Adapting RRT Growth for Heterogeneous Environments

Jory Denny[1], Marco Morales[2], Samuel Rodriguez[1], and Nancy M. Amato[1]

***Abstract*— Rapidly-exploring Random Trees (RRTs) are effective for a wide range of applications ranging from kinodynamic planning to motion planning under uncertainty. However, RRTs are not as efficient when exploring heterogeneous environments and do not adapt to the space. For example, in difficult areas an expensive RRT growth method might be appropriate, while in open areas inexpensive growth methods should be chosen. In this paper, we present a novel algorithm, Adaptive RRT, that adapts RRT growth to the current exploration area using a two level growth selection mechanism. At the first level, we select groups of expansion methods according to the visibility of the node being expanded. Second, we use a cost-sensitive learning approach to select a sampler from the group of expansion methods chosen. Also, we propose a novel definition of visibility for RRT nodes which can be computed in an online manner and used by Adaptive RRT to select an appropriate expansion method. We present the algorithm and experimental analysis on a broad range of problems showing not only its adaptability, but efficiency gains achieved by adapting exploration methods appropriately.**

I. INTRODUCTION

Often in robotic applications, computing a feasible path between start to goal locations is needed, which is referred to as *motion planning*. This can be challenging due to uncertainty, optimality, or even planning through heterogeneous environments. Motion planning not only has applications in robotics but also influences areas such as computer aided design (CAD) [2], bioinformatics (e.g., protein folding) [26], [27], and assembly mechanisms [13]. As this problem is difficult to solve with exact methods [23], sampling-based methods have been extensively used. Rapidly-exploring Random Trees (RRTs) [14] have been successful in single-query problems, while Probabilistic RoadMaps (PRMs) [11] are useful for multi-query use.

Sampling-based approaches construct a graph representing the connectivity of the space of valid robot configurations. RRTs use local exploration by progressively expanding a tree outwards in random directions until an input query can be solved, or a maximum number of iterations has been reached. PRMs use a global strategy of randomly sampling the entire space followed by connecting the sampled configurations with simple paths. With these methods, constrained regions (e.g., narrow passages) become difficult to discover. Narrow passages often arise in high dimensional problems or application domains such as CAD [2] applications (e.g., removal of an engine component). Moreover, heterogeneous environments mixed with narrow passages, cluttered regions, and free space are an additional challenge.

Simple visibility metrics [17], [19], [22] have been proposed that can be efficiently computed in an online manner and used to effectively improve roadmap construction in narrow passages and heterogeneous environments. For PRMs visibility is approximated as a simple ratio of successful connections over total connection attempts, calculated during the node connection phase. This has been used to filter sampling to structurally improve the roadmap [22], and exploited for heterogeneous environments in [18], [28]. However, it is non-trivial to utilize visibility like this in RRTs.

In this paper, we seek to develop strategies to approximate and utilize visibility to improve RRT performance in narrow regions and heterogeneous environments. In particular, we present a novel online method for approximating visibility of RRT nodes based on tree growth. We then show how these metrics can be used in an Adaptive RRT that chooses an appropriate RRT exploration method in a two level selection mechanism based upon the visibility of the node being expanded and a cost-sensitive adaptive strategy over the growth methods. Specific contributions of this paper include:

- A novel definition of visibility for RRTs.
- A new RRT variant, Adaptive RRT, that selects a growth method based on the visibility metric and a cost-sensitive adaptive strategy.
- An experimental analysis of visibility and Adaptive RRT in a broad set of example queries showing its effectiveness and adaptability in heterogeneous environments.

II. RELATED WORK AND PRELIMINARIES

In this section, we discuss motion planning preliminaries, RRTs, and utilizing metrics in PRMs.

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 object component (e.g., object positions, object orientations, link angles, and link displacements). Hence, a robot's placement, or configuration, can be uniquely described by a point $(x_1, x_2, ..., x_n)$ in an n-dimensional space (where x_i is the ith DOF). This space, consisting of all possible robot configurations (feasible or not), is

This research supported in part by NSF awards CNS-0551685, CCF-0833199, CCF-0830753, IIS-0917266, IIS-0916053, EFRI-1240483, RI-1217991, by NIH NCI R25 CA090301-11, by THECB NHARP award 000512-0097-2009, by Chevron, IBM, Intel, Oracle/Sun, by Award KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST), and by Instituto Tecnologico Autonomo de Mexico and Academia Mexicana de Cultura A.C.

[1]J. Denny, S. Rodriguez, and N. M. Amato are with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX, 77843, USA {jdenny, sor8786, amato}@cse.tamu.edu.

