

# Adaptive RRTs for Validating Hybrid Robotic Control Systems

Joel M. Esposito<sup>1</sup>, Jongwoo Kim<sup>2</sup>, and Vijay Kumar<sup>2</sup>

<sup>1</sup> WSE Department, US Naval Academy, Annapolis, MD 21403, USA

<sup>2</sup> MEAM Department, University of Pennsylvania, Philadelphia, PA 19104  
esposito@usna.edu, jwk,kumar@grasp.cis.upenn.edu

**Abstract.** Most robot control and planning algorithms are complex, involving a combination of reactive controllers, behavior-based controllers, and deliberative controllers. The switching between different behaviors or controllers makes such systems hybrid, *i.e.* combining discrete and continuous dynamics. While proofs of convergence, robustness and stability are often available for simple controllers under a carefully crafted set of operating conditions, there is no systematic approach to experimenting with, testing, and validating the performance of complex hybrid control systems. In this paper we address the problem of generating sets of conditions (inputs, disturbances, and parameters) that might be used to "test" a given hybrid system. We use the method of Rapidly exploring Random Trees (RRTs) to obtain test inputs. We extend the traditional RRT, which only searches over continuous inputs, to a new algorithm, called the Rapidly exploring Random Forest of Trees (RRFT), which can also search over time invariant parameters by growing a set of trees for each parameter value choice. We introduce new measures for coverage and tree growth that allows us to dynamically allocate our resources among the set of trees and to plant new trees when the growth rate of existing ones slows to an unacceptable level. We demonstrate the application of RRFT to testing and validation of aerial robotic control systems.

## 1 Introduction

Hybrid systems provide mathematical models of discrete/continuous dynamic systems. Many robotic systems including walking robots, grasping and manipulation, or logic-based software controlled robots can be modelled under this framework. In fact most robot control and planning algorithms are complex, involving a combination of reactive controllers [2], behavior-based controllers [23], and deliberative controllers [11,16]. While it is possible to analyze each controller in isolation it is well known that the interaction between discrete and continuous time dynamics of such systems can produce rich and often unexpected behavior. For this reason, as these systems grow in complexity and sophistication, the need for automated design tools increases. The focus to date in the literature has been on the formal verification of safe operation, via the solution of the reachability problem, initially through symbolic methods [27,17] and later through numerical techniques [3,1,5,24,9]. However, it soon became apparent that the class of hybrid control systems for which the reachability problem was decidable is quite limited in both expressiveness and dimensionality.

*Test generation* – a well established concept for software design – is a relatively new approach to analyzing control systems. Rather than prove controller safety exhaustively, our approach is to try to generate a set of test scenarios, using Rapidly Exploring Random Trees (RRT), that cause the system to fail. In addition to considering traditional continuous inputs, we have extended the method to consider uncertain time invariant parameters. We call our algorithm the rapidly exploring Forest of Trees (RRFT). The merit of this approach as compared with reachability analysis is that decidability issues do not come into play because we are not attempting to represent or manipulate a reachable set. The drawback of the approach is that it is a semi-decision method, meaning that we can only *disprove* system safety by counter-example – safety cannot be proved. Despite this drawback we feel that randomized approaches hold the most promise for addressing complex nonlinear real-world problems for which trial and error testing is not sufficient; and formal analysis is intractable. Similar work has recently appeared which uses genetic algorithms [26]. Some works have used RRTs as a synthesis tool for hybrid control systems [10,4]; but the idea of using RRTs to explore a system’s faults has only appeared in passing [15,8].

Our approach consists of drawing a parallel to, and using methods from, motion planning. Informally the *motion planning problem* is: given a robot with dynamics and constraints (obstacles), to find a path (if one exists) from the starting configuration to the goal configuration of robot in some complex high dimensional configuration space. Similarly, the goal of test generation is to find a sequence of inputs (or disturbances) and parameters (if one exists) which will take a hybrid system from an initial state to some unsafe set in the hybrid state space. Interestingly motion planning research experienced a similar evolution from exact (symbolic) methods [28], followed by the result that the problem was fundamentally hard [6], to a shift toward approximate methods that worked well in practice [25]. Most recently research activity focuses on *randomized* approaches to the problem which have been shown to scale well with dimension.

