while the union of the infeasible configurations is the *obstacle space* ($\mathcal{C}_{obst}$). Thus, the motion planning problem becomes that of finding a continuous trajectory in $\mathcal{C}_{free}$ from a given start configuration $q_s$ to a goal configuration $q_g$.

In general, it is intractable to compute explicit $\mathcal{C}_{obst}$ boundaries [15], but we can often determine whether a configuration is valid or invalid quite efficiently, e.g., by performing a collision detection (CD) test in the *workspace*, the robot's natural space. The *clearance*, or *obstacle clearance*, of a configuration is its distance to the closest point on the boundary of $\mathcal{C}_{obst}$. In low dimensions, clearance can be computed exactly with CD tests [8]. Because of the cost of explicitly computing $\mathcal{C}_{obst}$ boundaries, research has turned to sampling-based techniques [6], [9] to efficiently explore $\mathcal{C}_{free}$ for valid paths.

## A. Rapidly-Exploring Random Trees (RRTs)

Rapidly-exploring Random Trees (RRTs) [9] are a sampling-based approach to solving single-query motion planning problems. They iteratively grow a tree outwards from a root configuration $q_{root}$. RRTs have been quite useful for a broad range of robotic systems including dynamic environments, kinodynamic robots, and non-holonomic robots. In each expansion attempt, a random configuration $q_{rand}$ is chosen, and the nearest configuration within the tree $q_{near}$ is extended towards $q_{rand}$ up to or at a fixed step size $\Delta q$. From this extension, a new configuration $q_{new}$ is computed and added to the tree if and only if there is a valid path from $q_{near}$ to $q_{new}$. RRTs are probabilistically complete and have an exponential convergence to the sampling distribution over $\mathcal{C}_{space}$ because of Voronoi bias, i.e., RRTs explore $\mathcal{C}_{free}$ efficiently. Despite these important properties, RRTs suffer in the presence of narrow passages or very complex planning systems. In this section, we highlight a few of the RRT variants most relevant to this research.

In an effort to solve single query problems faster, RRT-Connect [7] constructs two trees, one rooted at a start configuration $q_s$ and one rooted at a goal configuration $q_g$. Each tree grows towards the other using a greedy heuristic. Once the two trees meet, a path can be extracted between $q_s$ and $q_g$ using a simple path finding algorithm in the tree. An important contribution of this work was showing the benefit of allowing RRTs to grow with variable step sizes in their greedy heuristic. More specifically, a tree expansion is sometimes allowed to extend until either an obstacle, a maximum expansion distance, or $q_{rand}$ is reached.

Volume-based sampling with RRTs [17] is an approach which utilizes clearance information of nodes in the tree. The clearance of a node defines a hypersphere in $\mathcal{C}_{free}$ which is entirely visible to the node. In this way, volume-based sampling defines an RRT with each node as a hypersphere and defines an efficient exploration technique biasing tree growth away from already covered volumes of $\mathcal{C}_{free}$. However, this is an efficiency improvement for RRT growth and provides no clearance guarantees on the paths created or tree constructed.

Obstacle-based RRT (OBRRT) [16] exploits obstacle information in biasing the growth of the tree to more effectively plan in narrow passages. OBRRT introduces nine growth variations based upon random vectors, random obstacle vectors, tangent obstacle vectors, and even a medial axis biased growth. OBRRT incrementally chooses growth methods based on user-provided probabilities. However, this method does not aim to increase clearance.

Retraction-based RRT [21] uses information gained via obstacle contact analysis and optimization to retract RRT growth along the boundary of $\mathcal{C}_{obst}$ to improve RRT performance in narrow passages. It was later extended to handle articulated models [14]. Selective Retraction-based RRT (SR-RRT) [10] uses simple tests to focus the retractions to narrow passages and avoid expensive growth in open areas of $\mathcal{C}_{free}$.

T-RRT [4] is a method for growing trees along a cost-map over $\mathcal{C}_{space}$. In this approach, a cost-map is defined over the space, and optimization techniques are used to explore this space. However, there needs to be a definition of a threshold allowed for transition costs, and this can be difficult to tune. This method does not guarantee any growth along the medial axis of the space and does not optimize the cost of the path. However, T-RRT has been adopted for obstacle clearance in the work space and performs well in practice.

RRT$^*$ [5] is an approach to ensure asymptotic optimality of tree growth. For example, ensuring the tree asymptotically finds the shortest path is possible and has been shown to be quite effective with this approach. The method can optimize other cost functions, but these have not been well explored and in general must be additive cost metrics. RRT$^*$ expands in the same way as RRT except after expansion the tree will locally "rewire" itself to ensure optimization of the cost function.

