VF-RRT: Introducing Optimization into Randomized Motion Planning

Inyoung Ko
School of Mechanical and Aerospace Engineering
Seoul National University
Seoul, Korea
Email: iyko2001@gmail.com

Beobkyoon Kim
School of Mechanical and Aerospace Engineering
Seoul National University
Seoul, Korea
Email: beobkyoon.kim@gmail.com

Frank Chongwoo Park
School of Mechanical and Aerospace Engineering
Seoul National University
Seoul, Korea
Email: fcp@snu.ac.kr

***Abstract*—The Vector Field Rapidly-exploring Random Tree (VF-RRT) algorithm is an extension of the RRT algorithm for planning in the presence of vector fields; its main distinguishing feature is that random nodes are generated in such a way that the trees tend to extend along the directions of the given vector field. By constructing vector fields to be aligned in the direction that minimizes the upstream cost, which is a new criterion for measuring the extent to which a path moves against the vector field flow, the VF-RRT algorithm can be used to efficiently generate nearly optimal paths while remaining with a probabilistic planning setting. Experimental results comparing our paths with those produced by the T-RRT algorithm and the basic RRT algorithm are presented.**

I. Introduction

The Rapidly-exploring Random Tree (RRT) algorithm [1], [2] and its many variants are used widely because it possesses, among other things, the probabilistic completeness property, and also produces paths efficiently. The basic RRT algorithm in its present form, however, is unable to generate paths that are optimal or nearly optimal, by virtue of its random sampling nature and its emphasis on quickly finding a feasible path.

Some recent works have tried to find better quality paths using sampling-based methods. The heuristically guided Rapidly-exploring Random Tree [3] and Anytime Rapidly-exploring Random Tree [4] are representative of these efforts. However, these algorithms require a heuristic cost to goal, and the ability to lower costs degrades considerably as the configuration space dimension increases.

The successful RRT-based algorithm that produces better quality paths (in the sense of lowering a quantitative cost associated with each path) is the Transition-based Rapidly-exploring Random Tree (T-RRT) algorithm [5]. The gradienT-RRT algorithm [6] can be regarded as an extension that addresses one weakness of the T-RRT algorithm, namely, that it does not extend well in narrow-shaped cost distributions. They uses the statistical test (called a transition test in [5]), which decreases the probability of node creation exponentially with increasing the mechanical work. Since acceptance or rejection of a new node creation is determined by the transition test, they often often brings about inefficient exploration by rejecting too many potential nodes.

Another successful algorithm is the RRT* algorithm of [7]. It uses tree refinement methods, that is, it modifies the branches into a given node within a fixed radius. It is the first algorithm in which the generated path provably converges to the optimal one as the number of samples increases to infinity. RRM algorithm proposed in [8] found by converting subtrees into roadmaps. While both of them appeals property of converging to the optimal path as the number of samples increases, the optimization proceeds at a very slow rate when there are only a few branches in the configuration space.

To address some of the deficiencies of the present algorithms pointed out above, in this paper we propose a new sampling-based algorithm, the Vector Field Rapidly-Exploring Random Tree (VF-RRT) algorithm. The VF-RRT is an extension of the RRT algorithm for planning in the presence of vector fields; its main distinguishing feature is that random nodes are generated in such a way that the trees tend to extend along the directions of the given vector field. First, we propose a new criterion for path quality, which we call the **upstream criterion**, that measures the extent to which the path goes upstream against the optimal (or desired) vector field. The criterion, expressed as a functional, is invariant with respect to the parametrization of the path and also defined continuously. In conservative vector field case, by constructing vector fields to be aligned in the direction that minimizes some physical cost, the VF-RRT algorithm can be used to efficiently generate good quality paths based on upstream criterion while remaining within a probabilistic planning setting. By doing so, the VF-RRT algorithm inherits the probabilistic completeness properties of the basic RRT algorithm. Moreover, the VF-RRT algorithm has an efficient exploration ability since it does not reject potential nodes. The number of iterations tends to be considerably lower, resulting in greater computational efficiency.

