Robotics: Science and Systems 2005
Cambridge, MA, USA, June 8-11, 2005

An RRT-Based Algorithm for Testing and Validating Multi-Robot Controllers

Jongwoo Kim
GRASP Laboratory
University of Pennsylvania
Philadelphia, PA 19104
Email: jwk@grasp.cis.upenn.edu

Joel M. Esposito
Weapons and Systems Engineering
US Naval Academy, MD 21402
Email: esposito@usna.edu

Vijay Kumar
GRASP Laboratory
University of Pennsylvania
Philadelphia, PA 19104
Email: kumar@grasp.cis.upenn.edu

***Abstract*— We address the problem of testing complex reactive control systems and validating the effectiveness of multi-agent controllers. Testing and validation involve searching for conditions that lead to system failure by exploring all adversarial inputs and disturbances for errant trajectories. This problem of *testing* is related to *motion planning*, with one main difference. Unlike motion planning problems, systems are typically not controllable with respect to disturbances or adversarial inputs and therefore, the reachable set of states is a small subset of the entire state space. In both cases however, there is a goal or *specification set* consisting of a set of points in state space that is of interest, either for demonstrating failure or for validation.**

In this paper we consider the application of the Rapidly-exploring Random Tree algorithm to the testing and validation problem. Because of the differences between testing and motion planning, we propose three modifications to the original RRT algorithm. First, we introduce a new distance function which incorporates information about the system's dynamics to select nodes for extension. Second, we introduce a weighting to penalize nodes which are repeatedly selected but fail to extend. Third, we propose a scheme for adaptively modifying the sampling probability distribution based on tree growth. We demonstrate the application of the algorithm via three simple and one large scale example and provide computational statistics. Our algorithms are applicable beyond the testing problem to motion planning for systems that are not small time locally controllable.

I. Introduction

As the use of logic-based or reactive control laws grows in both robotics and other fields, so does the need for automated design and analysis tools. The focus to date in the automated safety verification literature has been on the solution of the reachability problem, initially through symbolic methods (e.g., [11], [18]) and later through numerical techniques (e.g., [6], [17]). However, the class of systems for which the reachability problem is tractable is quite limited in both expressiveness and dimensionality. An alternative approach to exhaustively proving safety is to simply search for a single counter example – a series of inputs, disturbances or parameters that causes a system to fail. We term this semi-decision approach the *Testing Problem*.

Inspired by the connections between the Testing Problem for complex control systems and the Motion Planning problem, we have recently applied the Rapidly-exploring Random Tree (RRT) algorithm [13] to the testing problem [1], [7] with considerable success. RRT algorithm is an incremental searching algorithm which explores state space fast and uniformly. However, the two problems are different. Perhaps the most significant difference between the two problems lies in the nature of the system dynamics in each case. Robotic systems are almost always controllable (by design), so the reachable space is often the entire free space. With the exception of any workspace obstacles, whose configurations are known in advance, the tree can be expected to extend to fill the entire state space. On the other hand, when we test complex control systems, it is frequently with respect to disturbances or adversarial inputs. These systems are frequently *not* controllable with respect to disturbances or adversarial inputs — in fact, the reachable set is usually a tiny fraction of the entire state space. In such systems, the traditional uniform sampling distribution, combined with the inherent Voronoi bias of the RRT algorithm, leads to a slow reduction (improvement) in dispersion (coverage). The issue is not easily remedied because, unlike C-space obstacles, the reachable set is not *a priori* known.

Accordingly, we propose three modifications to the original RRT algorithm. First, we develop a new distance function which encodes local information about the system's dynamic constraints with a first order approximation. Second, because the reachable state space is generally a small fraction of the total state space, we introduce a weighting factor which penalizes the repeated extension of boundary nodes. Finally, we propose a scheme for adaptively modifying the sampling probability distribution between the traditional uniform distribution and heavily biased toward the specification set based on tree growth.

The paper is organized as follows. In Section II-A we formally define the testing problem. Section II-B reviews the original RRT algorithm and reviews the most relevant literature. Section III examines three key features of the traditional RRT algorithm which are troublesome for testing problems; proposes methods to remedy them and presents simple illustrative examples, complete with comparative computational statistics. A new algorithm unifying the enhancements is presented in Section IV. The algorithm is used to solve a multi-agent pursuit-evasion problem and performance statistics are discussed. Concluding remarks follow in Section V.

II. Background and Related Work

A. Problem Statement

Definition 2.1: We define a **Finite Time Control System** as a tuple $C = (X, U, T, Init, f)$ where

- $X \subset \mathbb{R}^n$ is a set of *free* state variables;
- $U \subset \mathbb{R}^m$ is a compact set of input values;
- $T = [t_0, t_f] \subset \mathbb{R}$ is a compact time interval the system evolves over;
- $Init \subset X$ is a compact set of possible initial conditions;
- $f : X \times U \to \mathbb{R}^n$ is a vector field which prescribes the time derivative of the state variables.

We are generally interested in systems with collections of rigid bodies with very complicated dynamics, especially high-dimensional continuous systems or hybrid (discrete/continuous) and switched systems where f may be a non-smooth function of x. We do not impose any structure on the nature of the dynamics (except assuming that solutions exist in the sense of Fillipov). Note that in the case of rigid body systems X is essentially $\mathcal{C}_{free} \times T\mathcal{C}_{free}$. We use the term "input" in the most general sense in that it can include yet unspecified feedback control inputs, human in the loop inputs, and disturbances.