[2]M. Morales is with the Academic Department of Digital Systems, Instituto Tecnologico Autonomo de Mexico, Progreso Tizapan, CP 01080, Mexico marco.morales@itam.mx.

called *configuration space* ($\mathcal{C}_{space}$) [16]. The subset of all feasible configurations is the *free space* ($\mathcal{C}_{free}$), with the complement being the infeasible configurations, the *obstacle space* ($\mathcal{C}_{obst}$). Thus, the motion planning problem becomes that of finding a continuous trajectory in $\mathcal{C}_{free}$ between given start and goal configurations. In general, it is intractable to compute explicit $\mathcal{C}_{obst}$ boundaries [23], but we can often determine whether a configuration is feasible or not quite efficiently, e.g., by performing a collision detection (CD) test in the *workspace*, the robot's natural space. Sampling-based methods (Probabilistic RoadMaps (PRMs) [11] and Rapidly-exploring Random Trees (RRTs) [14]) have been shown to be successful at solving a diverse set of problems.

Advances in RRTs. RRTs [14] incrementally explore $\mathcal{C}_{space}$ from some start configuration q_{root} by iteratively selecting a random direction q_{rand} followed by finding q_{near}, the nearest node in the tree to q_{rand}, and steering q_{near} towards q_{rand} to create q_{new} which is then added to the tree. More precisely, once q_{near} is selected, an *Extend* operation is done on q_{near} towards q_{rand} which iteratively takes steps at the environmental resolution, checking validity at each step. This continues until either a distance of Δq, the $\mathcal{C}_{obst}$ boundary, or q_{rand} is reached. RRTs explore space with Voronoi bias explaining their efficiency in expansive spaces [14]. Even so, RRTs have deficiencies in traversing narrow spaces of the environment, often colliding with obstacles.

Much work has been done in advancing RRTs. RRT-Connect [12] utilized a bidirectional search with two trees, one at the start and one at the goal, and biases exploration between the two trees. Rapidly-exploring Random Graphs (RRGs) and optimal RRT (RRT*) [10] are able to asymptotically find optimal paths, e.g., shortest paths. Execution-extended RRT (ERRT) [4] biases exploration towards goals and waypoints along a path in real-time. Obstacle-based RRT (OBRRT) [24] biases exploration with using the workspace obstacle geometries in nine different growth methods: G0 and G1 utilize random $\mathcal{C}_{space}$ directional vectors, G2 and G3 utilize random obstacle vectors, G4 rotates a robot and then translates it, G5 and G6 use tangent obstacle vectors, G7 uses tangent $\mathcal{C}_{obst}$ vectors, and G8 biases towards the medial-axis. Dynamic-Domain RRTs [29] restrict the Voronoi region in the sampling domain of boundary nodes. EG-RRT [9] adapts the distribution bias accounting for dynamic limitations of the robot and reduces the risk of unproductive expansions. Retraction-based methods [15], [21], [30] have shown great success in exploring narrow passages.

One method, Selective Retraction-based RRT [15], filters the usage of Retraction-based RRT by applying a possibly expensive filtering test, *bridge-line test*, to determine whether the tree is within or near a narrow passage. Our work is similarly motivated, except we propose a general approach to combine growth methodologies, and we use an inexpensive local strategy for determining the growth method selected. Moreover, the Selective Retraction-based RRT could be used as one of the subroutines in our algorithm.

Another method adapts RRT growth for constrained systems reducing metric sensitivity in approximating the *cost-to-go* between configurations [6]. While exploring the space, information is collected, such as the *constraint violation frequency*. This metric is then used to define better near nodes for the RRT growth. We approach the problem differently, in that our visibility metric allows selection of growth methods based upon the near node instead of finding more appropriate near nodes for expansion. Nonetheless, this method could be used in conjunction with our approach.

Utilizing metrics in PRMs. PRMs [11] map $\mathcal{C}_{free}$ by randomly sampling valid configurations and validating simple straight-line paths between nearby samples to form the edges of the map. Then, the map is queried for a final solution. However, PRMs are inefficient at mapping narrow passages and heterogeneous spaces [7].

Many approaches have been proposed to improve mapping techniques, e.g., Obstacle-based PRM [1] and Gaussian filtering [3]. Visibility PRM [25] defined heuristics for only accepting samples that benefit the roadmap. Specifically, this method required samples to improve the roadmap by discovering new samples not visible from any previous sample or merging disconnected parts of the map together. This was evolved to define metrics for analyzing the evolution of roadmaps and effectively utilize them to bias roadmap growth [19], [22]. Essentially, the analysis classifies nodes by their affect on the roadmap and utilizes visibility that is computed for each node during roadmap connection as a ratio of successful to total connection attempts. Unfortunately, this only applies to PRMs. Other methods use visibility to bias samples as well, e.g., feature-sensitive motion planning [18], [20], [28] which adapts the sampling method based upon a decomposition and classification of regions. Hybrid PRM [8] uses a cost-sensitive adaptive strategy to select sampling methods for PRMs.

Even though many advances have been made for RRTs, to the best of the authors' knowledge no approach generalizes growth method adaptation based upon an online metric to take advantage of the strengths of each method as presented in this paper.

