Rapidly Exploring Random Search Explorer

Aakriti Upadhyay and Chinwe Ekenna

***Abstract*— Motion planning for robots have generally worked with a single goal solution but in real world problems, multi-goal positions are more often the case. In this work, we propose a Rapidly Exploring Random Search Explorer (RESE) algorithm, that works for multi-goal scenarios. RESE generates connection mediators, called witness nodes, within a radial distance from a configuration in the graph, to make feasible connections. RESE exploits the ideas from Rapidly-exploring Random Graphs (RRGs) and includes a Q-learning technique to help track witness nodes of successful trajectories while also attempting connections in narrow regions by considering multiple directions. We test RESE in three different scenarios:(1) where the robot has to move from a single start to a single goal position, (2) a single start to multiple goal positions and (3) multiple start from any random position to multiple goal positions. Our results show RESE quickly finds a solution in multi-goal experiments and performs comparably well in single goal experiments compared to other existing motion planning methods.**

I. Introduction

Motion planning algorithms exist that solve a varying number of tasks e.g., character animation for games and movies, virtual prototyping and search and rescue missions [7], [24], deformable robots [12], [23], [26], and manipulation planning [16]. Sampling Based Motion Planning methods (SBMP) [8] are one of the widely used [families of] motion planning algorithms because they are known to be probabilistic complete i.e., if a solution path exists, the probability of finding that solution increases as the number of samples tends towards infinity. Sampling-based methods are broadly classified into two main classes: roadmap or tree/graph-based methods such as the Probabilistic Roadmap Method (PRM) [18] and tree-based methods such as Rapidly exploring Random Tree (RRT) [19]. RRT's and its' variants [25], [29], [30] have been successful in planning single query trajectory but are not tailored towards multiple start and goal positions. This finds applicability in situations where more than one robot has to be deployed to successfully complete a mission e.g, space exploration or search and rescue operations. When robots are deployed for example during search and rescue, there is a need to monitor their locations, communication accessibility and their ability to adapt to changes they might encounter not planned before going in for the mission. Time utilization and planning is very important in these situations or else the consequence could be disastrous. In this work we implement a new SBMP algorithm called Rapidly Exploring Random Search Explorer (RESE), that aims to address the challenge of planning in areas where robots are deployed to any unknown position and requires to reach goal positions from there, in a multi-goal environment. Our method considers random start state representations in addition to having the ability to perform multi-goal planning. RESE differs from RRT algorithms because it builds a graph of trees, similar to the Rapidly exploring random graphs (RRG) algorithm [17]. RESE however creates witness nodes and incrementally builds a graph of trees based on how important the witness node is in reducing the number of separate connected components in the graph. We use a reward and cost approach (i.e. reinforcement learning) to update and monitor the performance of the witness node based on successfully formed connections between two random nodes. Witness nodes are created within a determined radius area act as a connection mediator between two configurations added to the roadmap. We perform experiments using 3 test cases, i.e. single query, multi query and random multi query. The first case considers path planning from a single start to a single goal position, the second case considers a single start to the multiple goal positions and last case considers multiple start from any random position to the multiple goal positions. We perform experiments in a variety of environments and compare with existing PRM, RRT and it's variants. Our results show that in the multi query and random multi query experiments RESE outperforms other methods and performs comparably well in the single query experiments.

A.Upadhyay, C.Ekenna are with the Department of Computer Science, University at Albany, New York, 12222, USA. {aupadhyay,cekenna}@albany.edu

II. Related Work

In this section, we define important preliminaries of motion planning and describe relevant related work.

A. Motion Planning Primitives

The motion planning problem involves finding a valid path (e.g., collision-free and satisfying all joint limit and/or loop closure constraints) for a movable object starting from its start configuration to a goal configuration in an environment [8]. A single configuration is defined based on the movable object's d independent parameters or DOFs (degrees of freedom).