Problem 2.2: **Testing Problem**: Given a tuple (C, x^0, S), where

- $C = (X, U, T, Init, f)$ is a finite time control system,
- $x^0 \in Init$, and
- S is a specification set,

the goal is to determine an open loop control law $\mathcal{U} : T \to U$ such that $\exists t \in T$ for which $x(t) \in S$.

In other words, the goal is to determine a counter-example – an input sequence which will cause the system to fail by entering S – if one exists. However, in order to make the problem algorithmically tractable, instead of searching the set of all possible functions $\mathcal{U} : T \to U$, the search must be restricted to some subset of functions with finite dimensional parametrization.

For the sake of convenience we make three additional assumptions. First, assume $X \subset \mathbb{R}^n$ is defined in such a way that a point in $\mathbb{R}^n$ can be easily tested for membership in X. Second, assume the specification set S can be defined as the sub-level set of some function $S = \{x | x \in X, s(x) \leq 0\}$. Finally, we restrict our search over $\mathcal{U}$ to piecewise constant functions of time with k segments, each of time duration δt. Thus, instead of the continuous map $\mathcal{U}$, we consider the search over $\bar{\mathcal{U}} : T \to U$, as the search for a k-vector of parameters. With $u^i \in U$

$$\bar{u} = [u^1, u^2, \ldots, u^k]^T$$

so the input $u(t)$ is given by

$$u(t) = u^i \in U \;\; if \;\; t_o + (i-1)\delta t \leq t < t_o + (i)\delta t$$

for $i = 1, \ldots, k$.

B. Related Work

We base our approach on the Rapidly-exploring Random Tree (RRT) algorithm [13]. A very basic algorithm is given in Algorithms 1 and 2, where ρ is some suitable metric and pdf is a probability distribution. RRTs are attractive because they work directly in the space of admissible inputs making them suitable for systems with dynamic constraints and because they are *probabilistically complete* [13]. While much work on safety verification exists, the approach of using RRTs to analyze hybrid systems is recent. In [8] RRTs were used to design trajectories of hybrid systems. The first published work using RRTs for analyzing hybrid systems is [3], [15]. In a similar vein, a blimp system control law was validated under unpredictable but bounded disturbances [10]. In [2], the reachable set for aircraft collision avoidance problem was obtained and several extensions of the RRT approach were mentioned. We have applied a variant of this method [1] to testing hypotheses and establishing properties of biological networks.

We review two developments from [7], used later in this paper: coverage estimation and the RRFT algorithm. First, the coverage of the state space with tree nodes is important both because it can be used as a termination criteria in the event a solution is not found and because it provides a methodology for comparing the effectiveness of two algorithms that is not dependent on the goal position. Typically *Dispersion* is used [12] which is loosely defined as the radius of the largest ball in X which does not contain a tree node. We reject it on the grounds that it is difficult to compute and because, by only focusing on the largest such ball, it yields an overly conservative estimate of coverage. We introduce a coverage measure which can be thought of as a discretized average dispersion. Given an RRT $(\mathcal{T})$ and a set of grid points $G \subset X$ with spacing δx

$$c(\mathcal{T}, G) = 1 - \frac{1}{|G| \cdot \delta x} \sum_{x^g \in G} \min(\rho(x^g, \mathcal{T}), \delta x). \qquad \text{(II.1)}$$

Second, the Rapidly-exploring Random Forest of Trees (RRFT) algorithm searches over time invariant parameters and initial conditions by planting many RRTs at a sampling of parameter values. Individual trees are grown and terminated by monitoring c. Both c and the RRFT method are used in Sect. IV.

Algorithm 1 Generate RRT: $\mathcal{T}$

```
Initialize RRT: T.addVertex(x^0)
while ∄x ∈ T such that s(x) ≤ 0 do
    Extend(T)
end while
```

There have been several enhancements to the basic RRT algorithm. In [5] a method for penalizing the repeated selection of collision prone nodes for extension is introduced. In [16] a node selection strategy is described which increases the natural Voronoi bias of the method for the purposes of dispersion

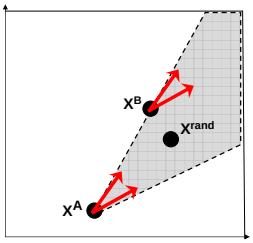

Fig. 1. The reachable space is shown as the shaded gray region. The arrows indicate the possible velocity vectors at each node. Nodes selected for extension on the basis of their distance from x^{rand} (x^B) may be difficult to connect from when the system is not small time locally controllable. x^A is a better candidate for extension.

Algorithm 2 Extend($\mathcal{T}$)

$x^{rand} \in X \leftarrow$ pdf()
$x^{near} \leftarrow \arg\min_{x^j \in \mathcal{T}} \rho(x^j, x^{rand})$
$u^{new} = \arg\min_{u \in \bar{U}} \{\rho(x^{rand}, x^{near} + \int^{\delta t} f(x,u)dt)\}$
$x^{new} = x^{near} + \int^{\delta t} f(x, u^{new}(t))dt$
$\mathcal{T}$.addVertex(x^{new})
$\mathcal{T}$.addEdge(u^{new}, $x^{near} \rightarrow x^{new}$)

reduction. However, neither approach is able to reduce the dispersion (which must be measured within the reachable set) for uncontrollable systems. Biasing the sampling toward regions close to the goal state has been tried in [14], [15] and [3] with some success. However the sample bias factor is fixed *a priori* and it can lead to difficulties in non-convex systems because of the presence of local minima. In [10], a metric accounting for under-actuated dynamics is suggested but is specific to the aerial robots example considered there.