III. VISIBILITY

In this section, we introduce a novel approximation of visibility that can be computed and utilized during RRT construction. Visibility is an estimation of how easy it is to connect a configuration q to the other configurations in its local surroundings. This is based on the visibility between configurations: a configuration q' is visible from another configuration q if some method, e.g., a local planner, can produce a continuous sequence of adjacent, at some resolution, configurations $\tau = (q_0, q_1, q_2, \ldots, q_n,)$, where $q = q_0$, $q' = q_n$, and $\forall q_i \in \tau, q_i \; is \; valid$. While exact visibility is intractable to compute in high dimensional spaces, it can be easily approximated for PRMs based upon a success-fail ratio from node connection [19]. For RRTs, however, it may not be as simple as a success-fail ratio of expansion attempts. In particular, since RRTs can expand very small distances, there is a need to differentiate between large expansions, e.g., all the way to q_{rand}, and encountering the boundary of $\mathcal{C}_{obst}$.

Algorithm 1 Updating the visibility for q_{near} and q_{new}.

Input: Configurations q_{near}, q_{new}, q_{rand}; Distance metric δ; Penalty for hitting $\mathcal{C}_{obst}$ $HitPenalty$; Expansion distance Δq.

1: $expandDist = \delta(q_{near}, q_{new})$
2: $expandRatio = expandDist/\Delta q$
3: $nearVis = q_{near}.visibility$
4: $q_{near}.UpdateAverageVisibility(expandRatio)$
5: **if** $0 < expandRatio < 1$ **then**
▷ *Partial Expansion*, $\mathcal{C}_{obst}$ hit
6: $q_{new}.visibility = HitPenalty$
7: **else if** $expandRatio = 1$ **then**
▷ *Full Expansion* at Δq or to q_{rand}
8: $q_{new}.visibility = (expandRatio + nearVis)/2$
9: **end if**

Building on this idea, Algorithm 1 presents a novel, online, updating mechanism for approximating visibilities in RRT methods. Configurations q_{near}, q_{new}, and q_{rand} are provided as input from an RRT growth attempt. Additionally, the roots of the tree are initialized with visibility 1 in an optimistic manner.

There are three possible outcomes that we differentiate during RRT expansion – null, partial or full expansion – each of which causes the node visibilities to be updated accordingly, as shown in Algorithm 1. In *null expansion*, q_{near} could be against $\mathcal{C}_{obst}$ and expansion towards q_{rand} fails (Figure 1 (a)). In *partial expansion*, q_{near} can expand part of the way to Δq or q_{rand}, implying that q_{new} is against $\mathcal{C}_{obst}$ (line 5, also Figure 1 (b)). In *full expansion*, expansion succeeds at a distance of Δq or expansion reaches q_{rand}, (line 7, also Figure 1 (c)). In all cases, q_{near}'s visibility is averaged with $expandRatio$ (the distance traveled divided by Δq) as a running average with $UpdateAverageVisibility()$.

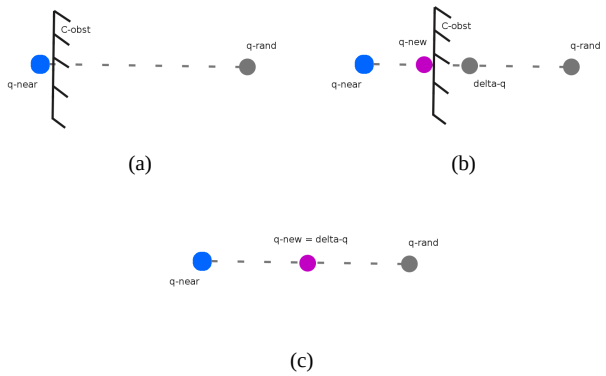


Fig. 1. Possible outcomes during RRT expansion: (a) *null expansion* of q_{near} against $\mathcal{C}_{obst}$; (b) *partial expansion* of q_{near} which collides with $\mathcal{C}_{obst}$ before reaching Δq; and (c) *full expansion* of q_{near} to Δq.

For *null expansion*, $expandRatio$ is 0. If q_{near} is against $\mathcal{C}_{obst}$, this penalty quickly reduces q_{near}'s visibility towards 0 over subsequent iterations.

When *partial expansion* occurs, q_{new} is initialized with a wall hit penalty $HitPenalty$. This penalty is user defined, and can affect the growth method chosen in subsequent iterations. In this scenario, q_{new} is in close proximity to $\mathcal{C}_{obst}$. When q_{near} is very close to $\mathcal{C}_{obst}$, $expandRatio$ is low and reduces q_{near}'s visibility. However, if q_{near} is slightly less than Δq away from $\mathcal{C}_{obst}$, then q_{near} is likely to still be in a high visibility area, so its visibility will be little affected.

