Proceedings of the ASME 2016 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2016
August 21-24, 2016, Charlotte, North Carolina, USA

DETC2016-60547

RRT-HX: RRT WITH HEURISTIC EXTEND OPERATIONS FOR MOTION PLANNING IN ROBOTIC SYSTEMS

Nahas Pareekutty, Francis James
Robotics Research Center
IIIT Hyderabad
Hyderabad-500032, Telangana,
India
Email: nahas.p@research.iiit.ac.in

Balaraman Ravindran
Department of Computer Science
and Engineering
Indian Institute of Technology Madras
Chennai- 600036, Tamilnadu
India
Email: ravi@cse.iitm.ac.in

Suril V. Shah [*]
Department of Mechanical Engineering
Indian Institute of Technilogy Jodhpur
Jodhpur-342011, Rajasthan
India
Email: surilshah@iitj.ac.in

ABSTRACT

This paper presents a sampling-based method for path planning in robotic systems without known cost-to-go information. It uses trajectories generated from random search to heuristically learn the cost-to-go of regions within the configuration space. Gradually, the search is increasingly directed towards lower cost regions of the configuration space, thereby producing paths that converge towards the optimal path. The proposed framework builds on Rapidly-exploring Random Trees for random sampling-based search and Reinforcement Learning is used as the learning method. A series of experiments were performed to evaluate and demonstrate the performance of the proposed method.

1 Introduction

The research for this paper originated in the search for a motion planner for a non-holonomic dual-arm robotic system (similar to [1]). Such planners are expected to derive an optimal motion plan from the initial configuration to the goal configuration while satisfying certain motion constraints. These systems are characterized by high dimensionality, complex dynamics and non-holonomic constraints. In light of these factors, random sampling-based planners were considered. Random search methods such as Rapidly-expanding Random Trees (RRT) have been proven to be effective in path planning for high-dimensional systems even with the presence of obstacles [2]. Advantages of RRT include rapid exploration and option to include motion constraints directly into the node creation operation. However, solutions provided by RRT are generally sub-optimal [3], which motivated us to search for RRT variants that can achieve optimal results.

In systems where cost of actions in the configuration space is either known or calculable, optimized random search methods like A* [4], BIT* [5] can converge to an optimal solution under certain conditions. However, cost-to-go information or heuristic functions are not always available for most complex systems, making the above-mentioned methods unviable. Difficulty in linearization also prevented use of popular methods like RRT-LQR [6] and Kinodynamic RRT* [2].

Methods like RRT* [7] rely on rewiring connections to trace low cost paths and gradually optimize the solution. However, rewiring is only possible in systems with simple dynamics. In non-holonomic systems, rewiring connections is not always feasible due to the presence of differential constraints. Since the guarantee of optimality in these methods depends on rewiring, optimality is lost upon removing this step. For similar reasons, route planning in graph based methods such as PRM [8] and BIT* are also not a viable option for our system.

[*]Address all correspondence to this author.

In the absence of a cost-to-go information, *informed* and heuristically-guided RRT methods showed promise. Heuristically-Guided RRT (hRRT) [9] proposes biasing Voronoi regions based on the estimated cost of the region. Such a biasing would guide the RRT growth towards the lower cost regions of the workspace. Similar heuristically-guided methods, such as Anytime RRT [10] and RRT-PI [11], have been developed, which do not require accurate cost-to-go information. These are multi-query methods in which multiple RRT solutions (queries) are executed with each query using information acquired through previous queries. They begin with an approximate heuristic to approximate cost that is iteratively improved with every query. Anytime RRT uses heuristics to restrict the area explored by the random search. Here, the heuristics provide an estimate of the value/cost-to-go of each extension. Using this value, potential high cost sub-paths are avoided, leading to greater exploration among the low cost regions of the workspace. RRT-PI proposes replacement of the Euclidean distance used in nearest node search with a value-based measure. Here, the nodes of the solution tree provided by RRT are evaluated using Reinforcement Learning (RL) methods. These nodes are used to estimate the value of new nodes in subsequent queries. The use of RL and a value-based distance measure not only eliminates the need for accurate prior path cost information, but also promotes iterative improvement of the cost estimate.

A drawback of both Anytime RRT and RRT-PI is that search in the configuration space is biased towards the initial solution. In systems where multiple homotopic solutions exist, this may fail to find the alternate solution(s). Also, in Anytime RRT, the performance is dependent on the choice of heuristic, which may not be known for some complex systems. For non-holonomic systems, the path planner should employ a random sampling based search method along with a heuristic that can independently estimate the region cost.