The primary differences between motion planning and testing lie in the types of systems considered. For example, motion planning approaches do not traditionally consider hybrid systems (though recent work has [10]). Another difference is that in motion planning problems the state space is not simple connected, in the geometric sense, due to the presence of obstacles, necessitating the use of sophisticated collision detection algorithms. For hybrid system the state space is usually simply connected with a given mode. Perhaps most importantly, robotic systems are almost always output controllable (by design), so the reachable space is the entire output space. Therefore a solution usually exists, unlike testing problems. As a result considerations of when to stop growing the tree are rarely discussed.

The contributions and outline of this paper are as follows

- Formally introduce the Test Generation problem for complex control systems, point out the similarities to motion planning (Sections 2-3).
- Define new coverage criteria for RRTs (Section 4).

- Introduce the RRFT algorithm which is capable of searching both over traditional continuous inputs (like the RRT) *and uncertain time invariant parameters*. RRFT modifies its search strategy based on the run time coverage estimates (Section 5).

Incidentally, the algorithm can be used for motion planning or testing. Finally we apply the algorithms to validate two controllers for robotic unmanned aerial vehicles in 6.

## 2 Problem Statement

**Definition 1.** We define a **Finite Time Hybrid Control System** with unknown parameters dependencies as (modified from the Hybrid Automata, see [22]) as a tuple  $H = (X, Q, U, T, P, Init, f, Inv, E, G, R)$  where

- $X \subset \mathbb{R}^N$  is a set of continuous variables;
- $Q \subset \mathbb{N}$  is a set of discrete variables which index the system modes;
- $U \subset \mathbb{R}^m$  is a compact set of continuous input values;
- $T = [t_0, t_f] \subset \mathbb{R}$  is a compact time interval the system evolves over;
- $P \subset \mathbb{R}^p$  is a compact set of uncertain, time invariant parameters;
- $Init \subset Q \times X$  is a set of possible initial conditions;
- $f : Q \times X \times U \times P \rightarrow \mathbb{R}^N$  is a vector field which prescribes the time derivative of the continuous variables (*i.e.*,  $\dot{x} = f(q, x, u; p)$ );
- $Inv : Q \rightarrow 2^X$  assigns to each  $q \in Q$  an invariant set;
- $E \subset Q \times Q$  is a collection of edges describing the possible discrete transition (a.k.a.- mode switches);
- $G : E \rightarrow 2^{X \times P}$  assigns to each  $e = (q, q') \in E$  a guard; and
- $R : E \times X \rightarrow 2^X$  assigns to each  $e = (q, q') \in E$  a reset relation.

Throughout this paper we refer to  $(x, q)$  as the state of the hybrid system. Note that we use the term “input signal” in the most general sense in that it can include yet unspecified feedback control inputs, human in the loop type inputs, disturbances, etc. Note that the uncertain parameters can affect the continuous or discrete dynamics. Again, many robotic system can be modelled in this way (see Sect. 1). Examples of  $P$  could include a control gain, the initial condition of an adversarial agent, or the width of a narrow passage.

**Definition 2.** The **Testing Problem**  $TP$  is specified as a tuple  $(H, x^0, q^0, s, \bar{U}, \delta t)$  where

- $H$  is a finite time hybrid control system as described above;
- $x^0, q^0 \in Init$ ;
- $\bar{U}$  is user defined discretization of  $U$ ;
- $\delta t$  is the fixed time period for which a constant  $u \in \bar{U}$  is applied such that  $(t_f - t_o)/\delta t = k$  is an integer;
- $s : X \times Q \times P \rightarrow \mathbb{R}$  is a specification;

Given an initial state and a particular control function  $u(t)$ ,  $s$  can be used to define a set. If  $s(x, q; p) > 0$ , then  $(x, q)$  is an acceptable state inside the specification set for  $p$ ; otherwise it is unacceptable.