For *full expansion* cases, q_{new} is likely in an open area, but not guaranteed since full expansion can occur in a constrained region of $\mathcal{C}_{space}$. Since this is unpredictable, q_{new} "inherits" part of the visibility of q_{near} averaged with $expandRatio$. q_{near} is rewarded with $expandRatio$ as it is likely to be in a high or mid-visibility area.

Through this definition, configurations within narrow spaces of $\mathcal{C}_{free}$ will have low visibility as expansion will either fail or occur in small increments. In open spaces, configurations will have high visibility as expansions will either expand at Δq or reach q_{rand}.

IV. Adaptive RRT

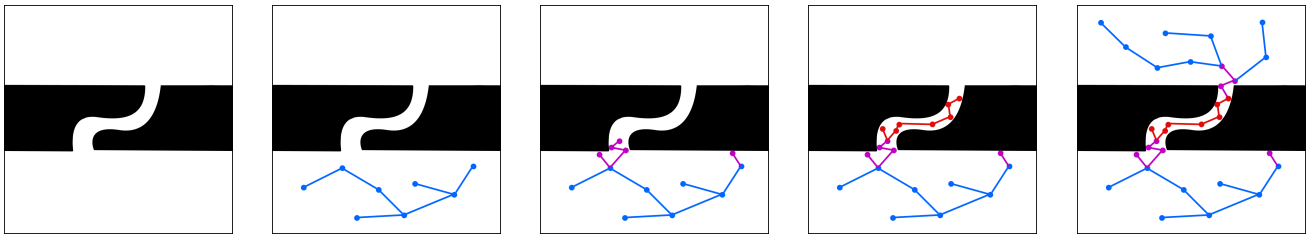
In this section, we present Adaptive RRT, a novel approach that exploits visibility to guide RRT growth. Intuitively, if we are expanding from a node in an open space, we should apply an inexpensive growth method such as randomly choosing a direction. However, if we are expanding within a narrow passage, more expensive growth methods may be more successful. Algorithm 2 summarizes Adaptive RRT.

Algorithm 2 Adaptive RRT which chooses an RRT growth method based upon visibility.

Input: Query Q, growth distance Δq, and an ordered set of visibility/growth strategy set pairs $S = \langle (d_1, GS_1), (d_2, GS_2), \ldots, (d_i, GS_i) \rangle$

1: $T = \emptyset$
2: **while** Q is not solved **do**
3: $q_{rand} =$ `RandomCfg`$()$
4: $q_{near} =$ `NearestCfg`(T, q_{rand})
5: $vis = q_{near}.visibility$
6: $GS \leftarrow$ `SelectGrowthStrategySet`(vis)
7: $G \leftarrow$ `SelectGrowthStrategy`(GS)
8: $q_{new} = G.$`Grow`$(q_{near}, q_{rand}, \Delta q)$
9: `UpdateTree`(q_{near}, q_{new})
10: `UpdateVisibility`$(q_{near}, q_{new}, q_{rand})$
11: `Reward`(GS)
12: **end while**

In addition to the standard RRT input such as Δq, Adaptive RRT requires an ordered set S of visibility/growth strategy set pairs for a two level growth strategy selection mechanism. The algorithm begins like a standard RRT. T is initialized, and until the query is solved, we iterate over choosing q_{rand}, selecting q_{near} and expanding. The difference is that, after q_{near} is found, we select the growth method based upon q_{near}'s visibility. We split this decision in two levels: in the first level, we select the set of growth methods GS in use for the approximated visibility vis, in the second level we choose a particular method G from the selected growth set based on a probability distribution that is



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

Fig. 2. Progression of Adaptive RRT in a simple environment (a). (b) shows rapid exploration through high visibility areas (nodes shown in blue). (c) shows the tree adapting to enter a narrow passage, which the tree classifies as mid visibility nodes (magenta). (d) shows exploration through the passage classifying nodes as low visibility (red). (e) shows completion of the method and classification of various parts of the tree as high, mid, and low visibility corresponding to the correct areas of the environment.

adapted according to performance. Then, using G, q_{near} is grown towards q_{rand} to define q_{new}, and `UpdateTree`() adds q_{new} to T and adds the edge (q_{near}, q_{new}) to T. Visibility of q_{near} and q_{new} is updated through `UpdateVisibility`(), as described in Section III. A cost-sensitive adaptive strategy in [8] is adapted for RRT to define and update the weights of the methods in GS with the `Reward`() function. The process of selection and reward is fully described in Section IV-A. Adaptive RRT stops when either a maximum number of iterations is reached or the input query Q is solved.