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, link displacements). Hence, a robot's placement, or configuration, can be uniquely described by a point $(x_1, x_2, ..., x_n)$ in a n dimensional space (x_i being the ith DOF). This space consisting of all possible robot configurations (feasible or not) is called a configuration space (C-Space) [21]. The subset of all feasible configurations is the free space (C-free), while the union of the unfeasible configurations is the blocked or obstacle space (C-Obstacle). Thus, the motion planning problem becomes that of finding a continuous trajectory in C-free connecting start and goal pair of configurations.

The general motion planning problem of trying to find a collision-free path from some starting state to some goal region has been extensively studied. In particular, sampling-based techniques have received much attention over the last 15 years. The Rapidly-exploring Random Tree (RRT) operates by growing a tree in state space, iteratively sampling new states and then steering the existing node in the tree that is closest to each new sample towards that sample. The RRT has many useful properties including probabilistic completeness and exponential decay of the probability of failure with the number of samples [20]. The Rapidly-exploring Random Graph (RRG) is an extension of the RRT algorithm [17]. In addition to the nearest connection, new samples are also connected to every node within some connected component. The result is a connected graph that not only rapidly explores the state space, but also is locally refined with each added sample. This continuing refinement ensures that with infinite samples, the RRG contains all possible paths through the environment that can be generated by the steering function used to connect samples.

Previous work [28] looked into improving local planning paths by developing a method that uses poisson point process to sample the points in a given configuration and can be applicable to PRM methods such as RRG, FMT (Fast Marching Tree) [15] and BTT (bottleneck tree) [27]. A similar work was also presented in paper [9] where authors proposed a new local planning method, that increases the connectivity of the roadmap by extending simple one-dimensional interpolation methods such as the straight-line methods to a two-dimensional subspace of the total planning space. The local planner degrades performance during execution in complex environments due to high collision detection calls and unreachable samples.

Methods like PRM* and RRT* in [17], has shown that PRM methods can be asymptotically optimal i.e. the cost of the returned solution converges almost surely to the optimum, which hinges on connections between stochastic sampling-based path planning algorithms and the theory of random geometric graphs. But in practice, it requires many iterations for samplers to produce an optimal solution.

B. Some RRT Variants

Many enhancements has been made in the past for RRT algorithms in order to comply with the desired applications of these algorithms in a given environment. [4] compares and improves on the RRT* and the PRM* algorithms using a new incremental sampling based motion planning algorithm based on Rapidly-exploring Random Graphs (RRG), denoted by RRT‡‡ (RRT sharp), which guarantees asymptotic optimality and ensures that the constructed spanning tree rooted at the initial state contains lowest-cost path information for vertices which have the potential to be part of the optimal solution.

Another RRT variant described in [3] called CL-RRT‡‡, generates trajectories using closed-loop prediction. They plan with closed-loop prediction which allows for the handling of complex unstable dynamics and avoids the need to find computationally hard steering procedures. CL-RRT‡‡ algorithm finds two types of neighbors, one using euclidean distance and another with radial distance. A RRT variant in [5] explores a different class of dynamic programming algorithms for solving shortest-path problems on random graphs generated by iterative sampling methods. It improves the rewiring step in RRT‡‡ algorithm, the policy improvement during the rewiring step is performed not only locally but rather on a set of vertices that are classified as promising during the current iteration.

To ameliorate the performance of RRT methods on dynamically changing regions in a given workspace, Yershova et. al. in [30] proposed a sampling framework, Dynamic-Domain RRTs, which grows voronoi based regions in the dynamic environment. It biases the random node selection to be within a radius r of q_{near} or expansion will not occur. The radius is dynamically determined from failed expansion attempts. On improving in this approach, Denny et. al. in [11] proposed Dynamic Region-biased RRT, that biases growth based on dynamically moving regions using reeb graphs. The algorithm captures the workspace topology by dynamically moving sampling regions along an embedded graph and works well for holonomic problems but fails to perform in non-holonomic problems.