We propose RRT with Heuristic Extend operations (RRT-HX), a method that uses multi-query RRT for random search and RL for learning the cost of regions in the configuration space. In this method, RRT trees are generated and the *cost-to-go* is calculated for each node in the solution path using RL. These cost samples populate the configuration space and serve as guides for subsequent RRT queries to exploit low cost paths. Cost-guided branching is added to the RRT tree generation process to bias the growth towards low cost regions. Such an implementation has the following advantages over existing methods:

The exploratory feature of RRT is used to quickly search the configuration space and identify potential paths and sub-paths.

Learning of cost-to-go using RL implies that this method can be used without any prior information.

The cost-guided branching exploits the learned cost information while RRT random search remains unaffected, aiding identification of homotopic solutions.

The iterative process of exploration-cost evaluation-exploitation can produce increasingly optimal solutions.

The remainder of this document is organized as follows - Section 2 briefly describes the background of various methods used in this paper, Section 3 describes the development history of RRT-HX and the motivating factors behind it, Section 4 details the RRT-HX algorithm, Section 5 presents the experimental results and Section 6 discusses the observations made and conclusions.

2 Preliminaries

As the proposed RRT-HX method builds on RRT and RL, these methods are briefly described in this section.

2.1 Rapidly Exploring Random Trees

Path planning in the RRT method basically consists of a tree that grows from the start node, exploring the configuration space, until a branch reaches the goal node. Each node is a state within the configuration space. Creation of a new node consists of an *extend* operation that transits from an existing node on the tree to a new state. This extend operation is performed by first selecting a random target node within the configuration space. Then, the node on the tree closest to the target node is selected and transitions in the direction of the target are calculated. Such a transition has two advantages. Firstly, the extend operation favors unexplored areas in the configuration space. Secondly, any motion-related constraints can be included in the transition. These constraints could be motion constraints of the system itself or constraints introduced by obstacles in the workspace

The growth of the tree can be influenced by *biasing* regions within the configuration space. For example, in goal biasing, the goal state is chosen as target instead of a random node in the *extend* operation. This helps in guiding the search towards biased regions. In most RRT implementations, goal biasing is performed periodically to accelerate the search for solutions.

2.2 Reinforcement Learning

Reinforcement Learning is a learning method that uses interaction with the system to learn its characteristics [12]. The system is modeled as a Markov Decision Process (MDP) consisting of states and actions. The states describe the status of the system at any point of time and actions determine the transitions from one state to other. The effectiveness of an action is defined by its *quality* which includes both the immediate result of the action and the expected consequences further ahead in the future. The immediate result is alternatively defined as a *reward* and the future costs are defined as the *value* of the new state created by the action. Since a state can be followed by multiple actions, the value (or cost-to-go) of a state is determined by the quality of the various actions and the probability of performing each of those actions.

Consider a state s and an action a permissible from state s, resulting in a new state s'. The mapping between states and actions is given by policy π. The value of the state s while following policy π is given by

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P^{a}_{ss'} [R^{a}_{ss'} + \gamma V^{\pi}(s')] \tag{1}$$

where $\pi(s,a)$ gives the probability of choosing action a at state s as defined by the policy, $P^{a}_{ss'}$ is the probability of transitioning to state s' from s on performing action a, $R^{a}_{ss'}$ is the reward for the transition and γ is the discount factor. Future rewards are accumulated using the calculated value of the next state, $V^{\pi}(s')$.

TD(λ) algorithms are well-suited for sampling-based estimation of state values. Here, temporal differences are used to update state values with λ $(0 \leq \lambda \leq 1)$ determining the influence of successive rewards in the trajectory. The return of length m is weighted by $(1-\lambda)\lambda^{m-1}$, where m is the step number. The total backup is given by

$$R^{\lambda}_{t} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{m-1} R^{m}_{t} \tag{2}$$

where R^{m}_{t} is the return at step m. Multiplication by $(1-\lambda)$ is performed to ensure that the sum of weights is 1. The temporal difference update for a state, $V^{\pi}(s)$ is given by

$$\Delta V^{\pi}_{t}(s) = \eta [R^{\lambda}_{t} - V^{\pi}_{t}(s)] \tag{3}$$

where η is the learning rate.