III. Enhancements to the RRT Algorithm

In this section, we propose three modifications to the original RRT algorithm, all designed to deal with systems that may only traverse a small fraction of the entire state space and in which there are no obvious metrics to establish proximity relationships. Recall that the Voronoi bias coupled with the use of a uniform distribution decreases the dispersion of the tree nodes in X. However, for uncontrollable systems it may be impossible to reduce the dispersion of the tree nodes in X below a critical value, which is an unknown constant. Instead the goal is to simply find a solution quickly while reducing the dispersion of the tree nodes *within the reachable space*, $\mathcal{R}$, by using heuristics to account for the system's motion constraints.

A. Dynamics-based selection of proximal node

Example 3.1: Consider the trivial example

$$\dot{x}_1 = 2, \quad \dot{x}_2 = u, \tag{III.1}$$

where $u \in U = [1, 2]$. The reachable space, which is normally unknown can easily be computed by hand in this case, and is shown as the shaded region in Fig. 1. A state x^{rand} is generated and the planner must select the "closest" tree node, x^{near} to attempt to connect from. Line 2 of Algorithm 2 (traditional RRT) selects $x^{near} \leftarrow x^B$ for extension based on proximity to $x^{rand} \in X$, as determined by a distance metric ρ that is implicitly assumed to be a Euclidean metric.

However, none of the possible velocity vectors at that state (indicated as region between the thick arrows) are able to proceed in the required direction. Despite the fact that $\rho(x^A, x^{rand}) > \rho(x^B, x^{rand})$, x^A is actually more suited to extension because the possible velocity vectors include a direction that moves toward x^{rand}. In addition to testing problems, this situation arises in a variety of robotic applications where the system is nonholonomic (*e.g.*, wheeled carts), and particularly in systems with constraints on forward velocities (*e.g.*, unmanned aerial vehicles). Ideally both distance and velocity constraints should be used to estimate a "time to connection".

To remedy the situation in Fig. 1 we propose replacing $\rho(x^j, x^{rand})$ in Line 2 of Algorithm 2 with a local first order approximation of the time-to-go.

$$t_{2go}(x^j, x^{rand}) = \begin{cases} \rho(x^j, x^{rand})/g & if \;\; g > 0 \\ \infty & if \;\; g \leq 0 \end{cases} \tag{III.2}$$

where g represents the instantaneous speed with which x^{rand} can be approached

$$g = \max_{u \in \bar{U}} \left[-\frac{\partial \rho(x, x^{rand})}{\partial x} f(x,u)|_{x=x^j} \right]$$

Intuitively t_{2go} computes the distance from x^j to x^{rand} and divides by a first order approximation of the speed with which the distance can be decreased, giving t_{2go} units of time. Note that a negative value of g implies that the distance is actually increasing, which can be interpreted as infinite "time-to-go" (to first order). In a given iteration if none of the existing nodes have a finite value for t_{2go}, one can be chosen at random or based on some secondary criteria (such as distance as determined by ρ).

From a computational point of view the maximization may be done by exhaustive search or by exploiting some problem dependent feature. For example if $f(x,u)$ is an affine function of u and the set U is the Cartesian product of rectangles, the maximization is a linear program in n dimensions which can be solved efficiently. If no efficient methods exist to compute this quantity, evaluating every node via this method can be intensive. In such a case, t_{2go} can be used as a secondary criteria to select x^{near} among the, for example, 10 closest nodes according to the Euclidean metric.

We next consider an example that is from the verification community. Although it is not central to robotics, it has many of the properties that are central to multi-agent robotic systems.

Example 3.2: The hybrid automata model of a thermostat has been a popular example in the verification literature [9]. Fig. 2 shows the system model. $x = (x_1, x_2, x_3) \in X \subset \mathbb{R}^3$ where x_1 is the temperature in the room, x_2 is the elapsed time, and x_3 is a timer that measures the cumulative amount of time the heater has been on for. The dynamics have two modes which denote the heater being "on" or "off". U consists of $u_{on} = [2, 4]$; and $u_{off} = [-3, -1]$. The values u_{on} and

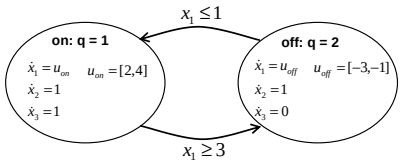

Fig. 2. The system dynamics of the thermostat.

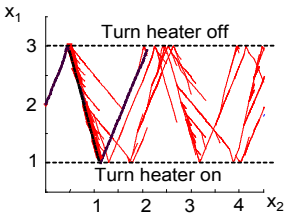

Fig. 3. The solution of the thermostat counter example via the RRT using the dynamics-based selection of proximal nodes (Temperature vs. time).

Metric	No. of Nodes	Computation Time (sec)
Euclidean	2284	376.4
t_{2go}	1627	231

TABLE I

THERMOSTAT EXAMPLE: A COMPARISON OF THE USE OF THE EUCLIDEAN METRIC AND t_{2go} INTRODUCED IN SECT. III-A, AVERAGED OVER 10 TRIALS ON A 1GHZ PC.