*Problem 1.* Given an initial state,  $(x^0, q^0)$ , the **Testing Problem** is to determine  $\{u_1, u_2, \dots, u_k\} \in \bar{U}$ , which define a piecewise constant control sequence

$$u(t) = u_i \text{ if } (i-1) \cdot \delta t \leq t < (i) \cdot \delta t \quad (1)$$

for  $i = 1, \dots, k$ , and  $p \in P$ , if they exist, such that  $\exists t \in T$  for which  $s(x(t), q(t); p) \leq 0$ .

In words, the goal of the test generator is to determine a counter-example – an piecewise constant input sequence and a value of the time invariant uncertain parameters which will cause the system to fail – if one exists. Note the similarity to trajectory planning (for example see problem statement in [7])

### 3 Testing Through Rapidly Exploring Random Trees

The similarities between the Testing Problem and the motion planning problem, suggest the use of a randomized methods such as Probabilistic Road Maps [13] or Rapidly Exploring Random Trees (RRTs) [19]. We choose the RRT primarily because it works directly with the set of admissible inputs and is therefore directly applicable to systems with complex dynamics. This algorithm has experienced widespread success in solving a variety of high dimensional and nonlinear problems in motion planning and has recently been applied to controller synthesis problems for hybrid systems [10,15,4]. Figure 1 illustrates the concepts and a very basic algorithm is given in Algorithms 1 and 2. Note that  $\rho$  is a suitable metric function; and the notation  $(x, q) + \int^{\delta t} H(u)dt$  means: using  $x, q$  as an initial condition, simulate the evolution of the hybrid system for  $\delta t$  seconds using  $u(t)$  as the control input. Various versions of the algorithm can be generated using different metrics, or random distributions. In Sect. 5 we focus on stopping criteria if no solution is found.

---

#### Algorithm 1 Grow Test Set T

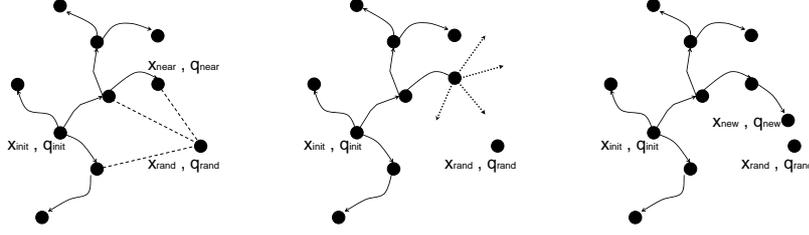
---

```
Initialize RRT:  $\mathcal{T}.\text{addvertex}(x^0, q^0)$ 
while  $\nexists (x, q) \in \mathcal{T}$  such that  $s(x, q) \leq 0$  do
  Extend( $\mathcal{T}$ )
end while
```

---

### 4 Coverage Measures

It has been shown that, for a controllable system, the RRT will ultimately cover the entire state space as the number of sample points goes to infinity [20]. Unfortunately,



**Fig. 1.** The Testing algorithm (inspired by the RRT [19]). The test set is represented as a tree  $\mathcal{T}$  with nodes as states  $(x, q)$  and edges as inputs  $u \in \bar{U}$ . First a new state is generated at random,  $x_{rand}, q_{rand}$ . The algorithm then determines the closest state,  $x_{near}, q_{near}$  in the tree to the random state (*left*). It determines which  $u \in \bar{U}$  brings  $x_{near}, q_{near}$  closest to  $x_{rand}, q_{rand}$  (*center*).  $u_{new}$  is applied for a duration  $\delta t$  and the new node  $x_{new}, q_{new}$  and edge  $u_{new}$  are added to the tree (*right*).