A. Selecting the Growth Method

Growth method selection happens on two levels in order to adapt weights of the methods for each visibility range independently. First, a growth strategy set GS is selected based upon vis in `SelectGrowthStrategySet`(). This is done by finding the appropriate pair (d_k, GS_k) in S, where d_k is a visibility threshold and GS_k is a growth strategy set, such that $d_1 = 0$ and $d_k \leq vis \leq d_{k+1}$ (if d_{k+1} does not exist, 1 is used as the upper bound). In the second level of growth method selection, `SelectGrowthStrategy`() selects a strategy from GS based upon a weighted probability distribution over the methods in the set. After expansion, the weights are updated through the `Reward`() function. In this section, we describe our cost-sensitive adaptive algorithm used in the later level of growth method selection. It is derived from Hybrid PRM [8] which adaptively selects sampling methods for PRMs.

Let $GS = \langle G_1, G_2, \ldots, G_j \rangle$ be the set of growth strategies from which we are selecting. Ideally, the growth strategies that are most effective will be selected more often. We define a given cost c_l and a weight w_l for all methods G_l with $1 \leq l \leq j$. Initially, $w_l = 1$ for each method and the cost is given as input. Through the weights and costs we can arrive at a probability p_l for each growth method and select randomly based upon these probabilities for each growth method. After the growth method is applied, reward will be given based upon its success. In this paper, we define the reward as a ratio of expansion distance $dist_e$ to Δq, which is always between 0 and 1. If the growth fails, we reward with a value of -0.001. Firstly, a cost-insensitive probability p_l^* is defined for each method

$$p_l^* = (1-\gamma)\frac{\log(w_l+1)}{\sum_{m=0}^{j}\log(w_m+1)} + \gamma\frac{1}{j}$$

where $\gamma \in [0,1]$ is a fixed constant describing a weighting factor on probability derived from a uniform distribution over the methods in the growth strategy set. The first component of the cost-insensitive probability is weighted based upon how effective the method is in relation to the other methods in the growth set. Cost is taken into account when defining the probability p_l

$$p_l = \frac{p_l^*/c_l}{\sum_{m=0}^{j} p_m^*/c_m}$$

where this fraction is a weighted sum compared to the other cost-insensitive probability/cost ratios. c_l can be updated during execution to keep track of the average cost of actually running the method in practice in relation to the others.

Finally, to update the weights based upon a reward r_l we apply the following formula

$$w_l = w_l e^{\gamma \frac{r_l}{p_l^* j}}$$

where $r_l = 0$ for any strategy not selected. The exponential factor allows for rapid adaptability throughout the execution of the method. The factor is taken in proportion to the cost-insensitive probability and uniform distribution weighting.

B. Example

An example is shown in Figure 2 which is a simple environment with a narrow passage (Figure 2(a)). High visibility nodes are shown in blue, mid visibility nodes in magenta, and low visibility nodes in red. In this example we define three growth strategy sets H, M, and L each used for high, mid, and low nodes, respectively. H includes the standard RRT growth suited to high visibility regions. M includes inexpensive growth methods from OBRRT [24] such as selecting a random workspace obstacle vector to bias the expansion direction. L includes expensive approaches, such as utilizing the medial axis (e.g., G8 in OBRRT). When the tree is expanding through a high visibility area (Figure 2(b)), randomized exploration is taken to quickly expand through the space. As the tree expands and approaches the mouth of the narrow passage (Figure 2(c)), the visibility of

TABLE I

ADAPTIVE RRT GROWTH SETS USED FOR EXPERIMENTAL ANALYSIS. VISIBILITY IS SPLIT INTO LOW, MEDIUM, AND HIGH VISIBILITY SETS.

Adaptive RRT Growth Sets			
Method	Low Vis	Mid Vis	High Vis
AdaptiveRRT1	{G0-G6}	∅	∅
AdaptiveRRT2	{G2-G6}	∅	{G0-G3}
AdaptiveRRT3	{G4-G6}	{G2-G6}	{G0-G3}

the nodes drops and growth methods will be chosen from M. As the tree expands into a narrow passage, expensive techniques can be used to guide the growth of the tree effectively, which are selected when the visibility for those nodes adapt and drop below the threshold for the growth set M (Figure 2(d)). The visibility adapts from failed expansion attempts or partial expansion attempts which occur often in mid and low visibility regions. Finally, as the tree expands out of the passage, and successful attempts occur often, the visibilities return to high values appropriately (Figure 2(e)).

V. EXPERIMENTS

In this section, we analyze the effectiveness of Adaptive RRT on a range of heterogeneous single-query problems comparing its effectiveness to that of RRT and OBRRT. Section V-A discusses experimental setup. Section V-B and Section V-C show advantages of using Adaptive RRT with varying cost metrics.

A. EXPERIMENTAL SETUP

All experiments were run on a Rocks Cluster running CentOS 5.1 with Intel XEON CPU 2.4 GHz processors with the GNU gcc compiler version 4.1.

RRT, OBRRT, and Adaptive RRT were implemented in a C++ motion planning library developed in the Parasol Lab at Texas A&M University which uses a distributed graph data structure from the Standard Template Adaptive Parallel Library (STAPL) [5], a C++ library designed for parallel computing. All methods use $\Delta q = 10$ with the Euclidean distance metric (Δq is about 5% of the length of the diagonal of the boundary used). All nearest neighbor computations are done in an exact brute force fashion.