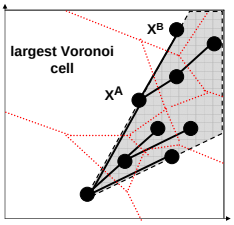

Fig. 4. The reachable space is shown as the shaded gray region, bold circles and lines are the RRT, and dotted lines are the Voronoi cells. Nodes on the boundary of the reachable space have disproportionately large Voronoi regions, causing them to repeatedly be selected as x^{near}.

u_{off} represent the possible heating and cooling rates in the two modes. The conditions $x_1 \le 1$ and $x_1 \ge 3$ enable the mode switches $off \rightarrow on$ and $on \rightarrow off$ respectively. In [9] a symbolic verification tool is used to answer the question: "After an initialization period of two minutes, is it possible for the heater to be on for more than two thirds of the total time at any point during the first hour of operation?" Such a question may be important from an energy consumption point of view. Therefore the specification set is

$$S = \{x \in X | 2/3x_2 - x_3 \le 0 \wedge -x_2 + 2 \le 0\}.$$

The initial conditions were mode = "on", and $x^o = [2\ 0\ 0]^T$. Aside from being a classical verification example, the senario is interesting in its own right. First, the system has quite nontrivial dynamics, since the control inputs do not appear in the right hand side of two of the state equations, or the specification equations. This, together with the narrow range of U, makes the reachable set, $\mathcal{R}$ a small subset of X. The set of possible velocity vectors at every point is very limited making this an ideal example to demonstrate the Dynamics-based selection of proximal node.

First the problem was solved 10 times selecting proximal nodes based on the Euclidean metric ρ; then 10 times with the Dynamics-based selection function t_{2go}. In all cases, the algorithm successfully computed a counter example as seen in Fig. 3. Table I shows the computational statistics for two algorithms.

B. History-based selection of proximal node

A second situation is shown in Fig. 4 where the traditional RRT is applied to the system and, after 8 iterations, the resulting tree is shown using dark circles and line segments. Because the reachable set is so small, nodes on the boundary will tend to have disproportionately large Voronoi regions, such as x^A in Fig. 4. When a uniform distribution is used to generate x^{rand}, most samples will fall outside the reachable set and these boundary nodes will be selected for extension repeatedly. Each time, the same extremal inputs will be used to connect x^A to x^{rand} in vain, instead resulting in x^B. Boundary nodes which are repeatedly selected but fail to extend should be penalized to counter balance this Voronoi bias so that they are less likely to be selected in the future.

If a node is selected for extention as x^{near} in Line 2 of Algorithm 2 and the minimization in Line 3 produces an input u^{new} which has been applied previously, the resulting x^{new} is already an element of $\mathcal{T}$. When this happens we say the node has "failed to extend"; and determine the next best u^{new} which extends the tree (suggested in [5]).

For each $x^j \in \mathcal{T}$ we propose storing the number of times the node has failed to extend n^j. This value can be used to compute a penalty weight to discourage the repeated selection of boundary nodes which fail to extend. Let $n_{\min}$ and $n_{\max}$ be the least and greatest values of n^j in the tree at a given iteration. The *History-based weighting* is defined as

$$H(x^j, x^{rand}; \rho) = \frac{\rho(x^j, x^{rand}) - \rho_{\min}}{\rho_{\max} - \rho_{\min}} + \frac{n^j - n_{\min}}{n_{\max} - n_{\min}} \quad \text{(III.3)}$$

where $\rho_{\min} = \min_{x^i \in \mathcal{T}} \rho(x^i, x^{rand})$ and $\rho_{\max}$ is defined in a similar manner. These bounds are used to normalize the distances so that the impact of the second term is not problem dependent. Note that any distance function, including t_{2go} can be substituted for ρ.

Example 3.3: The RRT algorithm is used to find trajectories of the linear dynamic system with bounded control inputs in the form of

$$\dot{x} = Ax + Bu + b \quad \text{(III.4)}$$

where $x \in X = [-200\ 200] \times [-200\ 200]$ and $u \in U = [-10\ 10] \times [-10\ 10]$.

Fig. 5 shows trajectories generated by the RRT algorithm using the Euclidean metric (*left*) and using the History-based weighting described above (*right*). Note that reachable set

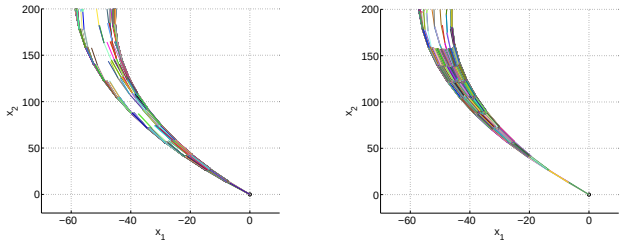

Fig. 5. RRT for a linear system using the Euclidean metric (*left*) vs. a History-based selection of proximal node (*right*). After 5000 nodes the coverage of the reachable space is much more dense when using the weighting.