In these research works, variants of RRT or RRT itself, are designed to find an optimal solution during path planning or to work in dynamic environments, etc. However, the implementation of these methods is restricted to solving a single goal problem and are not tailored towards solving a multi-goal problem. RESE overcome the restriction of these methods to solve for a multi-goal problem by using the witness nodes that help in making connections in diverse directions during connection and path generation phase.

C. SBMP Algorithms for Search and Rescue

One application of multi-robots is carrying out search and rescue missions while using robots with different capabilities. Consider a search and rescue scenario where a team of ground robots and UAVs are deployed, path planning for this team of robots will be difficult when you consider different constraints in these robots.

The paper [14] proposes Rapidly-exploring Information Gathering (RIG) algorithm that extends ideas from Rapidly-exploring Random Graphs (RRGs) and combines them with branch and bound techniques to achieve efficient optimization of information gathering while also allowing for operation in continuous space with motion constraints without knowing the goal position. The goal is to find a trajectory that maximizes an information metric (e.g., variance reduction, information gain, or mutual information) and also falls within a pre-specified budget constraint (e.g., fuel, energy, or time). Work in [6] had reduced the time needed to plan paths by 25 percent. The heterogeneous team of robots was divided into homogeneous groups of robots and an online planning method was applied where the plan of each group is updated iteratively with each new information.

Both search and rescue operations are complex optimization problems and it gets more complicated within an uncertain domain. In an unknown domain, the findings of the first search robot can significantly change the planning of other robots following its lead.

III. RESE ALGORITHM

A. Problem description

In this work, we define the problem of motion planning for multiple goals based on maximizing the number of witness nodes in the region to find the shortest path to reach these goals.

$$d(x_{rand}, w) = \theta, \quad r/2 \leq \theta \leq 2r. \tag{1}$$

Equation (1) defines w as a witness node, x_{rand} as a random configuration point, r as radial distance, d() as euclidean distance function and θ as the constant distance which lies between $r/2$ and $2r$. We define witness nodes as potential intermediate vertices/nodes in the configuration space that help in making a connection between any two sampled configuration points in the space. These witness nodes are workspace points that are not configuration points and vanishes from the $C-Space$ after the successful connection is made. Blocked nodes are considered as the nodes that are not able to make a connection with other neighboring nodes in a particular direction in space towards which path is expanding to reach the goal. Our algorithm presented in this paper tries to connect blocked nodes to a reachable node in the region by generating witness node at distance θ in different directions from the blocked node.

During the connection phase, RESE utilizes the capabilities of witness node to generate feasible paths towards multiple goal positions. RESE generates maximum witness nodes for a current node within a distance θ and these witness nodes throw random rays to make a connection with the new node. From the generated witness nodes, RESE selects the witness node that is capable of making a connection to new node from a current node. Witness node steers the connection between random node x_{rand} and new node x_{new} by expanding to possible maximum distance. Every time a new witness node w_{new} is added to the witness vertex set M due to a successful expansion, connections are attempted from all other vertices in the graph through a witness node that is within a ball of radius r for new node x_{new}. The algorithm attempts to connect nodes in narrow regions (during expansion) by diverging directions to expand from a witness node in C-free space with time complexity $O(\log n)$ where n is the blocked nodes. As seen in Figure 1, RESE connects the random nodes by producing witness nodes within a radial distance θ from the current random node. On successful connections between two random nodes, a witness node is generated for a new potential node during each iteration as the algorithm builds a path towards the goal.