This paper is organized as follows. Section II provides the definition of upstream performance criterion and compares features of minimum upstream paths with those of others. Section III describes our sampling-based planning algorithm based on the upstream criterion. Section IV presents results of numerical experiments with our algorithm, including comparisons with alternative planning algorithms. Conclusions are presented in Section V.

II. Definition of Minimum Upstream Path

Before defining the upstream path criterion, we first fix some notation, and review two basic criteria for path quality

978-1-4673-5769-2/13/$31.00 ©2013 IEEE

used in existing biased sampling approaches. Let $\mathcal{Q} \subseteq \mathbb{R}^n$ denote the configuration space, $\mathcal{Q}_{\text{obs}} \subseteq \mathcal{Q}$ the region occupied by obstacles, and $\mathcal{Q}_{\text{free}} \subseteq \mathcal{Q}$ the free space. Given initial and final configurations q_{init} to q_{final}, let $q : [0, L] \to \mathcal{Q}_{\text{free}}$ be an arc-length parametrized $\mathcal{C}^1$ curve connecting q_{init} and q_{final} (i.e., L is the arclength, $q(0) = q_{\text{init}}$ and $q(L) = q_{\text{final}}$, and at all points on the arclength-parametrized curve $q(s)$, the unit speed condition $\|dq/ds\| = 1$ holds). We further assume a $\mathcal{C}^1$ vector field $F : \mathcal{Q} \to T\mathcal{Q}$ is given that describes, at each point in the configuration space, an optimal or desired direction of motion. If F is conservative, then by definition $F(q) = -\nabla C(q)$ for some $\mathcal{C}^1$ potential function $C : \mathcal{Q} \to \mathbb{R}$.

Given a path q as characterized above, we can recall its work in the presence of a vector field F. There are some well-known defects associated with using this notion of work as a measure of path quality. For example, if the vector field F is conservative, the criterion is unable to distinguish between paths that have the same endpoints—the work is path-independent and given by $C(q_{\text{final}}) - C(q_{\text{init}})$. To overcome the deficiencies associated with the standard physics notion of work, [5] proposed an alternative definition of work, called **mechanical work** and denoted *MW*, which begins by assuming a choice of $\mathcal{C}^1$ cost function $C : \mathbb{R}^n \to \mathbb{R}$. Setting the gradient $-\nabla C(q)$ to be the vector field $F(q)$, the mechanical work criterion can then be written in the form

$$MW(q) = \int_0^L \sigma(q(s))\, ds + \epsilon \int_0^L ds,$$

$$\sigma(q(s)) = \begin{cases} \langle -F(q(s)), \dot{q}(s) \rangle & \text{if } \langle -F(q(s)), \dot{q}(s) \rangle > 0 \\ 0 & \text{otherwise,} \end{cases}$$

where ϵ is some small constant, and $\langle \cdot, \cdot \rangle$ denotes the standard Euclidean inner product in $\mathbb{R}^n$. While [5] does not explicitly do so, it is straightforward to generalize the mechanical work criterion to nonconservative vector fields (simply replace $-\nabla C(q)$ by the given nonconservative vector field). The *MW* criterion further possesses several attractive properties that make it preferable to using the standard path integral of the cost. It is not smooth, however, and the choice of ϵ is ad hoc. Moreover, the fact that for linearly varying $C(q)$ the minimum MW paths correspond to straight lines is not necessarily natural or always desirable from a physical perspective, as we argue later below.