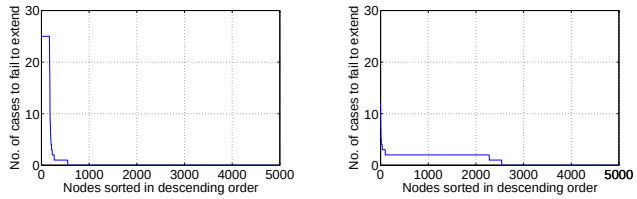

Fig. 6. Value of n^j for each node (sorted in descending order) using the unweighted Euclidean metric (*left*) and History-based weighting (*right*).

is small fraction of the environment. The interior of the reachable region with the History-based selection of proximal node method is much more densely covered than Euclidean metric. Fig. 6 shows n^j for each node in $\mathcal{T}$. Nodes are sorted in descending order to facilitate the visualization. In the conventional RRT algorithm, a smaller portion of nodes (on the boundary of the reachable set) have disproportionately high values of n^j.

C. Adaptively biased sample generation

Intuitively, biasing the sampling distribution for x^{rand} to generate a disproportionate number of samples inside the set S is effective when the system is easily steered toward S (*i.e.* the system is output controllable with respect to $s(x)$). In general, biasing the sample distribution toward the goal can make sense but it is difficult to decide *a priori* which problems will benefit.

We update the amount of biasing for every N_s iterations of the RRT algorithm, where N_s is user defined number. If in a given iteration $\rho(x^{near}, x^{rand}) > \rho(x^{new}, x^{rand})$, where ρ is a metric function, we call such an iteration *successful* because the tree has grown toward x^{rand}. We count the number of successful iterations n_s, out of the n_β iterations where random states are generated inside the set defined by $s(x) \leq 0$ and compute

$$\beta = \frac{n_s}{n_\beta} \tag{III.5}$$

If x^{near} is not successful in growing toward x^{rand} inside the specification set or the best candidate for x^{new} from x^{near} is already in a tree in the above test, we eliminate the x^{near} from consideration as x^{near} for the testing in future iterations to prevent it from being chosen repeatedly. Values of β close to unity indicate biasing sample generation inside S has been beneficial.

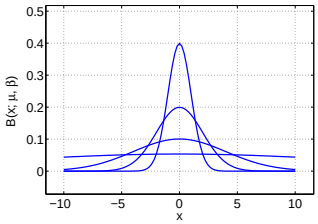

Fig. 7. The distribution $B(x;\mu,\beta)$ with $\mu = 0$ and various values of β.

Our proposed probability density function $B(x;\mu,\beta)$, to be used in Line 1 of Algorithm 2, resembles a Gaussian over some compact set, $a \leq x \leq b$

$$B(x;\mu,\beta) = \begin{cases} N(x;\mu,\sigma(\beta))+ \\ C_t/(b-a), & a \leq x \leq b \\ 0 & else \end{cases} \tag{III.6}$$

where $N(x;\mu,\sigma(\beta))$ is the Gaussian distribution with mean μ and standard deviation $\sigma(\beta)$. The last term, $C_t/(b-a)$, is added to ensure that the area under the curve is equal to one. C_t represents the area of the truncated portions above $x = b$ and below $x = a$

$$C_t = \int_{-\infty}^{a} N(x;\mu,\sigma)dx + \int_{b}^{\infty} N(x;\mu,\sigma)dx.$$

Obviously μ should be selected so that $s(\mu) \leq 0$. The standard deviation of $N(x;\mu,\sigma(\beta))$ effectively determines the bias and should be computed using $\beta \in [0,1]$

$$\sigma(\beta) = (1-\beta)(\sigma_{\max} - \sigma_{\min}) + \sigma_{\min}, \tag{III.7}$$

where $\sigma_{\max}$ and $\sigma_{\min}$ are user-defined values of the maximum and minimum standard deviations.

Fig. 7 illustrates the shape of $B(x;\mu,\beta)$ with different values of β. Distribution (III.6) can be easily implemented using any random normal generator and rejecting points generated outside the compact domain.

Example 3.4: We consider a hovercraft in constant altitude flight with 6 states, $x = (x_1, x_2, \theta, v_1, v_2, \omega)$. The dynamic equations are

$$\begin{aligned} m\dot{v}_1 &= (f_1+f_2)\cos(\theta) + f_{x_1 air}(x, v_{air}(x)) \\ m\dot{v}_2 &= (f_1+f_2)\sin(\theta) + f_{x_2 air}(x, v_{air}(x)) \\ J\dot{\omega} &= (f_2-f_1)l + \tau_{air}(x, v_{air}(x)) \end{aligned}$$

The control inputs are $u = [f_1 \;\; f_2]^T$ (forward actuating forces) and $U = [-10,10] \times [-10,10]$. Forces due to wind disturbances in the x_1, x_2 and θ directions are $f_{x_1 air}$, $f_{x_2 air}$, and τ_{air} whose exact expressions are omitted for brevity but are quadratic in the difference between the craft's velocity and the wind velocity v_{air} and vary with the orientation of the craft. Note that the state is partitioned into two regions (indoor and outdoor) which define the wind velocity differently:

$$v_{air} = \begin{cases} [-\alpha_v x_2 \;\; \beta_v x_1]^T, & \sqrt{(x_1)^2+(x_2)^2} \leq 100 \\ [0 \;\; 0]^T, & \sqrt{(x_1)^2+(x_2)^2} > 100 \end{cases}.$$

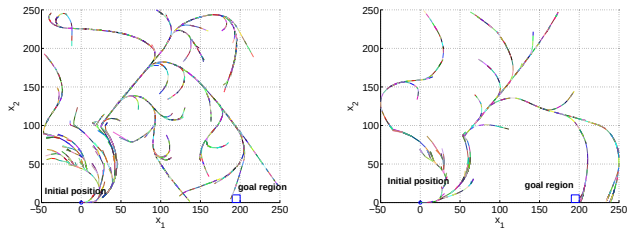

Fig. 8. RRTs of the hovercraft problem with uniform sampling (*left*) and with adaptive bias (*right*).