B. Algorithm Description

Algorithm 1 initializes the values for a witness node and new random node as it extends towards the target node from the start node. The query is solved when the last goal is connected to the graph. The reward of a witness node is increased or decreased with respect to the success or failure

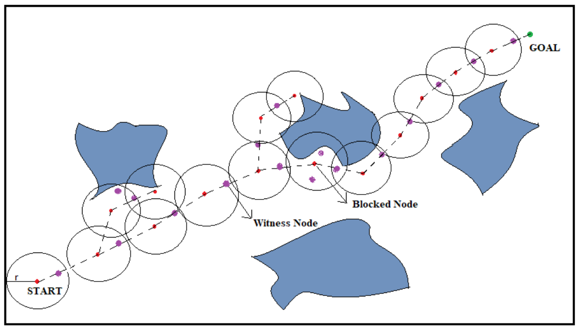


Fig. 1: *The figure gives a pictorial view of RESE in a simple environment. Random nodes are denoted by red dots, witness nodes are purple dots and a green dot represents the goal. Blue surfaces are C-Obstacle in the region and remaining are C-free.*

connection made between two nodes. The cost of a witness node is updated based on the distance it extends from the current random node to the new node using UpdateCost() method. The RewardNode() method calculates the reward for the witness node w as the success or failure in connecting a current random node x_{rand} to a new node x_{new} through it for parameters in equation (1).

Algorithm 1 Expanding Graph using witness node

```
Input. Query Q, S be start configuration point, $w$ be witness
    node within radial distance $r$ from S, $q_{new}$ is a new
    configuration point.
 1: $w \leftarrow$ GetWitnessPoint(S), $q_{new} \leftarrow$ RandomCfg()
 2: while Q is not solved and $w \neq$ S do
 3:     if Connect(S, $w$, $q_{new}$) is true then
 4:         UpdateCost($w$)
 5:         RewardNode($w$)
 6:         S $\leftarrow q_{new}$
 7:         $q_{new} \leftarrow$ RandomCfg()
 8:         $w \leftarrow$ GetWitnessPoint(S)
 9:     else
10:         $w_{new} \leftarrow$ GetWitnessPoint(S)
11:         $w \leftarrow w_{new}$
12:     end if
13: end while
```

On each iteration, once the connection between the current random node and the new node is made, the newly connected node becomes the source to make a connection with other nodes in the $C-Space$, in order to reach either of the goal node. RandomCfg() method outputs a new node on each call that is close to goal node and lies in C-free.

GetWitnessPoint(), i.e. Algorithm 2, selects the best nearest witness node within the radial distance r from the current random node. The current random node is taken as the center of the circle from where a witness node is selected within the radial distance to the current node, from set M iteratively, where M is the set of maximum witness nodes generated within the ball of radius r for the current node. The algorithm returns a potential witness node for the current random node, such that the witness node has not been visited before and lies in C-free. Initially, the values of cost $C = 0$ and reward $R = 1$ are given as input for a newly selected witness node.

Algorithm 2 Updating nearest witness node

```
Input. Configuration points $w_1$ and $q_1$ where $w_1 \neq q_1$,
    radial distance $r$, cost parameter C, reward parameter
    R and an ordered set of cost/reward for witness nodes
    M = $\langle(w_1, C_1, R_1), (w_2, C_2, R_2), ...., (w_k, C_k, R_k)\rangle$.
1: Initialize: C ← 0, R ← 1, r ← 1.0
2: while $w_1$ in radius($q_1$, $r$) or $d_{max}(w_1, q_1) \geq r$ do
3:    if $w_1$ is collision free then
4:       Update M($w_1$, C, R)
5:    end if
6:    if ¬visited($w_1$) then
7:       return $w_1$
8:    end if
      $w_1$ ← RadialRandomCfg($q_1$)
9: end while
```