To supplement defects mentioned above, we now define our upstream criterion. The upstream criterion measures the extent to which a path $q(s)$ goes against the vector field $F(q(s))$. The upstream functional is defined as follows:

Definition 1: Given a unit-speed path $q : [0, L] \to \mathcal{Q}_{\text{free}}$ and a $\mathcal{C}^1$ vector field $F : \mathcal{Q} \to T\mathcal{Q}$, the **upstream** functional $\mathcal{U}(q)$ is defined as follows:

$$\mathcal{U}(q) = \int_0^L \left(\|F(q(s))\| - \langle F(q(s)), \dot{q}(s) \rangle \right) ds. \qquad (1)$$

The first term of the integrand is a result of the unit speed assumption $\|\dot{q}\| = 1$; the corresponding definition for non-unit speed curves follows straightforwardly.

We now compare qualitative features of minimum upstream paths with those of minimum work and minimum mechanical work paths. All optimal paths are obtained by the RRT* algorithm of [7].

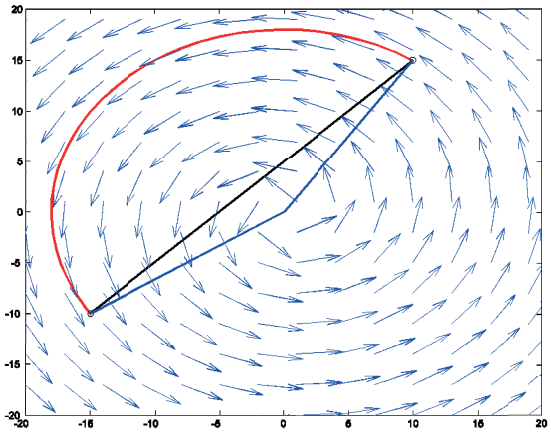

Fig. 1. Nonconservative (rotational) vector field: the minimum work path (blue) is piecewise linear via the center point, and the minimum mechanical path (black) is a straight line, while the minimum upstream path is clearly more aligned with the vector field than the other paths (red).

- In the first example we consider the nonconservative rotational vector field of Figure 1. The minimum work and mechanical work path are indicated in blue and black each, while the minimum upstream path is indicated in red. The minimum work path is a broken line that passes through the center and the minimum *MW* path becomes a straight line. The minimum upstream path is clearly more aligned with the vector field than the others. So, it can easily know the fact that to follow the upstream paths requires less efforts than others.
- In the second example, the configuration space is the two-dimensional *x*-*y* plane, and the value of the cost function $C(x, y)$ consists of three connected planes of different slope (see Figure 2). The black line segment is the minimum *MW* path and the red line is the minimum upstream path. While the former projects as a straight line in configuration space (the *x*-*y* plane), the latter becomes a piecewise linear path. That is, contrary to the minimum upstream path, the minimum *MW* path can not reflect the slopes of surfaces. The minimum upstream path is a lot more natural intuitively by imagining something comes down the slopes. The minimum work path are any paths that connect q_{init} with q_{final}.

In summary, minimum upstream paths are very natural and qualitatively closer to the most efficient path than minimum *MW* paths and minimum work paths in vector fields.

III. Algorithm for Minimum Upstream Paths

In [5], it is shown that if the probability that an edge extends toward any direction decreases exponentially with the edge's cost increment, then among paths of the same length, the path which has the smallest cost has the highest probability of being selected as the final path. With this fact, a new

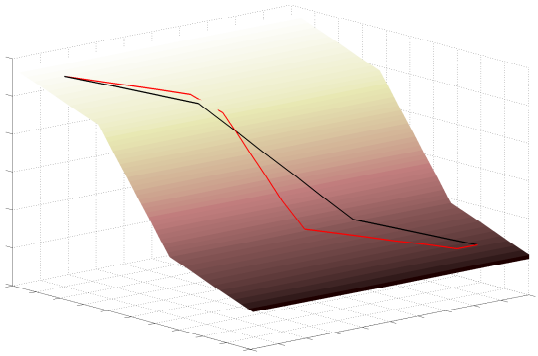

Fig. 2. While the minimum *MW* path (black) becomes a straight line in the *x*-*y* plane, the minimum upstream (red) is piecewise linear.