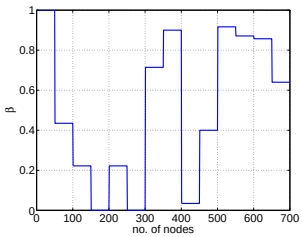

Fig. 9. The evolution of the biasing factor β for the hovercraft problem.

We would like to determine if a hovercraft under these wind conditions can reach some goal zone, $S = \{(x_1, x_2) \in [190, 200] \times [0, 10]\}$. Note that when outdoors the wind forces are significantly greater in magnitude than the control inputs, making the system uncontrollable.

The initial state is $x^0 = [0\ 0\ 0\ 0\ 0\ 0]^T$. The distribution (III.6) was used to solve the problem 10 times on a 1GHz PC. Fig. 8 shows the solutions of the problem with the uniform sampling distribution and adaptive bias. Fig. 9 shows how β changes as the algorithm evolves. The adaptive algorithm is able to exploit the situations in which biasing is effective. As shown in Table II, the adaptive biasing algorithm improves the efficiency of RRT method compared to other fixed bias strategies rather dramatically.

Sampling Method	No. of Nodes	Computation Time (sec)
Uniform	3556	1753.5
Med. Bias	1017	490.2
Heavy Bias	912	408.3
Adaptive Bias	678	342.5

TABLE II

HOVERCRAFT EXAMPLE: A COMPARISON OF THE SAMPLING STRATEGY INTRODUCED HERE (ADAPTIVE BIAS) TO FIXED-BIAS SAMPLING STRATEGIES, AVERAGED OVER 10 TRIALS ON A 1GHZ PC.

IV. UNIFIED ALGORITHM

A. Unified algorithm

Algorithm 3 and 4 present the unification of the enhancements presented in the previous section. Note that, since most robotic problems are controllable, the Algorithm 1 can terminate when a solution is found. In our case, it is a distinct possibility that no solution exists so we impose a secondary termination criteria. The change in coverage over the trailing N iterations Δc , measures the growth of the tree. If Δc drops below some user-defined Δc_{min} we terminate the search.

Algorithm 3 Generate enhanced-RRT: $\mathcal{T}$

Initialize RRT: $\mathcal{T}$.addVertex($x^0 \leftarrow x^{init}, n^0 \leftarrow 0$)
Global: $\beta = 1$
while ($\nexists x \in \mathcal{T}$ such that $s(x) \leq 0$) AND $\Delta c \geq \Delta c_{\min}$ **do**
  enhanced-Extend($\mathcal{T}$)
end while

Algorithm 4 enhanced-Extend($\mathcal{T}$)

$\sigma = (1 - \beta)(\sigma_{\max} - \sigma_{\min}) + \sigma_{\min}$
$x^{rand} \in X \leftarrow B(x; \mu, \beta)$ (see eq.(III.6))
$x^{near} \leftarrow \arg\min_{x^j \in \mathcal{T}}[H(x^j, x^{rand}; t_{2go})]$ (see eq.(III.2),(III.3))
$u^{new} = \arg\min_{u \in \bar{U}}[t_{2go}(x^{rand}, x^{near} + \int^{\delta t} f(x, u)dt)]$
$x^{new} = x^{near} + \int^{\delta t} f(x, u^{new}(t))dt$
if $x^{new} = x^j \in \mathcal{T}$ **then**
  $n^j + +$
  $\bar{U} \leftarrow \bar{U} - u^{new}$
  goto compute u^{new}
end if
$\mathcal{T}$.addVertex(x^{new}, $n^{new} = 0$)
$\mathcal{T}$.addEdge(u^{new}, $x^{near} \rightarrow x^{new}$)
reset $\bar{U}$
if N_s iterations **then**
  $\beta = \frac{n_s}{n_\beta}$
end if

B. A Multiagent Problem

We consider a problem where multiple autonomous vehicles must guard against an intruder entering a designated area. This scenario has applications in games such as "capture the flag" and can be viewed as a variant of the art-gallery problem. It has applications in homeland security where autonomous vehicles (boats, airplanes, ground robots) can be deployed to detect unidentified vehicles entering a cordoned-off area or an exclusion zone.