## B. Sampling-based Planning on the Medial Axis

The *medial axis* of a polytope is the set of all points equidistant to two or more obstacle boundaries. For an $n$-dimensional space, the medial axis is an $n-1$-dimensional manifold which represents the connectivity of the space; the medial axis is defined from some distance metric, e.g., Euclidean. There have been several motion planning techniques proposed that utilize the medial axis as points on it have maximal clearance and can be used to form 'safe' paths. A method that constructs a medial axis 'skeleton' of workspace biases PRM sampling to increase clearance of roadmaps [3]. Medial Axis PRM (MAPRM) [12], [20] constructs a roadmap with all nodes on or near the medial axis of $\mathcal{C}_{free}$. Figure 1 shows an example of a MAPRM roadmap for a 2D point robot. The strength of MAPRM is that it retracts all sampled nodes (valid or not) to the medial axis of $\mathcal{C}_{free}$ without explicitly computing the medial axis structure. By retracting *all* sampled nodes, the probability

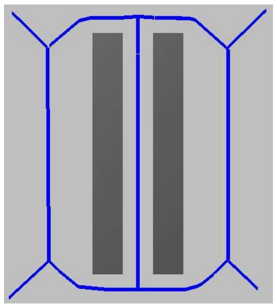


Fig. 1. Resulting MAPRM roadmap for a 2D point robot. Nodes in the roadmap have high clearance and navigate the narrow passage.

of sampling a narrow passage is not only dependent on its volume but also on the volume of the obstacles surrounding it. Thus, in most cases, MAPRM more efficiently maps narrow passages compared to uniform random sampling [20].

MAPRM's main operation, `PushToMedialAxis`$(q, \epsilon)$, retracts a configuration $q$ to within an $\epsilon$ distance of the medial axis of $\mathcal{C}_{free}$, where $\epsilon > 0$ is arbitrarily close to $0$. If $q$ is initially invalid, the function retracts $q$ to its nearest $\mathcal{C}_{obst}$ boundary point. Once in $\mathcal{C}_{free}$, the function computes the nearest $\mathcal{C}_{obst}$ boundary point to $q$, called a witness configuration $w$ (e.g., by using an exact computation of clearance in the workspace). The function pushes $q$ away from $w$, which is in a direction perpendicular to the $\mathcal{C}_{obst}$ boundary, until $w$ changes. Once $w$ changes, it performs a binary search between the current and previous configurations to locally maximize clearance on the ray $\overrightarrow{wq}$ and ensure $\epsilon$-closeness to the medial axis. We review a proof of this in Section III. Unfortunately, the edges in the roadmaps created by MAPRM are not $\epsilon$-close to the medial axis.

The expense of MAPRM is dominated by this retraction function, especially in higher dimensions. For this reason, approximate penetration and clearance computations can be used in higher dimension to push configurations to an approximation of the medial axis [12].

In [2], a path modification technique is introduced to push a path the medial axis of the $\mathcal{C}_{free}$. In low dimensions, it uses techniques similar to MAPRM to yield a path along the medial axis, and in high dimensions, uses optimization techniques to perturb a path towards the medial axis. This can be an expensive operation and can be used in conjunction with techniques shown here. It is irrespective of the planner used to create the path.

## III. MEDIAL AXIS RRT

In this section, we describe Medial Axis RRT (MARRT), a novel variant of the standard RRT algorithm that grows a tree on the medial axis of $\mathcal{C}_{free}$. Inspired by MAPRM, it uses medial axis retractions to generate RRT-style growth near the medial axis without having to explicitly compute its structure.

### A. Algorithm

MARRT is presented in Algorithm 1. Like RRT, it begins with a starting configuration $q_{root}$ and iteratively grows the tree outward toward unexplored regions of $\mathcal{C}_{free}$. In each iteration, it randomly samples a configuration $q_{rand} \in \mathcal{C}_{space}$ and identifies the nearest sample $q_{near}$ in the tree based on some distance metric $\delta$ (e.g., Euclidean distance). MARRT differs from RRT in that instead of growing from $q_{near}$ directly toward $q_{rand}$, it constrains the growth near the medial axis, see line 5 of Algorithm 1. This medial axis growth is recorded as a polygonal chain $I$ of intermediate configurations located $\epsilon$-close to the medial axis. Any intermediate configurations, along with their connections in the polygonal chain, will be added to the tree as nodes, and edges, respectively in `Update`$()$.

**Algorithm 1** Medial Axis RRT

**Input:** The number of expansion attempts $n$, a root configuration $q_{root}$, a maximum expansion length $l$, step size $\Delta q$, and a tolerance $\epsilon$.
**Output:** A tree $T$

```
1: $T$.AddNode(PushToMedialAxis($q_{root}, \epsilon$))
2: for $1 \dots n$ do
3:     $q_{rand} \leftarrow$ RandomCfg()
4:     $q_{near} \leftarrow$ NearestNeighbor($T, q_{rand}$)
5:     $I \leftarrow$ MARRTExpand($q_{near}, q_{rand}, \Delta q, \epsilon, l$)
6:     Update($T, q_{near}, I$)
7: return $T$
```

