SAMPLING-BASED PLANNING, CONTROL, AND VERIFICATION OF HYBRID SYSTEMS [1]

Michael S. Branicky [*,2] **Michael M. Curtiss** [*]
Joshua Levine [*] **Stuart Morgan** [*]

[*] *Electrical Engineering and Computer Science, Case Western Reserve University*

Abstract: In this paper, we survey a planning, control, and verification approach in terms of sampling-based tools, such as Rapidly-exploring Random Trees (RRTs) and Probabilistic RoadMaps (PRMs). We review RRTs and PRMs for motion planning and show how to use them to solve standard nonlinear control problems. We extend them to the case of hybrid systems and describe our modifications to LaValle's Motion Strategy Library to allow for hybrid planning and verification. Finally, we extend them to purely discrete spaces (replacing distance metrics with cost-to-go heuristic estimates and substituting local planners for straight-line connectivity) and provide computational experiments comparing them to conventional methods, such as A*. We also review our work on the coverage, optimality properties, and computational complexity of sampling-based techniques. *Copyright © 2005 IFAC*

Keywords: Sampling-based planning, motion planning, hybrid systems, discrete spaces, complexity, verification, computational methods and tools

1. INTRODUCTION AND OVERVIEW

In this paper, we review a planning, control, and verification approach applicable to complex control systems, including those that are nonlinear, nonholonomic, hybrid, or purely discrete. In general, complete algorithms for planning, control, and verification of such systems are exponential in the state-space and control dimensions.

Attempts to fight this curse of dimensionality, have led to the introduction of randomized (or Monte Carlo or *sampling-based*) approaches that are capable of solving many challenging problems efficiently, at the expense of being able to guarantee that a solution will be found in finite time. See (LaValle and Branicky, 2002) for a review.

Two sampling-based methods that have achieved considerable success in the robotics motion planning literature are Rapidly-exploring Random Trees (RRTs) and Probabilistic Roadmaps (PRMs). The methods have been shown to solve challenging planning problems that involve high state-space dimension. (Citations given in Section 2.)

Over the past several years we have adapted these algorithms for use in solving nonlinear control problems and multi-agent coordination (Branicky and Curtiss, 2002; Curtiss, 2002); hybrid-systems planning, control, and verification (Branicky *et al.*, 2003; Levine, 2003); as well as planning in completely discrete spaces (Morgan and Branicky, 2004; Morgan, 2004). Finally, we have also been involved in de-randomizing these algorithms (Branicky *et al.*, 2001) and in proving their complexity (LaValle and Branicky, 2002).

[1] Partially supported by NSF grant CCR-0208919.
[2] Corresponding Author: `mb@ieee.org`

In this paper, we review this work and place the algorithms in a broader context. In particular, herein, *we advocate a sampling-based approach to all reachability-based problems, including planning, control, and verification*, by solving:

Sampling-Based Reachability Algorithm.

```
SampledReachSet:=InitialSet
loop:
  $x_r$ = random successor of SampledReachSet
  SampledReachSet = SampledReachSet $\cup$ $x_r$
```

Search can be terminated after a fixed number of iterations, if a goal point is reached, etc. Extra information can be stored using a tree/graph/forest structure, with edges labeled by the control action taking a parent to a successor. Search can also proceed backwards from a final set. Throughout the paper, we will examine algorithms which instantiate this general pattern.

2. SAMPLING-BASED BACKGROUND

RRTs are a probabilistic exploration method developed for searching the high-dimensional continuous spaces encountered in motion planning and control problems (LaValle, 1998; LaValle and Kuffner, 2000). The RRT algorithm begins with an initial configuration q_{start} as the tree. At each step, it selects a random configuration q_{rand} from the configuration space, then finds the nearest configuration already in the tree, q_{near}, using some definition of nearness (often Euclidean distance). From q_{near}, it moves some distance ϵ toward q_{rand}, and adds that new configuration to the tree. These steps are repeated until some q_{goal} (which may be a specific goal state or an element of a set of goal states) is reached, or until the tree has reached a certain size. RRTs have been shown to be probabilistically complete, and to have good space-filling properties in that their growth is biased toward the largest unexplored regions in the space (LaValle, 1998).