---

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


---

```

 $x_{rand}, q_{rand} \leftarrow \text{random}()$ 
 $x_{near}, q_{near} \leftarrow \text{nearestNeighbor}(\mathcal{T}, (x_{rand}, q_{rand}))$ 
 $u_{new} = \arg \min_{u \in \bar{U}} \{ \rho((x_{rand}, q_{rand}), (x_{near}, q_{near})) + \int^{\delta t} H(u) dt \}$ 
 $(x_{new}, q_{new}) = (x_{near}, q_{near}) + \int^{\delta t} H(u_{new}(t)) dt$ 
 $\mathcal{T}.\text{add vertex}(x_{new}, q_{new})$ 
 $\mathcal{T}.\text{addEdge}(u_{new}, (x_{near}, q_{near}) \rightarrow (x_{new}, q_{new}))$ 

```

---

because many of the systems we consider are not controllable with respect to the test inputs, the reachable set is not the entire space. It is very difficult in practice to estimate coverage quality because the reachable set is not known a priori. It is therefore important for us to estimate coverage for two reasons.

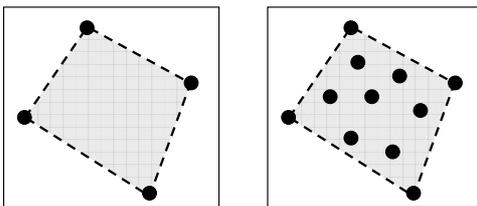
- Many testing problems may have no solution, meaning that there is not a counterexample to be found. In such a case we must decide when to terminate Algorithm 1.
- It is possible for Algorithm 1 to get stuck in “local minima” due to its greedy strategy [8] or to slow down because a tree is fully grown.

Regarding the second point, we can use coverage measures to determine when it might be appropriate to alter the search strategy. Indeed we explore this further in Section 5

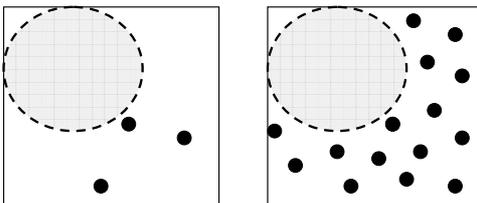
#### 4.1 Coverage: previous work

It has been pointed out many times [18] that coverage of  $X$  by  $\mathcal{T}$  is related to the *Voronoi Diagram* of the vertices of the tree. While this connection is useful for theoretical analysis the major problem is that it is impractical to compute Voronoi diagrams in dimensions over 2. The *Discrepancy* (a concept from the Monte Carlo literature) is also mentioned in [18] but it too is very difficult to compute. Another

appealing idea to measure the growth or coverage of the RRT is to compute the volume of the convex hull. Unfortunately the convex hull is more indicative of the distribution of the points than it is of the coverage. For example, in Figure 2 the left and right panels represent two sample sets whose convex hulls are identical. Obviously the sample shown to the right covers the state space better. In [21] a variant of the convex hull is explored. Rather than compute the hull of all tree vertices, vertices are grouped according to their depth from parent nodes. The union of these hulls clearly provides a better approximation however the selection of the grouping is somewhat arbitrary. It is not clear how to relate the union to coverage due to possible overlaps.



**Fig. 2.** Two sample sets which have the same convex hull. The set on the left clearly has inferior coverage to the set on right.

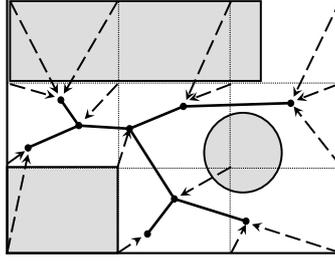


**Fig. 3.** Two sample sets which have the same dispersion (the size of the largest empty ball, drawn with dashed line). The set on the left clearly has inferior coverage to the set on right.