We experiment with three variants of Adaptive RRT, which use one, two, or three sets of growth methods, respectively. Growth sets are shown in Table I. AdaptiveRRT2 uses a high visibility growth set with threshold 0.4 and a low visibility growth set. AdaptiveRRT3 uses a high visibility growth set with threshold 0.6, a mid visibility growth set with threshold 0.4, and a low visibility growth set. G0-G6 are growth methods from OBRRT [24] and are described in Section II. Overlap of growth methods between the sets is motivated by multiple growth strategies being useful in different levels of visibility. We use only G0-G6 in OBRRT because these provided the best performance in extremely heterogeneous environments. All growth methods have equal weighting as it is unclear which is most beneficial for heterogeneous environments.

Environments are shown in Figure 3.

- In 2D (Figure 3(a)), a robot must traverse a series of difficult narrow passages and cluttered areas to solve a query between the two free spaces above and below the passage. We use both a 2DOF circle and a 3DOF stick robot, denoted 2D-Circle and 2D-Stick, respectively.
- In Tic-Tac-Toe (TTT) (Figure 3(b)), a robot must traverse a series of rooms starting from the middle going in a spiral to the bottom right room. Some rooms contain clutter. The robot is a 3DOF stick.
- In 3D (Figure 3(c)), a large spinning top-like robot (6DOF) must traverse two long narrow passages with a region of clutter in between to successfully solve a query from the bottom left to the top right of the environment.

All experiments are run on 10 random seeds until the example query is solved. For these experiments, we report the total number of nodes in the tree, the total number of collision detection (CD) calls, and the time required to generate the tree as metrics for efficiency of tree construction. All results are normalized to RRT and averaged over successful trials. Outliers are removed using 1.5 times the interquartile range of the data. All data is shown in Figure 4.

B. Expansion-based Cost Metric

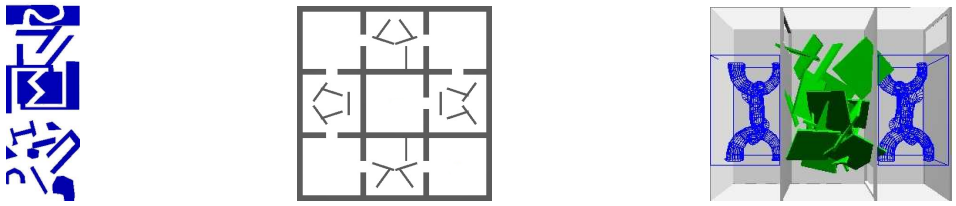
For the 2D environment, we defined cost as $c = \Delta q - dist$ where $dist$ is the expansion distance during growth. The number of nodes, number of CD calls, and time are compared in Figure 4(a), Figure 4(b), and Figure 4(c), respectively.

When comparing the number of nodes, the variants of Adaptive RRT generally need fewer nodes to solve the query as compared to OBRRT and RRT. This has a dual benefit in RRT exploration. First, fewer nodes implies more effective exploration of the space, and second, fewer nodes allows for more efficient nearest neighbor queries because their efficiency is proportional to the size of the tree. The number of CD tests performed is not necessarily a direct performance indicator, as more successful growth attempts require more CD tests. However, when compared with OBRRT, we can see that Adaptive RRT is able to successfully reduce the number of CD calls in the majority of cases. Adaptive RRT is able to efficiently explore the space by appropriately adapting the growth method selection to more successful methods in using this cost metric. Finally, we can see that Adaptive RRT ultimately provides a reduction in planning time for the 2D-Stick environment as compared with OBRRT and RRT. Adaptive RRT performed similarly to OBRRT in 2D-Circle.

In a deeper analysis of the cost metric used, we can see that the more successful growth methods will have cheaper cost, as it relates high cost with short expansion attempts. We found this metric to be most beneficial in environments where the Voronoi bias of RRT expansion is largely biased by the goal. In the 2D environment, this is apparent as the tree's largest region of Voronoi bias is towards the top.

C. Time-based Cost Metric

For the TTT and 3D environments, we use a cost proportional to the computational time required to execute each



(a) 2D (b) Tic-Tac-Toe (c) 3D

Fig. 3. Various environments for experimental analysis. All queries must traverse through difficult heterogeneous regions of narrow passages and clutter.