In addition to growing a tree from the starting state, many RRT implementations are "dual-tree," where one grows a second tree from the goal state and stops when the two merge.

PRMs constitute another sampling-based method for solving planning problems in high-dimensional spaces, especially multiple query path planning (Bekris *et al.*, 2003). The PRM algorithm operates by repeatedly selecting a random node $q_{\text{rand}}$and adding it to a set of nodes. The algorithm then attempts to connect $q_{\text{rand}}$to any other nodes in the set that are within some distance δ of q_{rand}.The connections are made by a local planner; in the simple case of holonomic planning,the local planner is often simply a straight-line generator with obstacle checking. This process is repeated until the PRM reaches a set size, or until most or all of the PRM is connected. Path planning queries are then solved by using a local planner to connect q_{start} and q_{goal} to the PRM, then finding a path between the connection points through the preplanned roadmap.

Like RRTs, PRMs are probabilistically complete using uniform sampling. Unlike RRTs, which expand out from one point, the nodes in a PRM follow the sampling distribution exactly, but do not create a connected graph until some threshold point density is reached.

3. NONLINEAR CONTROL VIA SAMPLING

Other researchers have applied RRTs to planning problems of various types including path-steering, manipulation planning for digital actors, varieties of holonomic planning, and kinodynamic planning (LaValle and Kuffner, 1999). To our knowledge, we are the first experimenters to test RRTs on standard control problems (Branicky and Curtiss, 2002; Curtiss, 2002). It is our hope that by studying the RRT's performance in these common problems, we will be able to gauge the strengths and weaknesses of RRT's compared to other approaches.

3.1 Pendulum Swing-Up

The first experiment we conducted was applying the RRT to the swing-up problem for a nonlinear pendulum (Branicky and Curtiss, 2002):

- Equation: $\ddot{\theta} = -3g/(2l)\sin\theta - 3\tau/(ml^2)$;
- Motor torques: $\tau \in \{-1, 0, 1\}$;
- Initial state of $\theta = 0$ (down) and $\dot{\theta} = 0$;
- Goal state of $\theta = \pi$ (up) and $\dot{\theta} = 0$.

The goal for the planner is to find a series of torque-time pairs that get the pendulum to the goal state. In all but the most trivial cases, the motor is unable to lift the pendulum to the goal state in one smooth motion. The pendulum therefore must be swung back and forth until it achieves sufficient velocity to reach the goal configuration. Our first try at solving the problem, a single-tree RRT using the straightforward Euclidean metric, $\rho = \sqrt{(\Delta\theta)^2 + (\Delta\dot{\theta})^2}$, proved to be quite successful. Usually finding a solution in less than 10,000 iterations (only a few seconds of computation on most modern computers), our implementation showed that the RRT algorithm is both fast and adaptable to many problem domains. See Figure 1 (left).

The dual-tree solution to the same problem was also impressive, sometimes finding a path to the goal state in close to half the time of its single-tree relative. One interesting characteristic of the solution trees is how clearly it demonstrates the dynamics of the system. See Figure 1 (right).

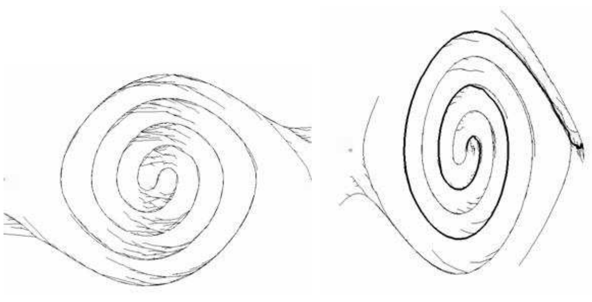


Fig. 1. Single- and Dual-RRT Solutions to the Pendulum Swing-Up Problem. The x-axis corresponds to θ and the y-axis to $\dot{\theta}$. The left image shows a single-tree RRT solution for the pendulum problem after 5600 iterations. The right image shows a dual-tree RRT search after 3300 iterations (solution in dark).