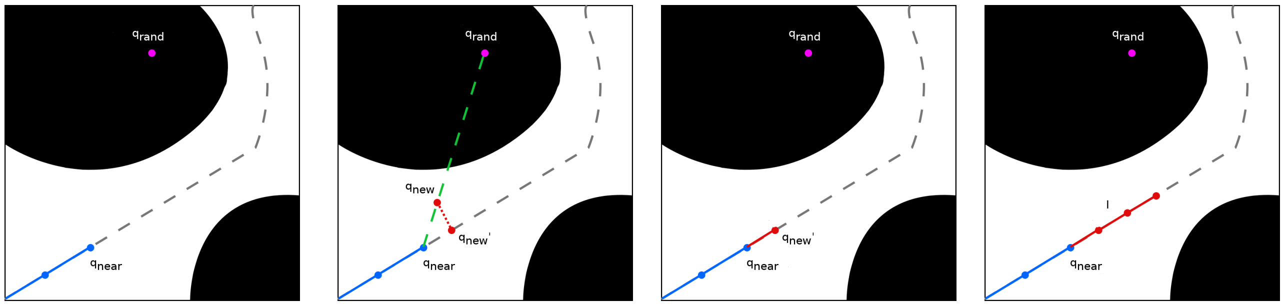
The key difference between RRT and MARRT is the expansion step. Algorithm 2 expands along the medial axis until a maximum expansion length $l$ is reached, expansion fails to make progress, or a connection between successive nodes is blocked by an obstacle (checked by a visibility test `Visible`()). A configuration $q_{prev}$ is first initialized with $q_{near}$ and $I$, the polygonal chain of intermediate nodes on the medial axis, is initialized to $\emptyset$. During each iteration, $q_{prev}$ is added to $I$, $q_{new}$ is extended from $q_{prev}$ toward $q_{rand}$ at a step size $\Delta q$, and $q_{new}$ is pushed to the medial axis. At this point visibility between $q_{prev}$ and $q_{new}$ and the terminating conditions are tested.

**Algorithm 2** MARRT Expand - expands $q_{near}$ towards $q_{rand}$ along the medial axis

**Input:** Configurations $q_{near}$ and $q_{rand}$, a step-size $\Delta q$, a tolerance $\epsilon$, and a maximum expansion length $l$
**Output:** Intermediate Configurations $I$

```
1: Polygonal Chain $I \leftarrow \emptyset$
2: Configurations $q_{prev} \leftarrow q_{near}, q_{new}$
3: repeat
4:     $q_{prev} \leftarrow q_{new}$
5:     $I \leftarrow I \cup \{q_{prev}\}$
6:     $q_{new} \leftarrow$ Extend($q_{prev}, q_{rand}, \Delta q$)
7:     PushToMedialAxis($q_{new}, \epsilon$)
8:     $dist \leftarrow \delta(q_{prev}, q_{new}) + \sum_{q_i, q_{i+1} \in I} \delta(q_i, q_{i+1})$
9: until $q_{prev} \equiv q_{new} \vee \neg$Visible($q_{prev}, q_{new}$)$\vee$ $dist > l$
10: return $I$
```

Figure 2 shows an example of medial axis expansion for a 2D point robot. $MA(\mathcal{C}_{free})$ is indicated by the dashed



(a) (b) (c) (d)

Fig. 2. Example of MARRT expansion for a 2D point robot. (a) $MA(\mathcal{C}_{free})$ is shown by the dashed line. The closest node $q_{near}$ in the tree (shown in blue) to the random sample $q_{rand}$ is selected for expansion. (b) $q_{near}$ extends towards $q_{rand}$ by $\Delta q$ (along the green dotted line) and is pushed to the medial axis to create $q_{new}{}'$ (shown in red). (c) $q_{new}{}'$ is connected to $q_{near}$, if possible. (d) The process repeats until the maximum number of intermediates $I_{max}$ is reached or $q_{new}{}'$ is farther than $l$ from $q_{near}$ along the configurations of $I$.

line (Figure 2(a)). The closest node $q_{near}$ in the tree (in blue) to the random sample $q_{rand}$ (in magenta) is selected for expansion. In Figure 2(b), $q_{near}$ is expanded toward $q_{rand}$ a step size $\Delta q$. The resulting $q_{new}$ is then pushed to the medial axis, and in Figure 2(c), connected to the previously pushed node, if possible. Upon termination, the resulting sequence of intermediates $I$ is constrained to the medial axis with a bias towards $q_{rand}$, as shown in Figure 2(d).

## B. Analysis

In this section, we theoretically analyze MARRT. We show that MARRT grows trees that are $\epsilon$-close to the medial axis and is probabilistically complete under mild assumptions regarding the input parameters and $\mathcal{C}_{free}$.