3 Evolution of Design

RRT-HX was developed as a solution for optimal path planning for high-dimensional, non-holonomic robots. To derive the sequence of actions from the current position to a required goal configuration in a n-DOF robot, the planner must handle transitions of an n-dimensional configuration while maintaining the restrictions of the robot motion. Devising a mathematical planner would have been both computationally intensive and time-consuming. RRT methods are known to be effective in high-dimensional systems due to rapid exploratory behavior and bias towards the unexplored regions of the configuration space. We therefore modeled the system as a path planning problem with RRT as the solver. This combination was able to deliver a feasible, but sub-optimal, solution in a relatively short duration.

To improve the quality (or reduce the cost) of the solution, Euclidean distance used in closest node identification was replaced by a local cost function that chose the node that produced the closest extension towards the random node. Although this method improved the quality of the solutions, there was no guarantee of optimality. This is due to the fact that reduction in local costs need not guarantee cost reduction over the entire solution path. An attempt was made to implement RRT* which required approximations in rewiring newly created nodes to existing nodes in the tree. These approximations were necessary since ideal connections could not be made from new nodes to existing nodes due to the above-mentioned motion constraints. However, the errors introduced in this process were unacceptably high. Since most Random Geometric Graph methods and Probabilistic Road Map methods also rely on rerouting between nodes, these methods would also be unsuitable for our application.

In the proposed work, since the path was to be designed for systems that did not have a known cost-to-go estimate, we had to ensure that the cost of regions within the configuration space was learned directly through experience. Also, the path search method should be able to use the cost information to trace potential regions for path planning. It was decided to develop a multi-query RRT method with heuristic extend operations guided by quality values of previous samples, and the method is referred to as RRT-HX hereafter. This heuristic extend or quality-guided extend operation forms the fundamental contribution of this work. To promote search for multiple homotopic solution paths, it was decided that random extend operations would remain unaffected by quality bias and biased extensions would be a separate operation. This is similar to methods employed in RL. Initially, it was proposed to discretize the configuration space into a *quality grid* which would contain the cost/quality values at each point in the grid. These grid points would be updated after each RRT query using TD(0) methods of RL. Quality of new nodes would be calculated based on their neighboring grid points. However, due to huge memory requirements, this was replaced with a sampling-based method which only stored nodes generated in RRT queries. This is one of the novelties of the proposed algorithm. After each query, rewards are assigned to each step in the solution path using TD(λ) method using each RRT solution path as an episode, as described in Eq.(2).

Modifications to the RRT algorithm for implementation of RRT-HX would primarily require changes in the extend operation. Here, in addition to the random node extend operation and goal biased extend operation of RRT, heuristic extend operation was added. This new step involved selection of a random node on the tree and extending it towards a high quality region in its neighborhood. The number of random extend operations and heuristic extend operations in a cycle is determined by an Exploration:Exploitation ratio which is gradually reduced with each query similar to RL methods. As the number of samples increase and queries become increasingly exploitative, latter trajectories generated by RRT are expected to be more optimal than the earlier ones. We therefore attempt to *forget* samples generated by older trajectories by periodically decaying their values. This is another novelty of the algorithm.

Since a solution is always available after the first query, RRT-HX is an anytime solution. However, an added benefit in cost was observed if an additional RRT query with only heuristic extend operations (no random extend operations) is executed after termination condition. When sufficient samples are available, this query traces the high quality path from start to goal in a very short duration. If there are no strict time constraints, such an additional step can prove beneficial.

To summarize, RRT-HX offers a novel method for cyclically generating trajectories that serve as new learning episodes while reusing cost information from older episodes to iteratively improve the path cost.

4 Algorithm

This section presents RRT-HX algorithm which is fundamental contribution of this work. RRT-HX algorithm can be functionally divided into two parts - quality-guided trajectory generation using RRT and trajectory evaluation using RL. Their systematic implementation is discussed below.

4.1 Quality-guided RRT for Trajectory Generation

The RRT-HX algorithm (described in Alg.1) executes multiple RRT queries until the termination condition has been achieved. Post each query, value of the nodes in the solution path is calculated using TD(λ) method and update of the quality samples (*QSamples*) is performed.