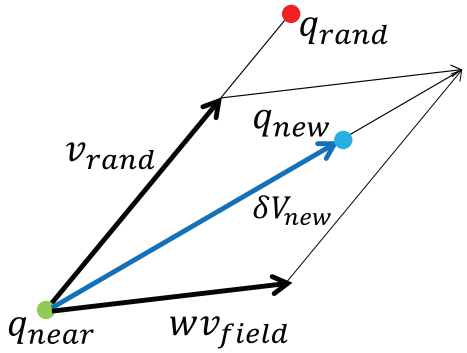

Fig. 3. Determining a new node in the VF-RRT algorithm: a new direction is determined from the sum of v_{rand} and wv_{field}. The new node is obtained by extending, from q_{near}, toward v_{new} by amount of δ.

algorithm is proposed in this section; in which the edge cost is set to be the upstream cost. Whereas existing algorithms have an explicit test in which new nodes are rejected or accepted, our algorithm does not have an explicit node acceptance-rejection step and does not require finding nearest k-nodes, computation times are considerably reduced.

A. Main Steps of Algorithm

The proposed algorithm extends trees in a similar way as the EXT-EXT version of the basic RRT algorithm; the difference is when trees extend their branches to explore the configuration space, the optimal (or desired) vectors $F(q(s))$ bias the selection of random directions in such a way that the trees explore the directions indicated by the vector field prior to other directions. This is the main concept behind the new algorithm, which we call the Vector Field RRT (VF-RRT) algorithm.

First, let us assume the initial configuration q_{init} and final configuration q_{final} are given. The initial configuration becomes the root node of a tree. Then, we generate a random node q_{rand} in the space, and select the nearest neighbor node q_{near} from q_{rand} in the tree. The following step determines a unit vector v_{rand} that indicates from q_{near} to q_{rand}. We need one more direction, which is v_{field} from the vector field. Let the vector from the vector field at q_{near} be F, the $v_{\text{field}} = F/\|F\|$.

TABLE I. VECTOR FIELD RRT ALGORITHM

	VF-RRT(q_{init}, q_{final}, field)
1	Tree
2	Tree.AddVertex(q_{init})
3	**while** $i < I_{\max}$ **do** $i \leftarrow i+1$
4	$q_{\text{rand}} \leftarrow$ RandomSampling()
5	$q_{\text{near}} \leftarrow$ FindNearestNodeOnTree(Tree, q_{rand})
6	$v_{\text{new}} \leftarrow$ GetChangedDirection(q_{near}, q_{rand}, field, Tree)
7	$q_{\text{new}} \leftarrow$ Extend(Tree, q_{near}, v_{new})
8	**if** Connect(Tree, q_{final})**then**
9	**return** Path(Tree)
10	**end**
11	**end**
12	**return** NULL

TABLE II. FUNCTION: GETCHANGEDDIRECTION

	GetChangedDirection(q_{near}, q_{rand}, field)
1	$v_{\text{rand}} \leftarrow \frac{q_{\text{rand}} - q_{\text{near}}}{\|q_{\text{rand}} - q_{\text{near}}\|}$
2	$F \leftarrow$ GetVectorFieldVector(q_{near}, field)
3	$w \leftarrow$ GetWeight(v_{rand}, field, F)
4	$v_{\text{field}} \leftarrow \frac{F}{\|F\|}$
5	$v_{\text{new}} \leftarrow \frac{v_{\text{rand}} + w v_{\text{field}}}{\|v_{\text{rand}} + w v_{\text{field}}\|}$
6	**return** v_{new}

The next step is to determine the weight w of v_{field} against v_{rand} (The method for determining w is presented in detail at next section). Then, the direction in which the tree extends is determined as follows:

$$v_{\text{new}} \leftarrow \frac{v_{\text{rand}} + w v_{\text{field}}}{\|v_{\text{rand}} + w v_{\text{field}}\|}. \tag{2}$$

The position of the new candidate node q_{new} is given by extending by as much as the stepsize (δ) from q_{near} in the direction of v_{new}. That is, $q_{\text{new}} = q_{\text{near}} + \delta v_{\text{new}}$.