**Assumptions and Definitions.** Let $\texttt{Witness}(q)$ be a function to compute the closest configuration on the boundary of $\mathcal{C}_{obst}$ to a configuration $q$. For the following lemmas, assume all robots are rigid bodies in 2D or 3D workspaces and all clearance and penetration computations are exact computations performed using collision detection libraries such as PQP [8]. With this assumption, $\texttt{Witness}()$ can be implemented to exactly compute the witness in the specific $\mathcal{C}_{free}$. The theory below will hold as long as clearance in $\mathcal{C}_{free}$ can be computed. Let $\epsilon > 0$ be an arbitrarily small constant. In the following Lemmas, medial axis refers to the medial axis of $\mathcal{C}_{free}$.

*Lemma 1:* $\texttt{PushToMedialAxis}(q, \epsilon)$ moves the input configuration $q$ to a configuration $q'$ $\epsilon$-close to the medial axis.

*Proof:* Let $w \leftarrow \texttt{Witness}(q)$. We select a directional ray $r$. If $q \in \mathcal{C}_{free}$, let $r = \overrightarrow{wq}$. If $q \in \mathcal{C}_{obst}$, let $r = \overrightarrow{qw}$. If $q \equiv w$, let $r$ be a ray through $w$ perpendicular to the boundary of $\mathcal{C}_{obst}$ extending into $\mathcal{C}_{free}$. In all cases $r$ is perpendicular to the boundary of $\mathcal{C}_{obst}$. $\texttt{PushToMedialAxis}$ steps along $r$ until the witness point from the clearance computation changes, i.e., the $q_{i+1}$ configuration's witness point $w_{i+1} \leftarrow \texttt{Witness}(q_{i+1})$ differs from the $q_i$ configuration's witness point $w_i \leftarrow \texttt{Witness}(q_i)$. The medial axis now lies between $q_i$ and $q_{i+1}$. A binary search will compute a configuration $q'$ of maximal clearance along $\overline{q_i q_{i+1}}$ within a resolution of $\epsilon$. ■

*Lemma 2:* The nodes of the tree built by MARRT are $\epsilon$-close to the medial axis.

*Proof:* MARRT begins by computing $q'_{root}$, the pushed configuration of $q_{root}$. $q'_{root}$ is $\epsilon$-close to the medial axis by Lemma 1. Only configurations from the medial axis extend operation are added as nodes to the tree. Each configuration $q_{new}$ of the medial axis extend operation is pushed to the medial axis. By Lemma 1, each $q_{new}$ is $\epsilon$-close to the medial axis. ■

*Lemma 3:* Assuming that the medial axis has as least $\epsilon$ clearance, when $\Delta q \leq \epsilon$, the entire tree built by MARRT is $\epsilon$-close to the medial axis.

*Proof:* Without loss of generality, assume $r \ll \epsilon$ is used by $\texttt{PushToMedialAxis}()$ as the resolution in the binary step, thus the nodes of the tree are $\epsilon$-close to the medial axis. Each edge of the tree is a polygonal chain $I = \{q_0, q_1, \ldots, q_n\}$. It suffices to show that each segment $(q_i, q_{i+1})$ of the polygonal chain will be $\epsilon$-close to the medial axis. Let $q_i$ be the current node being expanded in $\texttt{MARRTExpand}()$, $\vec{r_i}$ be the push direction after stepping $q_i$ towards $q_{rand}$, $\vec{m_i}$ be the tangent of the medial axis in the direction of $r_i$ at $q_i$, and $q_{i+1}$ the configuration after pushing to the medial axis. Because the medial axis has $\epsilon$ clearance, in the worst case $\vec{r_i}$ will be parallel to $\vec{m_i}$. To show that $\overline{q_i q_{i+1}}$, we analyze the shape of the medial axis between $q_i$ and $q_{i+1}$. There are three cases. (1) If the medial axis converges toward $\vec{r_i}$ or (2) if the medial axis stays parallel to $\vec{r_i}$, then $\overline{q_i q_{i+1}}$ is trivially $\epsilon$-close to the medial axis. (3) If the medial axis diverges from $\vec{r_i}$, then there must be another obstacle influencing the shape of the medial axis. However, if this is the case, then the witness point for the stepped configuration would have changed and the medial axis would have been found. This is a contradiction, thus $\overline{q_i q_{i+1}}$ is also $\epsilon$-close to the medial axis. ■

*Lemma 4:* Assuming that the medial axis has as least $\epsilon$ clearance, MARRT is probabilistically complete.

*Proof:* Let $q_s, q_g \in \mathcal{C}_{free}$ be any two configurations in the same connected component of $\mathcal{C}_{free}$. Then there exists a path between $q_s$ and $q_g$ of at least $\epsilon$-clearance from obstacles. Let $q'_s$ and $q'_g$ be $\epsilon$-close configurations to the medial axis after calling $\texttt{PushToMedialAxis}$ on $q_s$ and $q_g$, respectively. Thus, there is a path between $q'_s$ and $q'_g$ along the medial