Algorithm 1 Main RRT-HX Function

```
1:  procedure RRT-HX
2:      RRTree ← []
3:      QSamples ← []
4:      while TerminationCondition != TRUE do
5:          SolPath ← RRTQuery()
6:          QSamples ← QUpdate( SolPath, QSamples )
7:      if MAND_ANYTIME == TRUE then
8:          qSolPath ← SolPath
9:      else
10:         qSolPath ← QualBiasQuery()
        return qSolPath
```

The **TerminationCondition** function returns TRUE when termination condition has been reached. The termination condition could be time, number of queries or cost improvement threshold. The MAND_ANYTIME configuration parameter holds true if strict anytime solution is required and false if it is not applicable. **QualBiasQuery** performs a quality-biased sequence of steps from start to goal using quality values from *QSamples*. No random operations are performed in this case. This is similar to executing **RRTQuery** with sample type always set to quality-biased samples.

RRTQuery (Alg.2) is similar to the generic tree building in RRT, with the added step of **QExtend** which adds branches from *RRTree* towards high quality regions. Selection of which sample type to use -Random, Goal or QualityBiased - is based on goal bias ratio and exploration to exploitation ratio. The extend operations return a parent node and target node. **GenTransition** generates a transition from the parent node in the direction of the target node, ending in a new node.

Algorithm 2 RRT Trajectory Generation

```
1:  procedure RRTQUERY
2:      repeat
3:          SampleType ← GetSampleType()
4:          if SampleType == 'RandomNode' then
5:              [ TargetNode, Parent ] ← Extend()
6:          if SampleType == 'GoalBias' then
7:              [ TargetNode, Parent ] ← ExtendGoal()
8:          if SampleType == 'QualityBias' then
9:              [ TargetNode, Parent ] ← QExtend()
10:         NewNode ← GenTransition( TargetNode,
11:                                          Parent )
12:         RRTree ← AddNode( RRTree, NewNode,
13:                                          Parent )
14:     until distance( GoalNode,NewNode ) >
15:                                          GOAL_THRESH
        return RRTree
```

The **distance** function returns the Euclidean distance between the two input nodes. **GetSampleType** function returns the type of the next sample. The sample type is determined by the configuration parameters, goal bias and exploration:exploitation ratio. The **Extend** function is the same Extend operation of RRT, which finds a random node and extends the closest node on the tree towards it. In **ExtendGoal** operation, the goal is chosen instead of random node.

The primary contribution of the RRT-HX algorithm is the use of knowledge acquired during the learning process to bias the growth of the tree generated by RRT. This is done by **QExtend** (Algo.3) which performs quality-biased extend operations. Such operations are performed only on nodes for which **QExtend** has not yet been performed to ensure that redundant extensions are not made. Such nodes are identified by the state of the

Extend flags in *RRTree*. In the **QExtend** function, the list of new nodes are retrieved from *RRTree* and a random node is chosen. This node is extended towards the highest quality sample from *QSamples* in its neighborhood as demonstrated in Fig.1. The

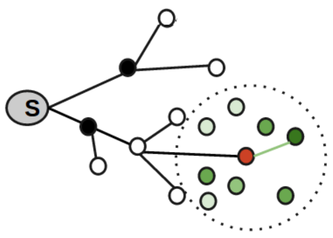

FIGURE 1: **Quality-biased extend operation**. Random node (red) is selected from nodes that have not yet been extended (white). The highest quality node among quality samples in the neighborhood (green) is found and the extension is performed towards it.

Algorithm 3 Quality-Biased Extend

```
1: procedure QEXTEND
2:     NewNodes ← GetUnextNodes()
3:     RandNode ← rand( NewNodes )
4:     NewNode ← GetNeighHQ( RandNode )
5:     SetExtendFlag( RandNode )
       return NewNode, RandNode
```

GetUnextNodes function returns nodes from *RRTree* for which Extend flag is not set. **SetExtendFlag** sets the Extend flag of the input node in *RRTree*.

GetNeighHQ (described in Alg.4) searches the neighborhood of the input random node and returns the sample from *QSamples* that has the highest quality.

Algorithm 4 Find High-Quality Node in Neighborhood

```
1: procedure GETNEIGHHQ( RandNode )
2:     MaxQ ← []
3:     NeighNodes ← GetNeighNodes( RandNode )
4:     for each NNode in NeighNodes do
5:         NodeQ ← GetQuality( NNode )
6:         if isempty( MaxQ ) || NodeQ > MaxQ then
7:             MaxNode ← NNode
8:             MaxQ ← NodeQ
       return QNode
```

The **GetNeighNodes** function returns nodes from *QSamples* that are in the neighborhood of the input node. **GetQuality** returns the quality of the input node from *QSamples*.