In this example, we examine a circular area, S_E guarded by 4 robots. Each robot has sensor foot prints which are assumed to be circular with radius R_d for detection and R_c for capture, as shown in Fig. 10. The guarding scheme is shown in Fig. 11. Initially, the guard robots distribute evenly along the perimeter of the exclusion zone. If the intruder enters the detection range of a guard robot, the robot pursues the intruder and other robots redistribute evenly along the circle C_E. If the intruder escapes the detection range of the pursuing robot, the robot returns to the perimeter and all robots redistribute evenly. The question we wish to answer is as follows. If an intruder or an adversary is allowed to start anywhere in a specified region S_I, and the guard robots are evenly distributed on the circle C_E, can the intruder enter the exclusion zone (S_E) uncaptured? The answer to our question can only be found by searching for an initial condition and a control input function for the intruder which drives it into the exclusion zone without crossing any of the capture ranges. We assume each of the intruder and guard robots has 5 states, $x^i = (x_1^i, x_2^i, \theta^i, v^i, \omega^i)$ and 2 control

inputs, $u^i = (u_1^i, u_2^i)$ where x^1 and u^1 indicate states and input of the intruder. The dynamics with nonholonomic constraints are given by:

$$\begin{aligned} &\dot{x}_1^i = v^i cos(\theta^i), \;\; \dot{x}_2^i = v^i sin(\theta^i), \;\; \dot{\theta^i} = \omega^i \\ &\dot{v^i} = u_1^i, \;\; \dot{\omega^i} = u_2^i. \end{aligned} \tag{IV.1}$$

We can define the free space $X = X_1 \times X_2 \times \cdots \times X_5 \setminus \bigcup_{i=2}^{5} B(x^i(t), R_c) \subset \mathbb{R}^{25}$ where

$$X_i = \{(x_1^i, x_2^i, \theta^i, v^i, \omega^i) \in \mathbb{R}^5 | (x_1^i)^2 + (x_2^i)^2 \leq R_I^2\}$$
$$B(x^i(t), R_c) = \{(x_1^i, x_2^i) | (x_1^1 - x_1^i)^2 + (x_2^1 - x_2^i)^2 \leq R_c^2\}.$$

Then the specification set S is defined by

$$S = \{x \in X | (x_1^1)^2 + (x_2^1)^2 < R_s^2\}$$

where R_s is the radius of the circle C_E.

To evenly distribute guard robots along the perimeter, we use the algorithm proposed in [4]. Each guard robot is subject to the force

$$\tau^j = -k \nabla \psi^2(q^j) - C\dot{q}^j + \sum_{k \in N_j} F_r(q^j, q^k) \tag{IV.2}$$

where $q^j = (x_1^j, x_2^j) \in \mathbb{R}^2$ is the position of robot j, $\psi : \mathbb{R}^2 \rightarrow \mathbb{R}$ is an implicit function description of the perimeter of the exclusion zone that must be guarded and N_j is the set of robots neighboring robot j. F_r is a Coloumb-like repulsive force that ensures that the robots do not cluster together, while C is a constant which provides a viscous damping term. The force is applied to a point that is at a finite distance away from a robot to address nonholonomic constraints. A detailed description of the control law including a proof of convergence to different shapes is provided in [4]. However, the analysis in the paper cannot be used to predict the transients as each guard robot moves toward the perimeter.

Note that the reachable set of states in X is a small subset of the entire state due to the fact that the system is uncontrollable and U is bounded. Finally, note that the intruder can start anywhere in the set S_I. In other words, the initial condition for the intruder must be chosen from this finite set, each condition leading to a RRT.

We apply the RRFT algorithm with enhancements suggested in Sec. IV-A to the problem. The control inputs are $u = (u_1^1, u_2^1) \in U = [-6\ 6] \times [-\pi/12\ \pi/12]$ with $R_I = 300m$, $R_s = 100m$, $R_d = 100m$ and $R_c = 40m$. Fig. 12 shows the forest of trees where a solution trajectory is found, visualizing the position of the intruder. Eight initial conditions are generated and a forest starts to grow until a solution is found. Due to the space limitation, we show only the trajectories obtained for the algorithm with the "dynamics-based selection of proximal node". However, the Table III shows the statistics obtained for this example with all the options. The second column shows the average number of nodes used to find a solution trajectory for the intruder robot (one such trajectory is shown in Fig. 12). The third column shows the computation time with different options. The first main point to note from these two columns is that the standard algorithm takes four times as long requiring

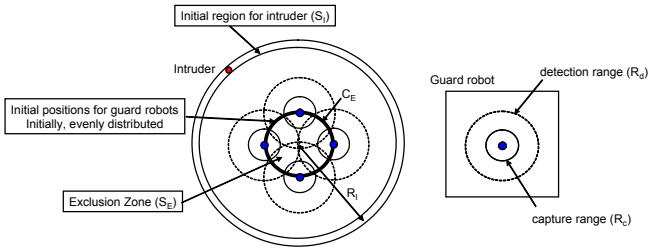

Fig. 10. Initial conditions for guard robots and intruder. Each robot has a detection range R_d within which the intruder is detected, and capture ragne R_c within which the intruder is captured.

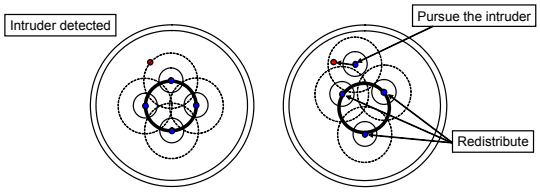

Fig. 11. Guarding scheme of the robots. Distribute until the intruder is detected (*left*) ; and pursue if the intruder is within the detection range of a guard robot (*right*).

three times the number of nodes to find the same solution. The second main point in this example is that adaptive biasing allows the most improvement in efficiency. The fourth column shows a snapshot of the coverage measure after 5000 nodes have been visited for all cases. The N/A is used because it is less meaningful to show this number for adaptive sampling when improving coverage is not the driving force behind using the adaptive bias.