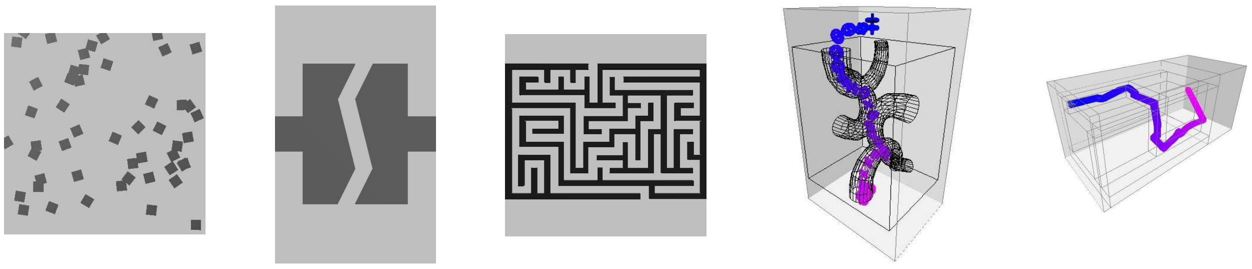

(a) Clutter (b) Tunnel (c) 2DMaze (d) 3DMaze (e) ZTunnel

Fig. 3. Clutter (a), Tunnel (b), and 2DMaze (c) are 2-DOF environments used to compare the RRT growth of various methods. 3DMaze (d) and ZTunnel (e) are 6-DOF problems used to compare the solvability of various methods with example paths shown from start (blue) to goal (magenta).

axis. Note there will be a valid path from $q_s$ to $q'_s$ and from $q_g$ and $q'_g$, and MARRT builds a tree from $q'_s$ to $q'_g$. Define a set of spheres of $\epsilon$ volume $B = \{B_0, B_1, \ldots, B_k\}$ along the path along the medial axis from $q'_s$ to $q'_g$ such that $B_0$ and $B_k$ are the $\epsilon$-spheres at $q'_s$ and $q'_g$, respectively, and for all $B_i \cap B_{i+1} \neq \emptyset$. It suffices to show that MARRT can successfully expand to any $B_i$.

Because each $\epsilon$-sphere has non-zero volume, there is a probability to sample the sphere and extend the tree into that sphere. Thus, there is a non-zero probability that a path along the medial axis will be found as the samples tend to infinity. ■

## IV. EXPERIMENTAL RESULTS

In this section, we experimentally analyze MARRT and compare to related approaches which have some method of ensuring clearance. We compare MARRT against the following methods:

- RRT [9] without any clearance restrictions, referred to as RRT,
- RRT with a clearance constraint for the entire tree, referred to as RRTObst,
- RRT$^*$ [5] optimizing for path clearance instead of path length, referred to as RRT$^*$,
- and OBRRT [16] which has the ability to bias some growth towards the medial axis, referred to as OBRRT.

RRT$^*$ is adapted by using the edge clearance as the cost metric, or the minimum distance between the configurations on the edge and the boundary of $\mathcal{C}_{obst}$ and defining the total cost along a path to be the minimum of the edge clearances along a path within the tree. To compute the edge clearance, the clearance of each configuration along the edge must also be computed. Thus this adaptation of RRT$^*$ is extremely expensive (as seen in the following sections). Note that only two of the above methods create paths along the medial axis of $\mathcal{C}_{free}$, RRT$^*$ and MARRT, and only one, MARRT, does this for the entire tree. We study both 2D and 3D environments of varying topology.

### A. Experimental Setup

RRT, RRTObst, RRT$^*$, OBRRT, and MARRT were implemented in a C++ motion planning library developed in the Parasol Lab at Texas A&M University. This library uses a distributed graph data structure from the Standard Template Adaptive Parallel Library (STAPL) [19], a C++ library designed for parallel computing. For each method, the growth step $\Delta q$ was kept constant at approximately $5-10\%$ of the diagonal of the environment boundary. All methods use PQP [8] for collision detection (CD call), Euclidean distance for distance computations, a brute force neighborhood finding strategy for determining the nearest neighbor of a configuration, and an $\epsilon = 0.01$ for `PushToMedialAxis` (where applicable). MARRT uses $\Delta I = \frac{\Delta q}{10}$ and straight-line local planning for visibility checks. RRTObst uses a clearance constraint proportional to the minimum clearance achieved using MARRT.

We study RRT growth for the various methods in two separate experiments:

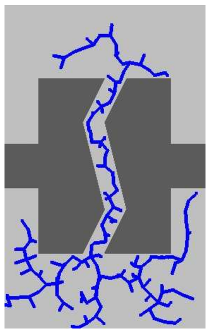
- a quantitative and qualitative look at RRT growth over a fixed number of extensions for 2-DOF problems including Clutter (Figure 3(a)), Tunnel (Figure 3(b)), and 2DMaze (Figure 3(c)) and
- analysis of RRT growth for practical problem solving in the following 6-DOF problems: 3DMaze (Figure 3(d)) and ZTunnel(Figure 3(e)).