In Algorithm 2, q_1 is the current random node for which a witness point w_1 is selected on meeting conditions from set M. RadialRandomCfg() method selects another nearest witness node for the current random node from set M on each call such that the witness node is unvisited. A cost-insensitive probability P is defined for each witness node,

$$P(w_h) = (1 - \kappa)\frac{log(R_h + 1)}{\Sigma_{m=0}^{S} log(R_m + 1)} + \frac{\kappa}{\tau} \quad (2)$$

where $\kappa \in [0, 1]$ is a fixed constant describing a learning factor on probability derived from a uniform distribution over the region for witness node set M and τ is defined as the size of witness node set and R_h is the reward parameter for witness node w_h in the set at index h. Cost is taken into account when defining the probability P^*

$$P^*(w_h) = \frac{P(w_h)/C_h}{\Sigma_{m=0}^{S} P^*(w_m)/C_m} \quad (3)$$

where this fraction is a cost-insensitive probability/cost ratio of witness node w_h compared to the total cost-insensitive probability/cost ratios of other witness nodes. C_h is updated during execution to keep track of the cost a witness node incurs in relation to the others. Finally, the reward is updated based upon a distance witness node covers to connect a new node x_{new}, i.e. r_h, we apply the following formula

$$R_h = R_h e^{\kappa \frac{r_h}{P(w_h) * S}} \quad (4)$$

where r_h = 0, if the witness node fails to connect the new node x_{new} in the C-free space.

IV. Experimental Setup

All methods were implemented in a C++ motion planning library developed by the Parasol Lab at Texas A&M University [1]. All experiments for each environment are averaged over 10 runs with different random seeds and were executed on a Dell Optiplex 7040 desktop machine running OpenSUSE operating system. The PRM methods use Obstacle based motion planning method (OBPRM) [2], all methods use euclidean distance metric, brute force K-Closest nearest neighbor technique [22] and the RAPID collision detection library [13].

We perform experiments on four different environments, considering single and multi-goal scenarios for a single robot in the environment as shown in Figure 2.

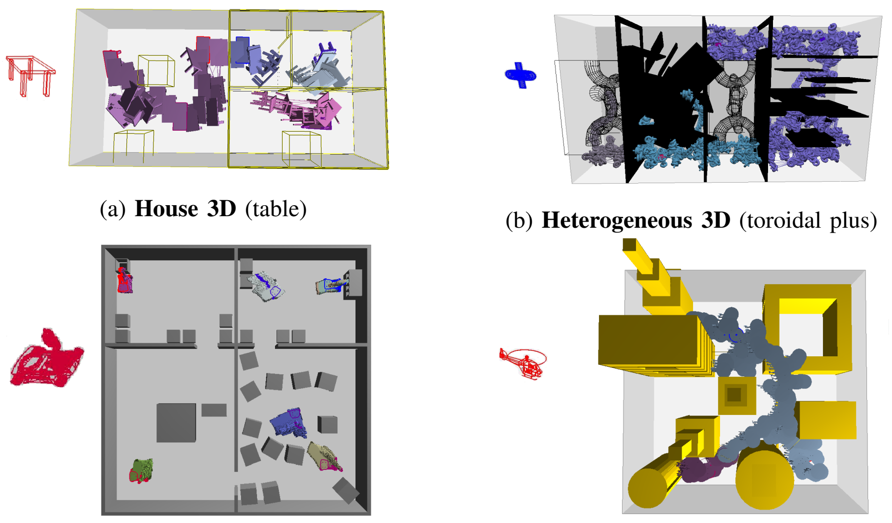
- **House 3D**: This environment is a 3D representation of a house where the robot is in the shape of a table which has to be moved from one room to another. There are four rooms in the house with each room having just one door to enter as shown in Figure 2a. In the first criteria, this environment is set to have just one start position of the table and one end position in another room which does not have a direct entry from the start position. In the second criteria, the environment has multiple goal positions where the table has to reach each room that has at least one goal position.
- **Heterogeneous 3D**: This environment is a 3D maze kind of structure with narrow passages between the walls. The robot has to pass through the tunnels to reach its goal(s) as shown in Figure 2b. In case of a single goal, the start and goal positions are kept separated by multiple walls and two tunnels. In case of multiple goal scenario, the goals are kept in intermediate places where the robot has to either pass through a hole in the wall or a tunnel to reach from start to goal.
- **KukaYouBot**: In this environment, an 8 DOF robot has been placed. The environment contains four rooms as shown in Figure 2c. The base of the robot has 5 DOFs that allow it to move forward. In a single goal scenario, the robot moves through narrow passages and arrives at a destination where it has to perform some action, like grasping or putting an object down. In a multi-goal scenario, the robot will perform the same task but at multiple destinations.
- **Helico**: In this environment, the robot is a helicopter that has to traverse through many obstacles like short and tall buildings and narrow passages between obstacles as shown in Figure 2d. The robot might have to change its vertical position to reach some goals. In a multi-goal scenario, the different goal locations have been kept at different heights from ground level to make it more complicated to solve.