If $q_{\text{new}} \subset \mathcal{Q}_{\text{free}}$, q_{new} is connected with q_{near} as a member of the tree. This process is repeated until the distance between q_{new} and q_{goal} is less than the connectivity criterion.

B. Biased Sampling

Let us consider the plane spanned by v_{rand} and v_{field}. In this plane v_{new} is the vector between v_{rand} and v_{field}. The factor w is a weight value that describes how much to draw v_{new} toward the v_{field} direction. As w becomes bigger, v_{new} in (2) tends to be directed toward v_{field}. Let us define θ_{rand} as the angle between v_{field} and v_{rand}, and further define θ_{new} as the angle between v_{field} and v_{new}. The factor w is then determined by the sine law (see Figure 4):

$$w = \frac{\|v_{\text{rand}}\| \sin(\theta_{\text{rand}} - \theta_{\text{new}})}{\|v_{\text{field}}\| \sin(\theta_{\text{new}})}.$$

TABLE III. GETWEIGHT FUNCTION

	GetWeight(v, field, F)
1	$\theta_{\text{rand}} \leftarrow \arccos(\frac{<F, v>}{\|F\|})$
2	$U_{\text{rand}} \leftarrow \|F\|\{1 - \cos(\theta_{\text{rand}})\}$
3	$U_{\text{new}} \leftarrow -\frac{log\{1 - \frac{U_{\text{rand}}}{2\|F\|}(1 - e^{-2\lambda\|F\|})\}}{\lambda}$
4	$\theta_{\text{new}} = \arccos(1 - \frac{U_{\text{new}}}{\|F\|})$
5	$w \leftarrow \frac{\sin(\theta_{\text{rand}} - \theta_{\text{new}})}{\sin(\theta_{\text{new}})}$
6	**return** w

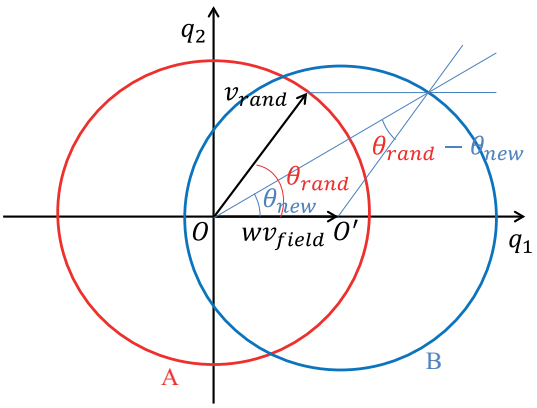

Fig. 4. The circle A shows the possible positions after extending one step from O via the basic RRT, while the point O' (center of circle B) shows the location after moving by wv_{field} from O. The weight w can be determined by θ_{rand} and θ_{new}.

Both v_{rand} and v_{field} are unit vectors, and θ_{rand} can be obtained by random sampling. Therefore, leaving thing is only θ_{new}. For obtaining θ_{new}, we first define the local upstream function U to be the integrand of the upstream functional (1):

Definition 2: The **local upstream function** $U : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is defined to be

$$U(v_1, v_2) = \|v_1\| \cdot \|v_2\| - \langle v_1, v_2 \rangle. \tag{3}$$

Let us assume $F\big(q(s)\big)$ is constant F over a small area, then, the local upstream functions U_{rand} and U_{new} can be defined from (3), e.g., $U_{\text{rand}} = \|F\| \cdot \|v_{\text{rand}}\| - \langle F, v_{\text{rand}} \rangle = \|F\|\{1 - \cos(\theta_{\text{rand}})\}$ and $U_{\text{new}} = \|F\| \cdot \|v_{\text{new}}\| - \langle F, v_{\text{new}} \rangle = \|F\|\{1 - \cos(\theta_{\text{new}})\}$. If the relation between U_{rand} and U_{new} is defined, θ_{new} can be determined by θ_{rand}. For making the extending direction distribution of the edges into the exponential distribution with respect to U_{new}, we set the relation between U_{new} and U_{rand} as follows:

$$U_{\text{new}} = -\frac{log\{1 - \frac{U_{\text{rand}}}{2\|f\|}(1 - e^{-2\lambda\|f\|})\}}{\lambda},$$

where λ is a rate parameter of the distribution. This formula is from the inverse cumulative distribution function of the exponential distribution. As a result, θ_{new} can be determined by $\theta_{\text{new}} = \arccos(1 - U_{\text{new}}/\|F\|)$.

The only remaining thing is to determine the parameter λ. It has to reflect whether trees explore the space well or not. As λ approaches zero, the VF-RRT algorithm becomes similar to the basic RRT algorithm. Whereas, as λ becomes larger, the tree tends to extend toward the vector field direction. Clearly as the configuration space contains the more obstacles, it is desirable to set the lower λ. A large λ in complex environments can cause inefficient exploration sometimes.

IV. Case Studies and Discussion

In this section, we present results of numerical experiments with the VF-RRT algorithm. We begin with an example that examine the shape of VF-RRT trees in a nonconservative vector field. We also apply the VF-RRT algorithm in conservative vector fields. We compare the performance results of the VF-RRT algorithm to those of the basic RRT and T-RRT algorithm. We use the bi-directional RRT version and MPNN algorithm [9] for all test (including the basic RRT and T-RRT algorithm), and all algorithms are run 20 times each.

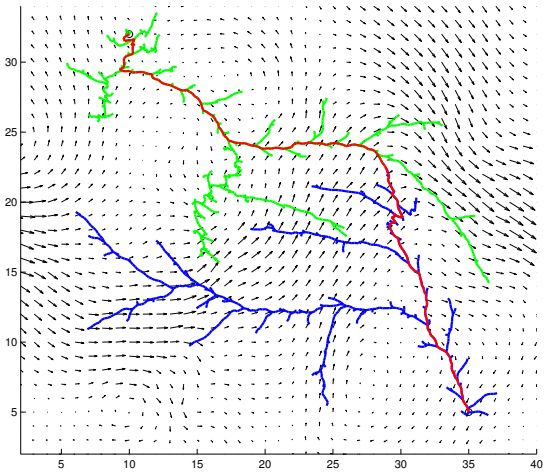

Fig. 5. The shape of VF-RRT trees in a wind field (Green: the tree from the start position, blue: the tree from the goal position, red: final path).

TABLE IV. Wind Field

	Upstream	Iterations	Length	Time(s)
RRT	354.2	231.7	183.1	0.02
T-RRT	219.3	795.3	195.0	0.08
VF-RRT	156.5	709.6	198.9	0.09

A. Nonconservative Wind Velocity Field

In the first example, paths are planned in a nonconservative wind velocity field. We apply the VF-RRT algorithm to a vector field constructed by the strengths and directions of the wind. In this case we set v_{field} in the reverse direction of the vector direction when trees extend from q_{final}. As shown in Figure 5, the branches are extended following the directions of the wind. The upstream criterion for the VF-RRT path is about 56%, 29% lower than that of the basic RRT and T-RRT path each.