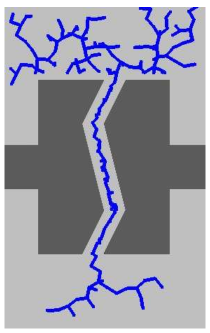
We use the total number of CD calls required for RRT growth as a standard efficiency metric, while the average and maximum tree/path clearances measure the quality of trees/paths generated. Average (maximum) tree/path clearance is computed as an average (maximum) over the clearances of all edges in the tree/path. We do not analyze the minimums here, as in these environments the medial axis of $\mathcal{C}_{free}$ touches the boundary of $\mathcal{C}_{obst}$ thus having an effective clearance of $0$. All experiments were averaged over 10 random seeds, and standard deviations appear as error bars.

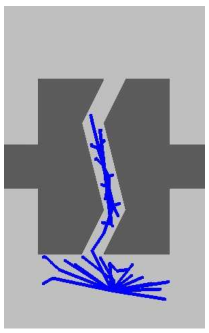
### B. RRT Growth

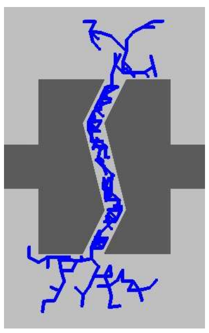
Here we analyze the quality and efficiency of varying RRT growth techniques in three 2-DOF environments: Clutter, Tunnel, and 2DMaze. Each method except RRT$^*$ expands its tree 200 times from a root at the origin of each environment. RRT$^*$ only expands 50 times due to the inefficiency of computing edge clearances when rewiring the tree.

Figure 5 demonstrates the ability of MARRT to create a tree with maximal clearance. MARRT has both the highest


(a) RRT


(b) RRTObst


(c) RRT*


(d) OBRRT

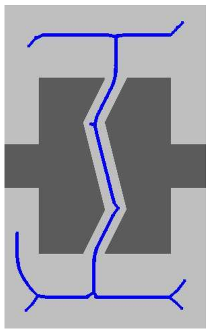
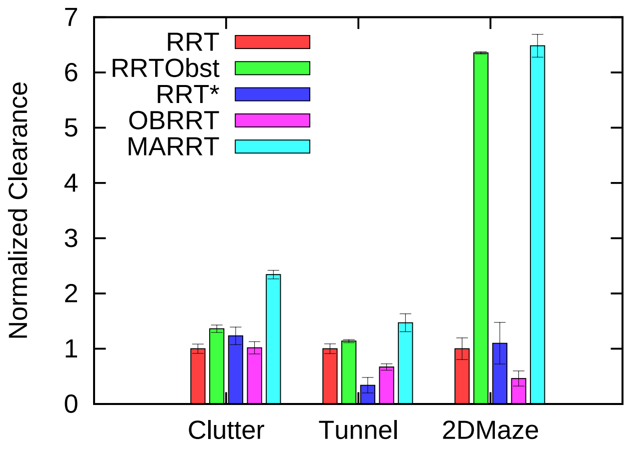
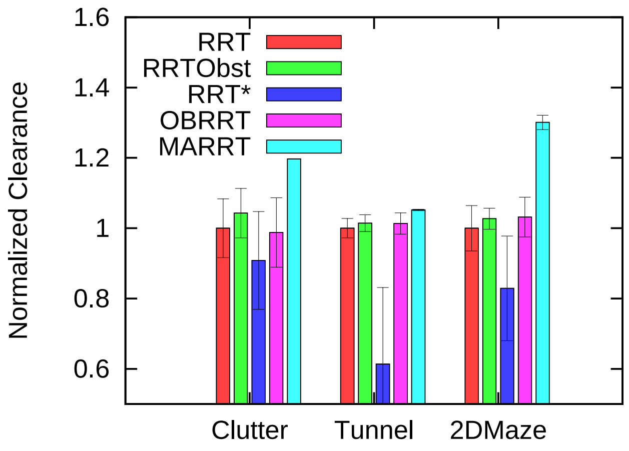

(e) MARRT

Fig. 4. Example trees in the Tunnel environment for each method.



(a) Average tree clearance



(b) Maximum tree clearance

Fig. 5. Tree quality: clearances normalized to RRT for the 2-DOF environments.