It appears that the most accepted measure to date is the *Dispersion* (see [12] or more recently [18]). Assuming we have a sample set  $\bar{X}$ , which contains  $N$  points, over the space  $X$ , it is defined as

$$\mu(\bar{X}, \rho) = \sup_{x \in X} \min_{\bar{x} \in \bar{X}} \rho(x, \bar{x}) \quad (2)$$

and can be thought of as the radius of the largest empty ball in  $X$  and obviously depends greatly on the choice of metric  $\rho$ . While its use has been advocated for analyzing planners we reject it for computation on two grounds: (1) it is impractical to compute in high dimensions; and, (2) it is an overly conservative coverage measure because it only considers the *largest* ball. For example, in Figure 3 the left and right



**Fig. 4.** A grid is super imposed on the state space. The shaded regions indicate unreachable sets. The distances from the grid points to the closest nodes are  $d_j$  (shown as dashed arrows) and the grid spacing is  $\delta$ .

panels represent two sample sets with the same dispersion. Obviously the sample shown on the right covers the state better.

## 4.2 New Coverage Measures

We have three goals: to measure the coverage of the state space  $X$  by the tree  $\mathcal{T}$ ; to measure the growth of the tree  $\mathcal{T}$  and to measure the coverage of the set of time invariant parameters  $P$  by a countable finite set of values  $\bar{P}$ .

*RRT Coverage* We begin by overlaying a grid of  $n_g$  points and spacing  $\delta$  on the state space. We calculate the minimum distance from each grid point  $j$  to the set of nodes in the tree,  $d_j$ . The quantity  $\min(d_j, \delta)$  may be thought of as the radius of the largest ball centered at each grid point which does not contain a tree node or adjacent grid points (see Figure 4). Given a tree  $\mathcal{T}$  we define its coverage,  $c(\mathcal{T})$ , as

$$c(\mathcal{T}) = \frac{1}{\delta} \sum_{j=1}^{n_g} \frac{\min(d_j, \delta)}{n_g} \quad (3)$$

which is the average of all the node distances, normalized by the grid spacing.

Our measure is similar to an approximation of an “average” dispersion, but far less conservative and faster to compute. Clearly this measure is a monotonically decreasing function. If it goes to zero on a given grid it tells us that any set whose distance along its smallest dimension is greater than the grid spacing has been entered. Said another way, the state space is covered up to a resolution equal to the grid spacing. Overall one of the advantages of this measure is that, the grid size can be as fine or coarse as one chooses. Finer grids will require more distance queries but are more accurate indications of coverage. Of course grids can be generated in the “output” or specification space to measure coverage there as well. From a computational point of view it should be stressed that this list of distances can be updated incrementally as new tree nodes are added, since the affect of each new node is local.

*RRT Growth* The derivative of  $c(\mathcal{T})$  with respect to the number of vertices in the tree,  $n_v$ , indicates the growth. Therefore

$$g(\mathcal{T}) = -dc(\mathcal{T})/dn_v \quad (4)$$

In practice the derivative is actually a finite difference and we may choose to look at the change in  $c$  over the course of adding several new vertices to  $\mathcal{T}$ .

*Time invariant parameter set coverage* The set of parameters over which we may tests the system may be generated directly since they are not subject to differential constraints and therefore the sample set can be engineered to have a certain coverage of  $P$ . Halton sequences [12] have naturally low dispersions and are cheap to compute. Therefore we define the coverage of  $P$  using the dispersion defined in eq.(2) normalized by  $\mu_{max}$ .

## 5 Forest of Random Trees Algorithm

The original RRT given in Alg. 1 only addresses time varying inputs such as  $u(t)$ . Recall that the evolution of our system is characterized by time invariant parameters,  $p \in P$  as well. In our RRFT algorithm (see Alg. 3), the repeated application of the RRT algorithm results in a tree for every choice  $p \in P$  (called the seed value). Accordingly, we need to consider a *set of trees* (a forest) that rapidly explore the state space. We call a RRT grown (or rooted) at  $p_i \in \bar{P}$ ,  $\mathcal{T}_{p_i}$ . Initially we plant trees at  $\bar{P} = \{p_1, \dots, p_{n_t}\}$ , where  $n_t$  is the maximum number of active trees we can consider concurrently. At any point if a counter example is found (a state and parameter  $s(x, q; p_i) \leq 0$  for which  $(x, q) \in \mathcal{T}_{p_i}$ ) the algorithm terminates.