B. Conservative Arbitrary Surface

We now design an arbitrary potential function as shown in Fig. 6, and apply the VF-RRT algorithm after setting F as the negative gradient direction of the potential function. That is, $F(q) = -\partial C(q)/\partial q$, where q is a configuration and $C(q)$ is a potential function. As shown in table V, the paths produced by the VF-RRT algorithm shows a similar performance compared with those of the T-RRT algorithm. The integral of cost is a

TABLE V. Arbitrary Potential Function

	Integral of Cost	Upstream	Iterations	Length	Time(s)
RRT	445.0	0.95	526.3	410.6	0.05
T-RRT	377.6	0.62	9221.0	492.5	2.08
VF-RRT	389.1	0.58	872.4	508.2	0.61

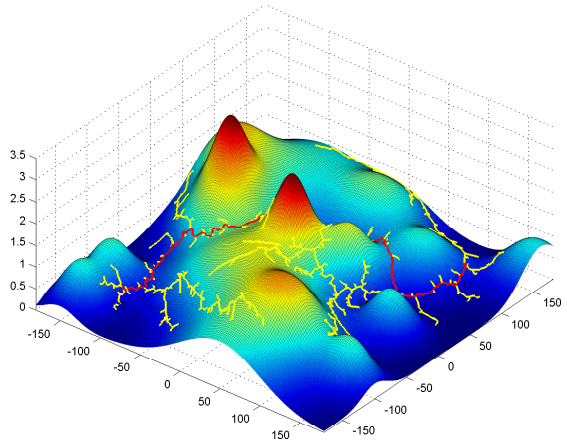

Fig. 6. The shape of VF-RRT Trees in a potential function. The VF-RRT trees are expressed as yellow trees, and the final path is shown as the red path.

little (about 3%) higher, but upstream is a little (about 7%) lower. However, the iteration number and time are decreased dramatically in comparison with the T-RRT algorithm.

V. Conclusion

We have proposed a new criterion, the upstream functional, to measure the path quality, and we have shown its suitability for near optimal random path planning. It can also be used to find paths for, e.g., underwater vehicles and flying vehicles because it is helpful to find reasonable paths in nonconservative vector fields. We also have presented a new randomized planning algorithm in vector fields, the VF-RRT algorithm. This algorithm is based on the Rapidly-exploring Random Tree (RRT) algorithm, with the distinguishing feature that trees tend to extend toward directions indicated by the vector field. By converting the cost distribution to a vector fields, the VF-RRT algorithm can produce good quality paths, that is, minimize the upstream functional.

References

[1] S. LaValle, *Rapidly-exploring random trees: a new tool for path planning*, TR 98-11, Computer Science Dept., Iowa State University, Oct. 1998.

[2] S. LaValle and J. Kuffner, *Rapidly exploring random trees: Progress and prospects*, In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, A K Peters, Wellesley, MA, 2001, pp. 293-308.

[3] C. Urmson and R. Simmons, *Approaches for heuristically biasing RRT growth*, *Proc. in IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2003, pp. 1178-1183.

[4] D.Ferguson and A. Stenz, *Anytime RRTS*, *Proc. in IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2006, pp. 5369-5375.

[5] L. Jaillet, J. Cortès, and T. Simèon, *Sampling-Based Path Planning on Configuration-Space Costmaps*, *IEEE Transactions on Robotics*, Aug. 2010, vol. 26, no. 4, pp. 635-646.

[6] D. Berenson, T. Simèon and S. S. Srinvasa, *Addressing Cost-Space Chasms in Manipulation Planning*, *IEEE Int. Conf. Robotics and Automation*, 2011, pp. 4561-4568.

[7] S. Karaman and E. Frazzoli, *Sampling-based algorithms for optimal motion planning*, *International Journal of Robotics Research*, vol. 21, no. 3, 2011, pp. 846-894.

[8] R. Alterovits, S. Patil and A. Dervakova, *Rapidly-exploring Roadmaps: Weighting Exploration vs. Refinement in Optimal Motion Planning*, *IEEE Int. Conf. Robotics and Automation*, 2011, pp. 3706-3712.

[9] A. Yershova and S. Lavalle, *Improving Motion-Planning Algorithms by Efficient Nearest-Neighbor Searching*, *IEEE Transactions on Robotics*, Feb. 2007, vol. 23, no. 1, pp. 151-157.