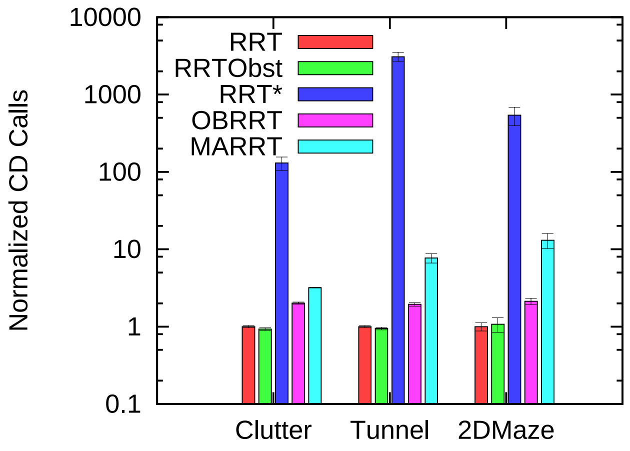
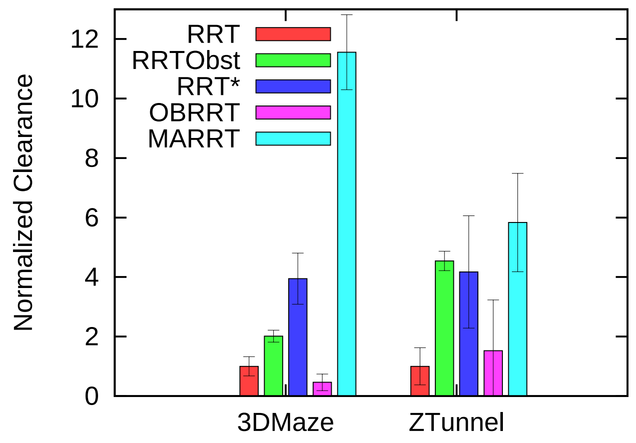


Fig. 6. Method efficiency: log scale of the CD calls required to construct the trees normalized to RRT for the 2-DOF environments.

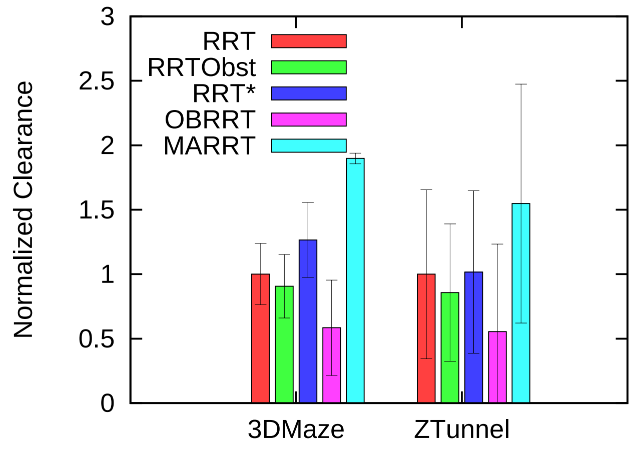
average and maximal clearance across the three environments, with RRT$^*$ coming in second in most cases. Recall that RRT$^*$ is only allowed to expand 50 times due to its high cost, so the coverage of these trees is lower and results might be skewed showing it having higher clearance. Even so, MARRT out-performs RRT$^*$ by this metric. Note that while RRT$^*$ is still able to return a path with asymptotically maximal clearance, it would require a large number of iterations to yield close to maximal clearance for paths. RRTObst is difficult to tune and is not sensitive to regions of the environment, so the clearance constraint can only be as large as the minimum clearance along the medial axis of the environment which explains its suboptimal performance.

Figure 4(e) shows an example MARRT tree grown in the Tunnel environment. Because RRT (Figure 4(a)), RRTObst (Figure 4(b)), RRT$^*$ (Figure 4(c)), and OBRRT (Figure 4(d)) are allowed to expand away from the medial axis, their trees have suboptimal average and maximum clearance values. Note the difference in overall tree clearance for MARRT as compared with the other methods.

Despite significantly improving the tree's clearance, MARRT has a comparable cost to the base RRT growth methods, as shown in Figure 6, within a factor of 10 in the worst case. Observe that RRT$^*$ with path clearance as an optimization parameter is extremely expensive to compute cost-to-go metrics between possible connections of the tree, causing an enormous cost as compared with the other methods. Considering this, MARRT provably and experimentally shows increased clearance for the entire tree, not just paths, with comparable computation overhead to other RRT variants.



(a) Average path clearance



(b) Maximum path clearance

Fig. 7. Path quality: clearances normalized to RRT for the 6-DOF environments.

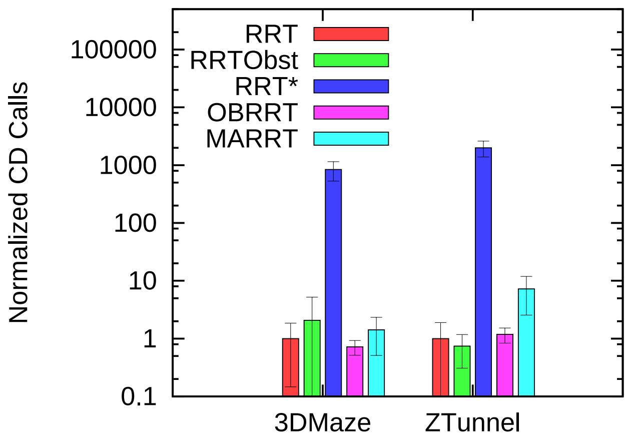


Fig. 8. Method efficiency: log scale of the CD calls required to construct the trees normalized to RRT for the 6-DOF environments.

### C. Querying

We have seen that MARRT experimentally improves the overall tree quality in terms of clearance as compared to contemporary approaches. In this experiment, we show how MARRT can be useful for solving example queries in difficult environments with high clearance paths.