4.2 RL for Trajectory Evaluation

This section describes the functions that form the *learning* part of the algorithm. They are responsible for evaluation of the every new trajectory using RL and update of existing quality samples. This is the secondary contribution of RRT-HX algorithm. For every query returned by **RRTQuery**, quality values are updated for the current trajectory using TD(λ) method. As described in Section 3, decay of older values in *QSamples* is also performed to progressively favor values obtained from new episodes. **QUpdate** (Alg.5) is called from the main function **RRT-HX** (Alg.1)

Algorithm 5 Quality Update of Trajectories

```
1: procedure QUPDATE( SolPath, QSamples )
2:     QNewSamples ← TDRewardTrajectory( SolPath )
3:     DecayOldQ( QSamples )
4:     QSamples ← AddNewQ( QSamples,
5:                                 QNewSamples )
       return QSamples
```

The nodes in the new solution path are rewarded with a high positive goal reward for the final node (near the goal) and a decayed reward is applied to each parent node in the path. This is performed by **TDRewardTrajectory** as defined in Eq.(3). Also, the values of the existing samples are decayed using the **DecayOldQ** function

5 Experimentation

In order to demonstrate efficacy of the proposed approach, two experiments were performed to test RRT-HX, namely, 2D Terrain with Varying Cost Regions and Two-Link Under-actuated Pendulum.

5.1 Motion Planning over 2D Terrain with Varying Cost

We used this experiment to validate the RRT-HX algorithm and to analyze the influence of various parameters. Also, this offers a good graphical representation of the convergence of the RRT-HX method. This experiment consists of motion planning for a simulated point robot over a 2D terrain with varying cost regions. The speed of the robot through a region decreases with the cost of

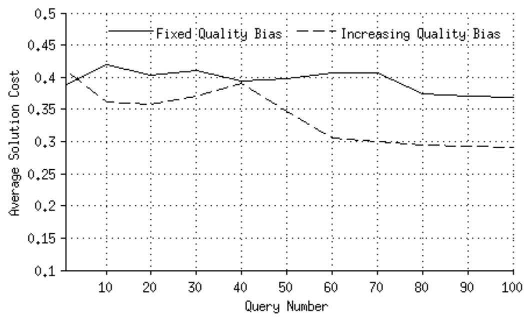

FIGURE 2: Comparison of performance with fixed and increasing quality bias for Motion Planning in 2D Terrain

the region. The global cost of a trajectory is therefore a function of time taken to reach the goal.

As seen in RL implementations, solution cost improvement was observed when quality bias was gradually increased as the number of samples increased. Experiments with constant and high value of quality bias showed no significant improvement over successive queries, implying lack of exploration. The cost variation for fixed and increasing quality bias is shown in Fig.2. The experiment was conducted with goal bias of 1%, quality bias increase at 10% every 10 queries and η of 0.1. A trajectory generated by RRT* in 30,000 steps was used as the baseline for the test. We observed that RRT-HX was able to find increasingly optimal (lower cost) solutions with each query and decreasing exploration:exploitation ratio. The paths converged to a cost of 0.291 against the baseline cost of 0.287. The results are shown in Fig.3.