Because we have limited computational resources, we must decide how to allocate them in growing the trees – choosing which to grow and which to terminate. As the RRT algorithm progresses, we monitor the progress of each tree. If at any point the growth of one of the trees as measured by  $g(\mathcal{T}_{p_i})$  drops below a threshold  $\bar{g}$ , the tree is considered no longer actively growing; or, if the coverage  $c(\mathcal{T}_{p_i})$  is less than a threshold  $\bar{c}$ , the tree is considered fully grown. In either case the tree is terminated. Provided the set  $P$  is not adequately covered with seeds (as measured by the dispersion) a new “seed” is planted and a new tree is initiated. The process of planting and growing new trees continues until a counter example is discovered, or until  $P$  is sufficiently covered ( $\mu(\bar{P}) \leq \bar{\mu}$ ) with seed values, whose trees have stopped growing.

One key component of this approach is that each RRT can be computed in parallel on a different CPU’s, therefore we assume a fixed computational resource that will dictate the number of trees that can be simultaneously computed in parallel.

## 6 Examples

We demonstrate the algorithm on two examples involving Robotic Unmanned Aerial Vehicles.

---

**Algorithm 3** Test Generation:  $\mathcal{T}_p = \text{RRFT}(H, x^0, q^0, s, \bar{U}, \bar{g}, \bar{c}, \bar{\mu}, n_t)$ 


---

Generate initial seed set  $\bar{P} = \{p_1, \dots, p_{n_t}\}$  where  $p_i \in P$

**for**  $i = 1, \dots, n_t$  **do**

Initialize RRT:  $\mathcal{T}_{p_i}.\text{addvertex}(x^0, q^0)$

**end for**

**while**  $n_t \neq 0$  **do**

**for**  $i = 1, \dots, n_t$  **do**

Extend( $\mathcal{T}_{p_i}$ )

**if**  $\exists(x, q) \in \mathcal{T}_{p_i}$  such that  $s(x, q; p) \leq 0$  **then**

return  $\mathcal{T}_{p_i}$

break (test case found)

**else**

**if**  $g(\mathcal{T}_{p_i}) \leq \bar{g}$ , OR,  $c(\mathcal{T}_{p_i}) \leq \bar{c}$  **then**

terminate  $\mathcal{T}_{p_i}$

$n_t \leftarrow n_t - 1$

**if**  $\mu(\bar{P}) > \bar{\mu}$  **then**

$n_t \leftarrow n_t + 1$

Generate new  $p_i \in P$  via Halton sequence and append to  $\bar{P}$

Initialize RRT:  $\mathcal{T}_{p_i}.\text{addvertex}(x^0, q^0)$

**end if**

**end if**

**end for**

**end while**

---

### 6.1 Example 1: Aircraft conflict resolution

As a first example we test an aircraft collision avoidance protocol proposed in [24]. We test over a continuous input ( $u(t)$ , a wind disturbance) and a constant parameter ( $p$ , the minimum separation distance) and consider a scenario involving 5 aircraft. The problem has 15 states, which is considerably larger than problems which have been considered in the literature on reachability and verification.