3.2 Acrobot

One can also apply this approach to more complex control problems, including the Acrobot (Sutton and Barto, 1998; Spong, 1994). See (Curtiss, 2002) for details.

3.3 Multi-Aircraft Planning

We also investigated prioritized RRT algorithms to plan for multiple aircraft in two-dimensions, traveling among six airports, with simple flight dynamics (Curtiss, 2002). We were able to generate plans for up to 800 holonomic agents in the air at one time. We also generated plans for tens of nonholonomic agents (with unicycle dynamics) in the air at one time.

4. HYBRID SAMPLING-BASED TOOLS

4.1 Hybrid Systems

Researchers in the computer science and control theory communities have produced many models for describing the dynamics of hybrid systems (Branicky, 1995; Branicky *et al.*, 1998). For the purpose of the discussion in this document, we consider a simple illustrative case, in which the constituent continuous state and input spaces (in each mode) are the same. Thus, we have a hybrid system of the form

$$\begin{aligned} \dot{x} &= f(x,u,q), & x &\notin J(x,u,q) \\ (x,q)^+ &= D(x,u,q), & x &\in J(x,u,q). \end{aligned} \tag{1}$$

Here, $x \in X$ is the continuous state, $u \in U$ is the input, and $q \in Q \simeq \{1, 2, \ldots, N\}$ is the discrete state or *mode.* Also, $f(\cdot,\cdot,q)$ is the continuous dynamics, $J(\cdot,\cdot,q)$ is the jump set, and $D(\cdot,\cdot,q)$ is the discrete transition map, all for mode q. The map D relates the post-jump hybrid state $(x,q)^+$ from the pre-jump hybrid state (x,q). The input u, which can include both continuous and discrete components, allows the introduction of non-determinism in the model, and can be used to represent the action of control algorithms and the effect of environmental disturbances. The evolution of the discrete state q models switches in the control laws and discrete events in the environment, such as failures.

Briefly, the dynamics are as follows: the system starts at hybrid state $(x(t_0), q_0)$ and evolves according to $f(\cdot,\cdot,q_0)$, until the set $J(\cdot,\cdot,q_0)$ is reached. At this time, say t_1, the continuous and/or discrete state instantaneously jump to the hybrid state $(x(t_1^+), q_1) = D(x(t_1), u(t_1), q_0)$, from which the evolution continues. While terse, the above model encompasses both autonomous and controlled switching and jumps, and allows modeling of a large class of embedded systems, including ground, air and space vehicles and robots; see (Branicky, 1995; Branicky *et al.*, 1998) for more details. Below, we describe an approach to hybrid planning, control, and verification based on RRTs.

4.2 Hybrid RRTs

A general, hybrid RRT (Branicky and Curtiss, 2002) can be achieved in various ways, depending on the underlying hybrid systems model and specifics of the continuous and discrete dynamics (and symmetries therein). We now wish to give a taste of the way a hybrid RRT might work for the model (1). A planning/control problem will have a *target set* $T \subset X \times Q$.

The simplest algorithm one might envision would explore reachable space by growing a *forest* of RRTs, one in each mode, with jump points among various trees in the forest identified. In the more general case, evolution will start from a set of seeds in a *start set* $S \subset X \times Q$, encompassing one or more modes, and proceed from there according the *hybrid-RRT algorithm* outlined below. One may think of the resulting tree as (a) growing in the hybrid state space, $X \times Q$, or (b) as growing in X, with nodes and arcs colored/labeled by the current mode.

Even under this setup, there are several cases to consider:

(1) General specifications; S, T, J, and D are arbitrary.

(2) Homogeneous specifications: $S = B \times Q$ and $T = G \times Q$. i.e., the start and target sets are independent of mode.
(3) Homogeneous switching: $J(x, q) \equiv J(x)$ and $D(x, q) \equiv D(x)$, independent of q.
(4) Unrestricted switching: $J(\cdot, q) = X$ for all q and $D(x, q) = x$ for all x, q.

While the above is not exhaustive, it provides a sense of a few types of symmetries in the discrete dynamics that can be exploited by the algorithm.