V. Experimental Results

In this section, we discuss our approach and results with RRT, AdaptiveRRT [10], DynamicDomainRRT and RRG algorithms for single query experiments, and PRM and RRG for multi query experiments.

A. Single Query Experiments

RESE was compared with RRT, AdaptiveRRT, Dynamic-DomainRRT and RRG algorithms using a single start and single goal scenario in the environments. The aim of this experiment is to compare the performance of RESE in terms of time to solve the query as shown in Figure 3b and the number of nodes generated as shown in Figure 3a. The


(a) **House 3D** (table)

(b) **Heterogeneous 3D** (toroidal plus)

(c) **KukaYouBot** (8 DOFs robot)

(d) **Helico** (helicopter)

Fig. 2: Heterogeneous Environments Studied

results show that RESE performs equivalent to RRG in KukaYouBot, Helico, and 3D Heterogeneous environments but not so well in House3D environment. Our results show during the multi-query environments that RESE is better suited for multi query scenarios due to the extra computation of witness nodes that enables RESE to solve quickly towards the nearest goal.

B. Multi Query Experiments

We compare RESE with PRM and RRG algorithms for the multi-query scenario where multiple goal states are defined in the environment and the aim is to reach all these states. Other previously defined methods do not consider multi goal states and so will not be considered in this experiment. We consider 5 multi goal positions placed at different positions in the environment. We consider a time cap of 24 hrs for all the methods to reach all the goal states successfully.

Table I and II give a summary of our results. RESE completes the queries in all the environments and out performs all the other methods in terms of time needed to solve the query and number of nodes needed. BasicPRM fails to complete in KukaYouBot and 3D Heterogeneous environments whereas it poorly performs in House3D and Helico environments compared to RRG and RESE algorithms. Except in House3D environment, RRG failed to complete execution of all seeds for other environments. These seeds randomize the placement of nodes during execution of any sampling based motion planning algorithm. The inability of PRM and RRG to successfully complete all the seeds shows its incapability to adapt to unfavorable regions whereas RESE completes executing for all seeds in all the four environments. Here, DNF stands for ”Did Not Finish”.

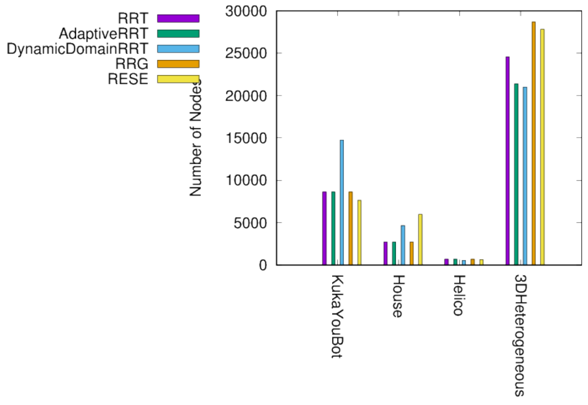
C. Random Multi-Query Experiments

Previously discussed and compared algorithms solve for single query problems or predefined multi query problems. In real-time applications such as search and rescue scenarios, having random goal positions (as in the case of sporadically falling debris which causes the goal positions to change continuously making the environment dynamic) is important. We simulate this scenario by generating random goal positions during the roadmap construction and the algorithm automatically terminates on reaching all the goal positions successfully.