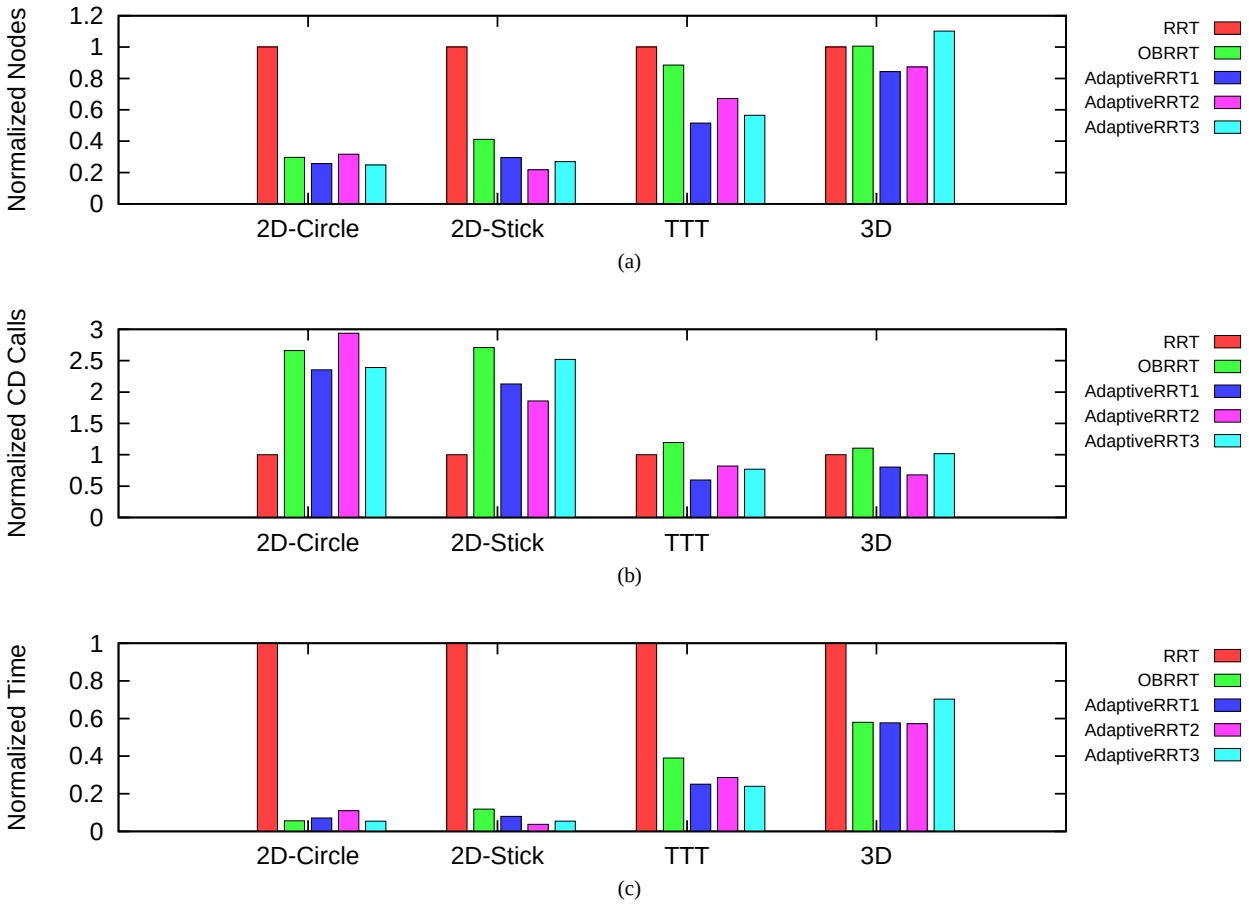


Fig. 4. The number of nodes (a), number of CD calls (b), and the time (c) compared between RRT, OBRRT, and Adaptive RRT in the various environments. Results are normalized to RRT.

growth method for Adaptive RRT. The number of nodes, number of CD calls, and time are compared in Figure 4(a), Figure 4(b), and Figure 4(c), respectively.

In the TTT environment, the number of nodes and time are clearly reduced for each of the Adaptive RRT methods, the number of CD calls is cheaper than OBRRT, and the total time required is lower. This implies again that the Adaptive RRT approach solves the problem more efficiently compared to OBRRT and RRT. In the 3D environment, the number of nodes is lower in AdaptiveRRT1 and AdaptiveRRT2, the number of collision detection calls compared with OBRRT is less for all AdaptiveRRT cases, and the total time is comparable to OBRRT for Adaptive RRT. From these metrics, we can see that the time required for neighborhood searches was slightly less than what is required for RRT. Adaptive RRT gained efficiency over OBRRT by significantly reducing both CD calls and number of nodes in the tree. From our analysis, we see that Adaptive RRT more effectively chose growth methods with average expansion success (implying fewer CD calls) allowing more trials for expansion to be completed. This was useful as the Voronoi bias towards the goal was very limited in this environment, so adapting based on computational cost of the growth method allowed cheaper methods to be selected as they were just as effective in this environment as the more successful methods.

Additionally, we can see that AdaptiveRRT3 suffered from an over segmentation of the growth sets. Not all methods are right for every environment, and still some tuning

of Adaptive RRT is required. However, we recommend AdaptiveRRT1, AdaptiveRRT2, and AdaptiveRRT3 as good heuristics for setting up growth sets.