In the case of unrestricted switching, the **hybrid-RRT algorithm** is exactly the same as outlined above, except that the control set is augmented to allow mode changes: $U \mapsto U \times Q$. The other cases are non-trivial. In the case of homogeneous specifications, x_{rand} lives, and distances are measured in, the continuous state space X; in the general case, x_{rand} lives, and distances are measured in, the hybrid state space $X \times Q$. The latter brings up the issue of designing metrics for combined continuous and discrete space, which is a topic of current research. In either case, the NEW-STATE function must respect the hybrid dynamics. Typically, for purely continuous RRTs, the states examined come from extending the state x_{near} according to the dynamics $f(x, \cdot)$ for a fixed time and for various (sampled) $u \in U$. In the hybrid case, this continues to hold for (x_{near}, q_{near}) if there are no intersections with the jump set $J(\cdot, q_{near})$. If there are, evolution continues from the destination point(s), using the same or different u, until the desired amount of time elapses.

In Figure 2 we give an example of a hybrid RRT. Pictured from left to right in each row are four square floors, 1 through 4. Stairs (jumps) are given by triangles, with destinations given by inverted triangles in the next highest floor. The tree started in the gray square in the center of floor 1, and the target set is the gray square on floor 4. Successive rows represent different stages in the expansion process. The hybrid state is $s = (x, y, q) \in [-20, 20] \times [-20, 20] \times \{1, 2, 3, 4\}$. The metric used is $\rho(s_1, s_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} + 20|q_1 - q_2|$.

4.3 Computational Tool for Hybrid Systems

We have built a visual tool for manipulating and studying hybrid systems; Our tool builds on the Motion Strategy Library (MSL) developed by Steve LaValle *et al.* (2003).

Details of our software implementation appear in (Branicky *et al.*, 2003) and (Levine, 2003). The former reference also showed two- and three-dimensional stair climbers solved using the tool. Later, the tool was used to solve other examples, including stair-climbing with obstacles and

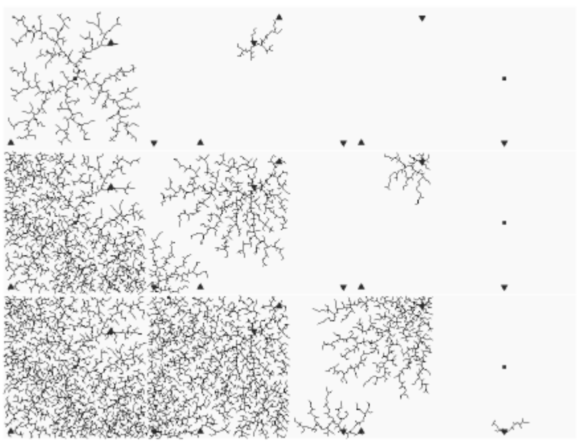
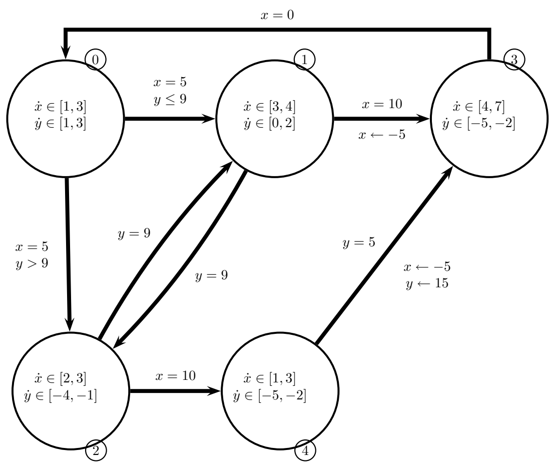


Fig. 2. Stair Climbing: an example hybrid RRT.