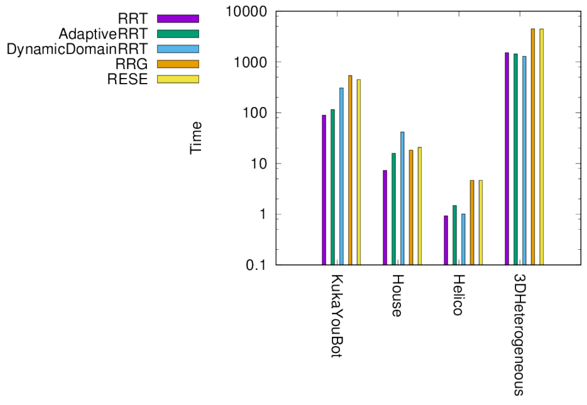
Table III and IV shows the performance of RESE across all experiments performed. This result shows that the random multi query experiments incur little overhead compared to the other experiments and in some cases, outperforms the single and multi query experiments as is the case for 3D Heterogeneous environment and 0.34 seconds slower than the single query experiments for the Helico environment. This shows intuitively how RESE will perform in a more dynamic environment (little overhead, adapt to changes in the environment) which we plan to explore in a future work.

VI. Conclusion

In this paper we have presented a new graph-based algorithm, RESE (Rapidly Exploring random Search Explorer), that plans the path from start to goal on making the connection between nodes through witness nodes and uses reinforcement learning approach to trace the successful witness nodes in the environment. RESE also attempts to connect blocked nodes in the narrow region by fostering witness nodes in different directions for connection. The algorithm has shown the capability to work well in multi-query environments by building paths from a fixed start or random start to the multiple goal positions. For future work, the algorithm will be further modified to test in better dynamic environments and will be integrated into a real-time robot for search and rescue scenarios.



(a) **Single Query Nodes**



(b) **Single Query Time (in seconds)**

Fig. 3

	KukaYouBot	**House 3D**	**Helico**	**Heterogeneous 3D**
PRM	DNF	2748.60	1692.60	DNF
RRG	DNF	2736.10	DNF	DNF
RESE	6314.10	2182.50	773.80	28251.90

TABLE I: **Nodes Generated in MultiQuery environement**

	KukaYouBot	**House 3D**	**Helico**	**Heterogeneous 3D**
PRM	DNF	363.32	49.02	DNF
RRG	DNF	82.99	DNF	DNF
RESE	**564.93**	**52.72**	**12.43**	**2210.77**

TABLE II: **Total time in MultiQuery environement**

RESE	**KukaYouBot**	**House 3D**	**Helico**	**Heterogeneous 3D**
Single query	7655.5	5987.3	654.1	27813.1
Multi-query	6314.1	2182.5	773.8	28251.9
Random multi-query	6314.1	5987.3	675.2	28251.9

TABLE III: **Nodes Generated**

RESE	**KukaYouBot**	**House 3D**	**Helico**	**Heterogeneous 3D**
Single query	450.7268	20.780873	4.624302	4474.5712
Multi-query	564.9383	52.721571	12.435526	2210.7718
Random multi-query	612.4274	66.72147	4.964374	2044.222

TABLE IV: **Total Time taken**

REFERENCES

[1] N. M. Amato. Motion planning benchmarks. http://parasol.tamu.edu/groups/amatogroup/benchmarks/.

[2] 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*, pages 155–168, Natick, MA, USA, 1998. A. K. Peters, Ltd. (WAFR ‘98).

[3] O. Arslan, K. Berntorp, and P. Tsiotras. Sampling-based algorithms for optimal motion planning using closed-loop prediction. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 4991–4996. IEEE, 2017.

[4] O. Arslan and P. Tsiotras. Use of relaxation methods in sampling-based algorithms for optimal motion planning. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2421–2428. IEEE, 2013.