D. Discussion

From the previous experiments, we can see that Adaptive RRT successfully adapts growth strategy selection to provide an improvement in efficiency over other methods. From our experiments, we suggest two ways to use Adaptive RRT effectively. One, we suggest understanding the environment to choose the cost metric effectively. When the environment is amenable to goal biasing, e.g., the 2D environment, we suggest using reward-based cost, otherwise we recommend time-based cost. Two, avoid over-fragmentation of visibility growth sets, using two sets of targeted growth, one for high and one for low visibility ranges seemed to be a good balance across all environments tested. In the future, we would like to gain a better understanding of building growth sets and assigning appropriate visibility thresholds to maximize effectiveness for input queries.

VI. CONCLUSION

In this paper, we presented a novel definition of visibilities for RRTs based upon local growth knowledge and an algorithm, Adaptive RRT, to effectively adapt RRT growth using a two level growth selection method based upon visibility and a cost-sensitive adapting scheme. This algorithm has little overhead in computing visibility as it adapts and updates based upon previous growth experience in an online manner. In the future, we plan to explore creation of a fully adaptable framework for RRTs.

REFERENCES

[1] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. Obprm: an obstacle-based prm for 3d workspaces. In *Proceedings of the third workshop on the algorithmic foundations of robotics on Robotics : the algorithmic perspective: the algorithmic perspective*, WAFR '98, pages 155–168, Natick, MA, USA, 1998. A. K. Peters, Ltd.

[2] 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.

[3] 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)*, volume 2, pages 1018–1023, May 1999.

[4] J. Bruce and M. Veloso. Real-time randomized path planning for robot navigation. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, Switzerland, 2002.

[5] A. Buss, Harshvardhan, I. Papadopoulos, O. Pearce, T. Smith, G. Tanase, N. Thomas, X. Xu, M. Bianco, N. M. Amato, and L. Rauchwerger. STAPL: Standard template adaptive parallel library. In *Proc. Annual Haifa Experimental Systems Conference (SYSTOR)*, pages 1–10, New York, NY, USA, 2010. ACM.

[6] P. Cheng and S. LaValle. Reducing metric sensitivity in randomized trajectory design. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, volume 1, pages 43–48 vol.1, 2001.

[7] D. Hsu, J.-C. Latombe, and H. Kurniawati. On the probabilistic foundations of probabilistic roadmap planning. *Int. J. Robot. Res.*, 25:627–643, July 2006.

[8] D. Hsu, G. Sánchez-Ante, and Z. Sun. Hybrid PRM sampling with a cost-sensitive adaptive strategy. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3885–3891, 2005.

[9] L. Jaillet, J. Hoffman, J. van den Berg, P. Abbeel, J. M. Porta, and K. Goldberg. EG-RRT: Environment-guided random trees for kinodynamic motion planning with uncertainty and obstacles. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2011.

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

[11] 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.

[12] 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.

[13] J. C. Latombe. Motion planning: A journey of robots, molecules, digital actors, and other artifacts. *Int. Journal of Robotics Research*, 18(11):1119–1128, 1999.

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

[15] 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.

[16] 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.

[17] M. Morales, R. Pearce, and N. M. Amato. Analysis of the evolution of C-Space models built through incremental exploration. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1029–1034, April 2007.

[18] M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. M. Amato. A machine learning approach for feature-sensitive motion planning. In *Algorithmic Foundations of Robotics VI*, pages 361–376. Springer, Berlin/Heidelberg, 2005. book contains the proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), Utrecht/Zeist, The Netherlands, 2004.

[19] M. A. Morales A., R. Pearce, and N. M. Amato. Metrics for analyzing the evolution of C-Space models. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1268–1273, May 2006.

[20] M. A. Morales A., L. Tapia, R. Pearce, S. Rodriguez, and N. M. Amato. C-space subdivision and integration in feature-sensitive motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3114–3119, April 2005.

[21] 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.

[22] R. Pearce, M. Morales, and N. M. Amato. Structural improvement filtering strategy for prm. In *Robotics: Science and Systems*, 2008.

[23] 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.

[24] 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.

[25] T. Simeon, J.-P. Laumond, and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics*, 14(6):477–493, 2000.

[26] A. P. Singh, J.-C. Latombe, and D. L. Brutlag. A motion planning approach to flexible ligand binding. In *Int. Conf. on Intelligent Systems for Molecular Biology (ISMB)*, pages 252–261, 1999.

[27] 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.

[28] L. Tapia, S. Thomas, B. Boyd, and N. M. Amato. An unsupervised adaptive strategy for constructing probabilistic roadmaps. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4037–4044, May 2009.

[29] A. Yershova, L. Jaillet, T. Simeon, and S. M. Lavalle. Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3856–3861, April 2005.

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