a bouncing ball—which incorporates impulsive jumps. Most significantly, we have augmented the tool to allow planning, verification, and testing for rectangular hybrid automata (RHA). To control state transitions, we made use of MSL's built-in collision detection algorithms: for each guarding condition, a polygonal region was constructed to represent the guard. The `RRT_Extend()` algorithm was modified so that before the new state is added, a check is done to see if the new state will collide with this region. If so, the tool performs a reset using the `EdgeReset()` function, and adds this reset state instead of the original new state calculated previously. One other feature necessary to implement here was the concept of a "multi-state view." To this end, the Render object was modified to be able to draw all states at the same time, on top of each other, using different colors depending on which mode it is in. See Figure 4, which shows trajectories for the RHA of Figure 3. There, the colors pink, brown, green, yellow , and cyan are used to represent discrete states 0–4, resp. The white lines represent state resets. Figure 4 also shows an example of using RRTs for the verification of hybrid systems, as explained in the caption.


$x = 0$
0
1
3
$\dot{x} \in [1,3]$ $\dot{y} \in [1,3]$
$x = 5$ $y \leq 9$
$\dot{x} \in [3,4]$ $\dot{y} \in [0,2]$
$x = 10$
$x \leftarrow -5$
$\dot{x} \in [4,7]$ $\dot{y} \in [-5,-2]$
$y = 9$
$x = 5$ $y > 9$
$y = 9$
$y = 5$
$x \leftarrow -5$ $y \leftarrow 15$
$\dot{x} \in [2,3]$ $\dot{y} \in [-4,-1]$
$x = 10$
$\dot{x} \in [1,3]$ $\dot{y} \in [-5,-2]$
2
4
Fig. 3. A Rectangular Hybrid Automata.

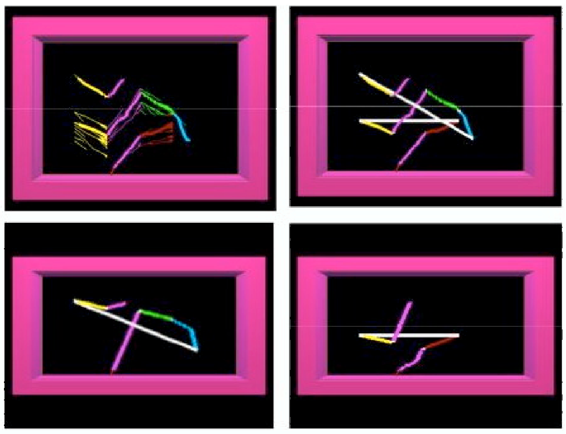


Fig. 4. Sampling-Based Verification. Four screen shots from our software tool. The upper left screen shows an RRT for the RHA grown using "multi-action" growth. The remaining three show hybrid traces from initial hybrid state $(x, y; q) = (1, 1; 0)$ to goal (2.5, 15; 0). The RHA was designed to accomplish this while cycling through discrete modes 0-1-3-2-2-4-3-0, as verified in the upper right diagram. Our software also found that the goal could be reached by cycling through 0-2-4-3-0 and 0-1-3-0 (bottom left and right, resp.)—specification violations.

5. DISCRETE SAMPLING-BASED TOOLS

In general a discrete planning problem exists in a discrete space consisting of a countable set of states $\mathbf{S}$, and a corresponding set of discrete dynamics that define, for each state q, a transition rule $\Delta : \mathbf{S} \rightarrow 2^{\mathbf{S}}$, where $\Delta(q) = \mathbf{Q}' \subseteq \mathbf{S}$ is the (finite) set of possible successors to q. The planning problem gives us a start state q_{start} and a set of goal states $\mathbf{G}$, and asks us to find a path from q_{start} to some $q_{\text{goal}} \in \mathbf{G}$. This solution path must consist of a sequence of states $q_{\text{start}} = q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow \cdots \rightarrow q_n = q_{\text{goal}}$ with each transition to a new state obeying the transition rule for the previous state (i.e., for each transition $q_i \rightarrow q_{i+1}$, it must be true that $q_{i+1} \in \Delta(q_i)$).