[5] O. Arslan and P. Tsiotras. Incremental sampling-based motion planners using policy iteration methods. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 5004–5009. IEEE, 2016.

[6] Z. Beck, L. Teacy, A. Rogers, and N. R. Jennings. Online planning for collaborative search and rescue by heterogeneous robot teams. In *Proceedings of the 2016 International Conference on Autonomous Agents &#38; Multiagent Systems*, AAMAS ’16, pages 1024–1033, Richland, SC, 2016. International Foundation for Autonomous Agents and Multiagent Systems.

[7] S.-Y. Chien, H. Wang, and M. Lewis. Human vs. algorithmic path planning for search and rescue by robot teams. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 54, pages 379–383. Sage Publications Sage CA: Los Angeles, CA, 2010.

[8] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, June 2005.

[9] J. Denny and N. M. Amato. The toggle local planner for sampling-based motion planning. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1779–1786. IEEE, 2012.

[10] J. Denny, M. Morales, S. Rodriguez, and N. M. Amato. Adapting rrt growth for heterogeneous environments. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1772–1778. IEEE, 2013.

[11] J. Denny, R. Sandström, A. Bregger, and N. M. Amato. Dynamic region-biased rapidly-exploring random trees. In *Twelfth International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016.

[12] R. Gayle, M. C. Lin, and D. Manocha. Constraint-based motion planning of deformable robots. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1046–1053, April 2005.

[13] S. Gottschalk, M. Lin, and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. Technical Report TR96-013, University of N. Carolina, Chapel Hill, CA, 1996.

[14] G. A. Hollinger and G. S. Sukhatme. Sampling-based motion planning for robotic information gathering. In *Robotics: Science and Systems*, volume 3. Citeseer, 2013.

[15] L. Janson, E. Schmerling, A. Clark, and M. Pavone. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International journal of robotics research*, 34(7):883–921, 2015.

[16] M. Kallmann, A. Aubel, T. Abaci, and D. Thalmann. Planning collision-free reaching motion for interactive object manipulation and grasping. *Computer Graphics Forum*, 22(3):313–322, Sept. 2003.

[17] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[18] 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.

[19] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 473–479, 1999.

[20] S. M. LaValle and J. J. Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001.

[21] T. Lozano-Perez. Spatial planning: A configuration space approach. *Computers, IEEE Transactions on*, 100(2):108–120, 1983.

[22] T. McMahon, S. Jacobs, B. Boyd, L. Tapia, and N. M. Amato. Evaluation of the k-closest neighbor selection strategy for prm construction. Technical Report TR12-002, Texas A&M, College Station Tx., 2011.

[23] T. Morita, J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi. Knot planning from observation. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 3, pages 3887–3892, sept. 2003.

[24] R. R. Murphy, S. Tadokoro, D. Nardi, A. Jacoff, P. Fiorini, H. Choset, and A. M. Erkmen. *Search and Rescue Robotics*, pages 1151–1173. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[25] C. A. Rodriguez, J. Denny, S. A. Jacobs, S. Thomas, and N. M. Amato. Blind RRT: A probabilistically complete, distributed RRT. Technical Report TR13-004, Parasol Lab, Dept. of Computer Science, Texas A&M University, Apr 2013.

[26] S. Rodriguez, J.-M. Lien, and N. M. Amato. Planning motion in completely deformable environments. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2466–2471, May 2006.

[27] K. Solovey and D. Halperin. Efficient sampling-based bottleneck pathfinding over cost maps. *arXiv preprint arXiv:1608.00261*, 2016.

[28] K. Solovey and M. Kleinbort. The critical radius in sampling-based motion planning. *arXiv preprint arXiv:1709.06290*, 2017.

[29] C. Suh, B. Kim, and F. C. Park. The tangent bundle RRT algorithms for constrained motion planning. In *13th World Congress in Mechanism and Machine Science*, 2011.

[30] 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.