Enhancement Method	No. of Nodes	Computation Time (sec)	Coverage Measure
No Enhancement	20544	9020.9	0.0190
Dynamics-based	12960	3655.6	0.0196
History-based	8176	3101.8	0.0246
Adaptively biased	6398	1791.6	N/A
Three Enhancements	7520	2429.2	N/A

TABLE III

GUARD-INTRUDER EXAMPLE: A COMPARISON OF THE ALGORITHM, WITH AND WITHOUT ENHANCEMENTS AVERAGED OVER 10 TRIALS ON A 3GHZ PC.

V. CONCLUSION

The RRT algorithm has been successful in solving complex motion planning problems. We explore the application of this algorithm and its variants to the problem of testing complex reactive control systems and validating the effectiveness of multi-agent controllers. Testing and validation involve searching for conditions that lead to system failure by exploring all adversarial inputs and disturbances for errant trajectories. Unlike motion planning problems, the systems may not be controllable with respect to disturbances or adversarial inputs and the reachable set of states is generally a small subset

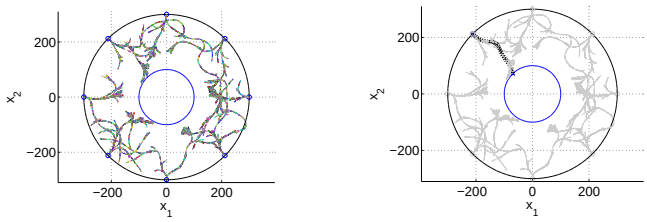

Fig. 12. The forest of RRTs with 8 different initial conditions. A solution trajectory of the intruder is highlighted on the right.

of the entire state space. Because of the differences between testing and motion planning, we propose three modifications to the original RRT algorithm. First, we develop a new distance function which encodes local information about the system's dynamics with a first order approximation. Second, because the reachable state space is generally a small fraction of the total state space, we modify the node selection strategy to discourage the repeated selection of boundary nodes. Finally, we propose a scheme for adaptively modifying the sampling probability distribution based on tree growth to the specification set. We demonstrate the application of the algorithm via three simple examples and one large scale (25 dimensions) multi-agent pursuit-evasion and provide computational statistics demonstrating a reduction of computation time by a factor of three.

ACKNOWLEDGEMENTS

This work was supported by NSF grant IIS-0413138, ONR grant FA8650-04-C-7133 and NSF grant CNS-0410514.

REFERENCES

[1] Calin Belta, Joel M. Esposito, Jongwoo Kim, and Vijay Kumar. Computational techniques for analysis of genetic network dynamics. *International Journal of Robotics Research*, 24(2-3):219–235, 2005.

[2] Amit Bhatia and Emilio Frazzoli. Incremental search methods for reachability analysis of continuous and hybrid systems. In *HSCC*, volume 2993 of *Lecture Notes in Computer Science*, pages 142–156. Springer, 2004.

[3] Michael S. Branicky, Michael M. Curtiss, Joshua Levine, and Stuart Morgan. RRTs for nonlinear, discrete, and hybrid planning and control. In *IEEE Conference on Decision and Control*, December 2003.

[4] Luiz Chaimowicz, Nathan Michael, and Vijay Kumar. Controlling swarms of robots using interpolated implicit functions. In *IEEE International Conference on Robotics and Automation*, 2005.

[5] P. Cheng and S. M. LaValle. Reducing metric sensitivity in randomized trajectory design. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 43–48, 2001.

[6] Alongkrit Chutinam and Bruce Krogh. Computational techniques for hybrid system verification. *IEEE transactions on automatic control*, 48(1):64–75, 2003.

[7] Joel M. Esposito, Jongwoo Kim, and Vijay Kumar. Adaptive RRTs for validating hybrid robotic control systems. In *International Workshop on the Algorithmic Foundations of Robotics*, Zeist, Netherlands, 2004.

[8] E. Frazzoli, M.A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. In *AIAA Confernce on guidance, navigation and control*, August 2000.

[9] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HYTECH: A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1(1–2):110–122, 1997.

[10] Jongwoo Kim, Jim Keller, and Vijay Kumar. Design and verification of controllers for airships. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2003.

[11] Gerardo Lafferriere, George J. Pappas, and Sergio Yovine. Symbolic reachability computation for families of linear vector fields. *Journal of Symbolic Computation*, 32:231–253, 2001.

[12] Steven M. LaValle and Michael S. Branicky. On the relationship between classical grid search and probabilistic roadmaps. In *International Workshop on the Algorithmic Foundations of Robotics*. December 2002.

[13] Steven M. LaValle and James J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, 2001.

[14] Steven M. LaValle and James J. Kuffner. Rapidly exploring random trees: Progress and prospects. In B. Donald, K. Lynch, and D.Rus, editors, *Algorithmic and computational robotics: new directions*, pages 293–308. A.K. Peters, Wellesley, MA, 2001.

[15] Joshua A. Levine. Sampling-based planning for hybrid systems. Master's thesis, Case Western Reserve University, September 2003.

[16] S. R. Lindermann and S. M. LaValle. Incrementally reducing dispersion by increasing Voronoi bias in RRTs. *IEEE Internation Conference on Robotics and Automation*, 2004.

[17] Ian Mitchell and Claire Tomlin. Level set methods for computation in hybrid systems. In B. Krogh and N. Lynch, editors, *Hybrid Systems : Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 310–323. Springer Verlag, 2000.

[18] R.Alur, T.A.Henzinger, and P.-H Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22:181–201, 1996.