In discrete planning, it is often useful to consider the transition rules for states as corresponding to a small set of universal or near-universal operators, each of which map $q \rightarrow q'$ in a predictable way. For example, in planning an agent's motion in a grid world, where each state corresponds to an (x, y) pair, the operators are the four possible moves to adjacent squares: $o_{\text{up}}(x, y) \rightarrow (x, y + 1)$, $o_{\text{down}}(x, y) \rightarrow (x, y - 1)$, $o_{\text{left}}(x, y) \rightarrow (x - 1, y)$, and $o_{\text{right}}(x + 1, y) \rightarrow (x, y)$ (with the constraint that an operator cannot be applied to a state if its resultant state would collide with an obstacle).

Although a variety of informed search algorithms exist for such problems, they generally either become infeasible as the problem size grows (as in the case of A*) or have poor performance in spaces where "obstacles" are not predicted by the heuristic, whether the obstacles are explicit or due to behavior of the system which is not predicted by the simplified heuristic (as in the case of best-first search).

5.1 Discrete RRTs

We have also introduced a discretization of the RRT algorithm, which replaces the distance metric used for determining nearness with a heuristic estimate of the cost-to-go of the same type that is used in general informed search methods, e.g., A*. As in the original RRT, the discrete algorithm begins with an initial state q_{start}. At each step, we select a random state q_{rand} from the state space(making sure that the state selected is not already in the tree). We find the nearest state in the tree, q_{near}, based on a heuristic estimate of the cost-to-go from each state to q_{rand}. Considering each possible operator on q_{near}, we select the one which yields the successor state q_{new} that is closest to q_{rand} but is not already in the tree, and add q_{new} to the tree with an edge from q_{near} to it. If q_{near} has no successors which are not in the tree, no new node is added during this iteration.

A new algorithm that mitigates the issue of sub-optimal steps is the *Rapidly-Exploring Random Leafy Tree* (RRLT). The RRLT algorithm keeps an open list of all states reachable in one step from the current tree: the "leaves" of the tree nodes. When q_{rand} is selected, the nearest leaf is located directly and added to the tree, and all of its successors are added to the open list (except those that are already tree or leaf nodes).The RRLT algorithm prevents the possibility of a failed ExtendRRT step, since every leaf is a state which does not exist in the tree, and is therefore a valid candidate q_{new}.

5.2 Experiments and Results

We have tested our discrete sample-based algorithms on a number of problems, including benchmarks such as the 8-puzzle, 24-puzzle, and $k \times k$ Knight-Swapping puzzles (Branicky *et al.*, 2003). In those experiments, our algorithms performed well with respect to traditional algorithms such as A^*, particularly when there were obstacles present that were not captured by the heuristic. In later work, we used discrete RRTs to plan for multiple agents in discrete grid worlds with obstacles (Morgan and Branicky, 2004; Morgan, 2004). We were able to plan the paths of 300 airplanes, with several hundred in the air at any given time, in 2–3 seconds on a 2 GHz G5 (2GB RAM), using simple single-directional RRLT search.

5.3 Properties of Sampling-Based Algorithms in Discrete Space

In order to better understand the adaptation of sampling-based planning to discrete space, we have directly examined the Voronoi regions that determine their properties, as well as the resulting coverage and optimality effects (Morgan and Branicky, 2004; Morgan, 2004).

The fundamental difference between Voronoi regions in discrete and continuous spaces is the introduction of overlap in discrete space. Although the definition of a Voronoi region guarantees non-overlapping regions in continuous space, discrete space heuristics generally cannot guarantee that there will not be ties. It is of course possible to modify any heuristic to prevent ties by introducing arbitrary tie-breaking rules, but doing so will introduce an equally arbitrary bias in the exploration of the RRT algorithm, which is generally undesirable. Thus, discrete RRT algorithms should expect the occurrence of ties. See Figure 5.