We compare the various methods in two complex 6-DOF problems, 3DMaze and ZTunnel, to see their comparative effectiveness in solving queries for difficult problems. We compare the average and maximum path clearance values as a metric of path quality, shown in Figure 7 and the cost in terms of CD calls, shown in Figure 8. We give each method at most 10 hours to solve the query. Because of RRT$^*$'s high cost, it was only able to complete 7 runs instead of 10 like the others.

Figure 7 echoes the quality metrics seen in tree clearance for path clearances. MARRT successfully yields the largest average and maximum path clearances, sometimes by a large margin as in the 3DMaze environment. RRT$^*$ and RRTObst are relatively close to MARRT on these environments in terms of clearances as the narrow passages are quite tight. RRT and OBRRT perform poorly on this metric because they create paths which typically scrape along obstacle boundaries.

Additionally, Figure 8 shows the minimal overhead of MARRT to optimize the path clearance as compared with RRT$^*$. MARRT is within a factor of 10 in the worst case to RRT, RRTObst, and OBRRT for ZTunnel and can be more cost effective that RRTObst in the 3DMaze. This emphasizes the relative efficiency used to guarantee the clearance properties of the tree. Again, RRT$^*$ is extremely expensive to compute and is in the best case still two orders of magnitude slower than MARRT. RRT, RRTObst, and OBRRT efficiently map the narrow passages but lack clearance guarantees, which could be quite important, e.g., when needing path safety.

## V. CONCLUSION

In this paper, we present a new technique, called Medial Axis RRT, that increases tree and path clearance for RRT growth. We illustrate MARRT's ability to maximize clearance in environments spanning 2-DOF and 6-DOF problems with minimal cost as compared to a contemporary optimization technique, RRT$^*$. In the worst case, MARRT is approximately only an order of magnitude off methods specifically designed for narrow passages. In the future, we would like to relax some of the assumptions made on the input parameters and analyze the growth rate of MARRT to see if it matches that of RRT. Additionally, improving the efficiency of approximate medial axis computations [12], [20] will allow this method to be applicable to high DOF problems, including articulated linkage robots, and improve the efficiency of MARRT in the worst case conditions to still be quite comparable to contemporary methods.

## REFERENCES

[1] O. B. Bayazit, G. Song, and N. M. Amato. Enhancing randomized motion planners: Exploring with haptic hints. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 529–536, 2000.

[2] R. Geraerts and M. H. Overmars. Creating high-quality paths for motion planning. *Int. J. Robot. Res.*, 26(8):845–863, 2007.

[3] C. Holleman and L. E. Kavraki. A framework for using the workspace medial axis in prm planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 1408–1413, San Franasisco, CA, 2000.

[4] L. Jaillet, J. Cortés, and T. Siméon. Sampling-based path planning on configuration-space costmaps. *Trans. Rob.*, 26(4):635–646, Aug. 2010.

[5] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research (IJRR)*, 30:846–894, 2011.

[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.*, 12(4):566–580, August 1996.

[7] 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)*, pages 995–1001, 2000.

[8] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha. Distance queries with rectangular swept sphere volumes. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 4, pages 3719–3726 vol.4, 2000.

[9] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *Int. J. Robot. Res.*, 20(5):378–400, May 2001.

[10] J. Lee, O. Kwon, L. Zhang, and S. Yoon. Sr-rrt: Selective retraction-based rrt planner. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2543–2550, 2012.

[11] J.-M. Lien, O. B. Bayazit, R.-T. Sowell, S. Rodriguez, and N. M. Amato. Shepherding behaviors. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4159–4164, April 2004.

[12] J.-M. Lien, S. Thomas, and N. Amato. A general framework for sampling on the medial axis of the free space. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 3, pages 4439 – 4444, sept. 2003.

[13] T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, October 1979.

[14] J. Pan, L. Zhang, and D. Manocha. Retraction-based RRT planner for articulated models. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2529–2536, 2010.

[15] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, pages 421–427, San Juan, Puerto Rico, October 1979.

[16] 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)*, 2006.

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

[18] G. Song and N. M. Amato. Using motion planning to study protein folding pathways. In *Proc. Int. Conf. Comput. Molecular Biology (RECOMB)*, pages 287–296, 2001.

[19] G. Tanase, A. Buss, A. Fidel, Harshvardhan, I. Papadopoulos, O. Pearce, T. Smith, N. Thomas, X. Xu, N. Mourad, J. Vu, M. Bianco, N. M. Amato, and L. Rauchwerger. The STAPL Parallel Container Framework. In *Proc. ACM SIGPLAN Symp. Prin. Prac. Par. Prog. (PPoPP)*, pages 235–246, San Antonio, Texas, USA, 2011.

[20] 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)*, volume 2, pages 1024–1031, 1999.

[21] L. Zhang and D. Manocha. An efficient retraction-based RRT planner. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2008.