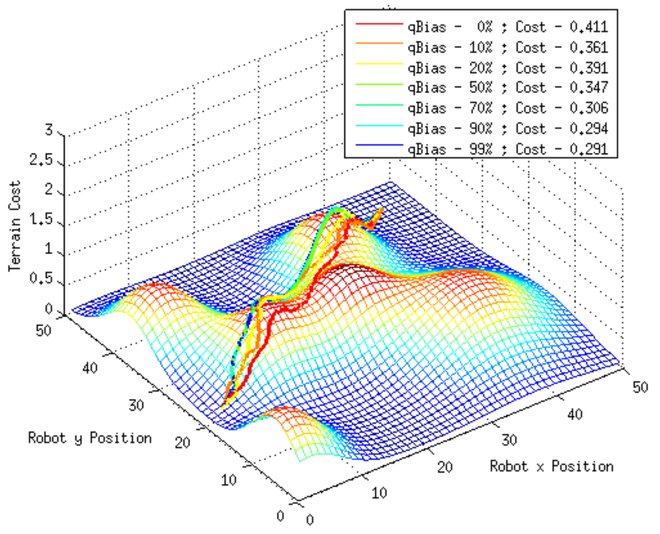

FIGURE 3: Paths generated by RRT-HX in 2D Terrain

5.2 Balancing Under-Actuated 2 Link Pendulum

The 2-link pendulum system consists of two arms connected by a powered joint and one of the links attached to a fixed point through an unactuated joint. The pendulum starts in the lower position with both links vertically down. The aim of the control is to achieve balance in the inverted position using application of torque in the actuated joint [13]. Experiments were conducted with both Anytime RRT (with Euclidean distance as heuristic) and RRT-HX. Using RRT-HX, we were able to improve both the query time (time taken to find a solution) and solution time (time taken to achieve balance).

In this implementation, the state consists of 5 variables - two joint angles, their respective angular velocities and the action (torque) applied to reach the state. During the quality-biased extend operation, the torque associated with the high quality node is used as control action to perform the extension. The quality bias was increased at 1% for every 10 queries and TD(1) was used for the update. Goal bias was fixed at 10%. Time taken to achieve balance was chosen as the cost for this experiment. The cost improvement seen in RRT-HX was significantly higher than Anytime RRT. The comparative results are shown in Fig.4.

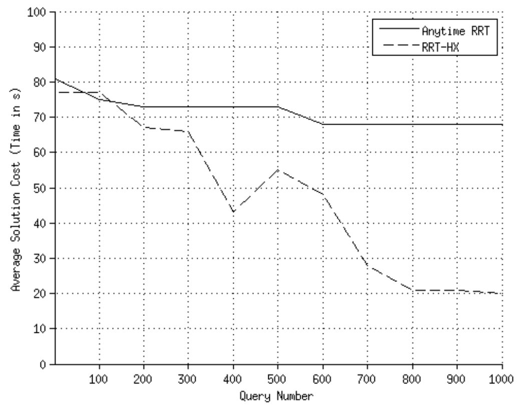

FIGURE 4: Comparison of Anytime RRT and RRT-HX for Balancing of Under-actuated 2-Link Pendulum

6 Conclusion

We present RRT-HX as an effective planner that can provide iteratively optimal solutions for robotic systems by combining exploratory search using RRT, cost-to-go estimation using RL and identification of low-cost paths using quality-biased exploitation of cost information. We have demonstrated that this method can be used in complex systems to identify and improve solution paths even in the absence of any cost information or known heuristic. The experiments also helped to establish the importance of the quality bias % and the impact of its variation on the path solving process.

Future work in this method includes automatic tuning of RRT-HX parameters based on success/failure, cost improvement, and sample density. We would also like to identify methods to reuse the cost-to-go samples in the configuration space even when the system has undergone changes.

REFERENCES

[1] Hafez, A. H., et al., "Reactionless visual servoing of a dual-arm space robot." *IEEE International Conference on Robotics and Automation (ICRA)*, 2014

[2] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning", *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378-400, May 2001.

[3] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846-894, June 2011.

[4] P. E. Hart, N. J. Nilsson, and B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *TSSC*, 4(2): 100107, Jul. 1968

[5] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, BIT*: Batch informed trees for optimal sampling-based planning via dynamic programming on implicit random geometric graphs, *Autonomous Space Robotics Lab, University of Toronto*, TR-2014-JDG006, 2014.

[6] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, LQR-RRT : Optimal sampling-based motion planning with automatically derived extension heuristics, *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2012, pp. 2537-2542.

[7] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning", arXiv preprint arXiv:1005.0416

[8] S. Karaman and E. Frazzoli, Sampling-based algorithms for optimal motion planning", *The International Journal of Robotics Research (IJRR)*, vol. 30, no. 7, pp. 846894, 2011.

[9] C. Urmson and R. Simmons, Approaches for heuristically biasing RRT growth", *IROS*, 2: 11781183, 2003.

[10] D. Ferguson and A. Stentz, "Anytime RRTs", *International Conference on Intelligent Robots and Systems*, October 2006

[11] M. S. Sivamurugan and B. Ravindran, "RRTPI: Policy Iteration on Continuous Domains using", *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[12] R.S. Sutton and A.G. Barto, "Reinforcement Learning: An Introduction", MIT Press, 1998.

[13] M. W. Spong, "The Swing Up Control Problem For The Acrobot", *IEEE International Conference on Robotics and Automation*, 1994