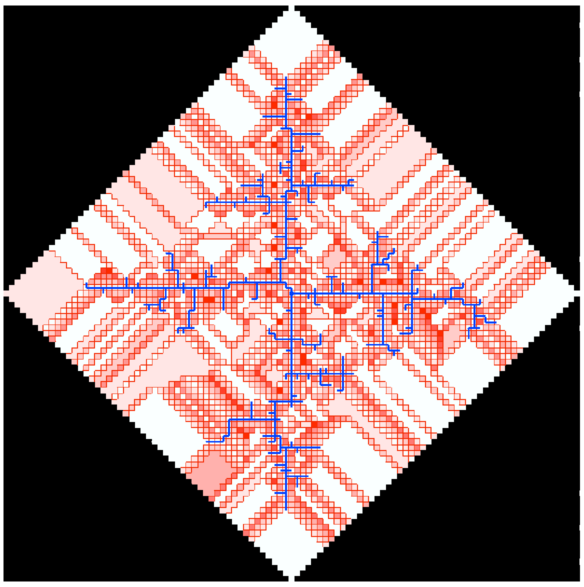


Fig. 5. Discrete Voronoi Regions.

ACKNOWLEDGMENT

This work was partially supported by the NSF Embedded and Hybrid Systems program, under the direction of Dr. Helen Gill (grant CCR-0208919); it does not necessarily reflect the views of the NSF.

REFERENCES

Bekris, K.E., B.Y. Chen, A.M. Ladd, E. Plaku, and L.E. Kavraki (2003). Multiple query probabilistic roadmap planning using single query planning primitives, *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Las Vegas, NV, pp. 656–661.

Branicky, M.S. (1995). *Studies in Hybrid Systems: Modeling, Analysis, and Control.* Sc.D. thesis, M.I.T., Cambridge, MA.

Branicky, M.S., V.S. Borkar, and S.K. Mitter (1998). A unified framework for hybrid control: Model and optimal control theory. *IEEE Trans. Automatic Control*, **43**(1):31–45.

Branicky, M.S., S.M. LaValle, K. Olson, and L. Yang (2001). Quasi-randomized path planning. *Proc. IEEE Intl. Conf. Robotics and Automation*, pp. 1481–1487, Seoul, KOREA.

Branicky, M.S. and M.M. Curtiss (2002). Nonlinear and Hybrid Control Via RRTs. *Proc. Intl. Symp. Mathematical Theory of Networks and Systems*, South Bend, IN, August 12–16.

Curtiss, M.M. (2002). *Motion Planning and Control using RRTs*, M.S. Thesis, Electrical Engineering and Computer Science Dept., Case Western Reserve U., Cleveland, OH. `http://dora.cwru.edu/msb/pubs/mmcMS.pdf`

Frazzoli, E., M.A. Dahleh, and E. Feron (1999). A hybrid control architecture for aggressive maneuvering of autonomous helicopters. In *IEEE Conf. Decision and Control.*

LaValle, S.M. (1998). Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State U.

LaValle, S.M. and M.S. Branicky. On the relationship between classical grid search and probabilistic roadmaps. *Proc. Workshop Algorithmic Foundations of Robotics*, December 2002.

LaValle, S.M. and J.J. Kuffner (1999). Randomized kinodynamic planning. *Proc. IEEE Int'l Conf. on Robotics and Automation*, pp. 473–479.

LaValle, S.M. and J.J. Kuffner (2000). Rapidly-exploring random trees: Progress and prospects. *Proc. Workshop Algorithmic Foundations of Robotics.*

LaValle, S.M. *et al.* (2003). Motion Strategy Library. `http://msl.cs.uiuc.edu/msl/`

Levine, J.A. (2003). *Sampling-Based Planning for Hybrid Systems*, M.S. Thesis, Dept. of Electrical Engineering and Computer Science, Case Western Reserve U., Cleveland, OH. `http://dora.cwru.edu/msb/pubs/jalMS.pdf`

Morgan, S.B. (2004). *Sampling-Based Planning for Discrete Spaces*, M.S. Thesis, Dept. of Electrical Engineering and Computer Science, Case Western Reserve U., Cleveland, OH. `http://dora.cwru.edu/msb/pubs/sbmMS.pdf`

Spong, M.W. (1994). Swing up control of the acrobot. *Proc. IEEE Intl. Conf. Robotics and Automation*, San Diego, CA, pp. 2356–2361.

Sutton, R.S. and A.G. Barto (1998). *Reinforcement Learning.* MIT Press, Cambridge, MA.