Each aircraft has three states,  $X_i = (x, y, \theta)$  and there are 5 aircraft so the continuous state space is  $X = X_1 \times \dots \times X_5$ . The continuous dynamics  $f : Q \times X \times U \times P$  are

$$\dot{x}_i = v \cos(\theta_i) + (-d_1 \sin(\theta_i) + d_2 \cos(\theta_i))(-\sin(\theta_i)) \quad (5)$$

$$\dot{y}_i = v \sin(\theta_i) + (-d_1 \sin(\theta_i) + d_2 \cos(\theta_i))(\cos(\theta_i)) \quad (6)$$

$$\dot{\theta}_i = \omega(q; p) \quad (7)$$

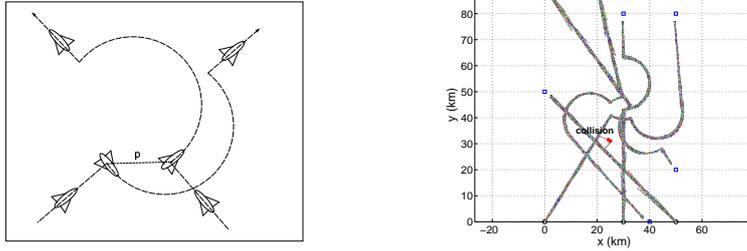
Where  $v$  is a constant forward velocity;  $u = [d_1 \ d_2]^T \in [-w, w] \times [-w, w]$  is a wind disturbance whose normal component to the planes alters their dynamics (this is the main difference versus [24]). Note that  $q$  and  $p$  do not explicitly appear in the dynamics but rather determine  $\omega$ , the preset yaw rate control law. The control law was designed to bring each plane from *Init* to its own predetermined final destination  $(x_i^g, y_i^g)$  without colliding. The function  $\omega$  switches depending on the

mode. At the start positions, the aircraft are in  $q = 1$ , (*heading mode*) and rotate until pointing toward their goal positions, so  $\omega(1) = \theta_{goal} - \theta_i$ . Once they reach the desired heading, they switch to  $q = 2$  (*cruise mode*),  $\omega(2) = 0$ , and cruise straight toward the goals. If two aircraft get within a distance  $p$  km of each other, each of the two aircraft enters  $q = 3$  (*avoid mode*) and makes instantaneous  $-90^\circ$  turns, then it follows a half circle with angular velocity  $\omega(3) = c$ . After finishing the circular turn, they make instantaneous turns again until pointing to their own goal positions and return to cruise mode. In case the aircraft sees another aircraft within  $p$  km during the avoid mode, it makes  $-90^\circ$  turn again and executes the same operation as above. This is illustrated in (see Figure 5 *left*). The specification is the minimum distance between all pairs of planes.

When  $\|u\| \leq 0.03km/sec$  and  $p = 5.25km$  a collision among the aircraft was discovered (see Figure 5 *right*) after about 8,600 nodes and 5 parameters were explored. A uniform distribution was used to generate samples, and a simple metric based on a weighted Euclidean distance is utilized

$$\rho = d + w_a|\Delta\theta| \quad (8)$$

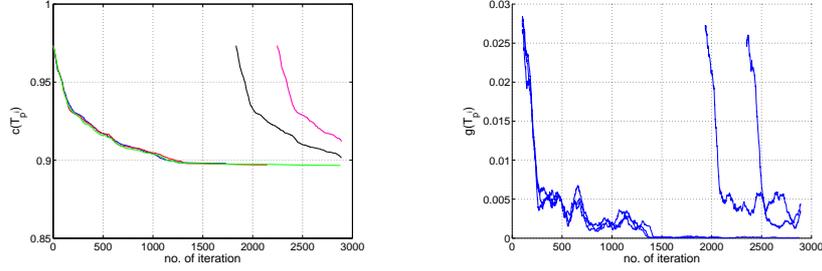
where  $d$  is a Euclidean distance between two  $(x, y)$  positions,  $\Delta\theta \in (-\pi, \pi]$  is a heading difference, and  $w_a$  is a weight factor. Figure 6 shows  $c(\mathcal{T})$  and  $g(\mathcal{T})$  for the trees. Figure 7 shows the coverage of the seed set,  $\mu(\bar{P})$  as new seeds are generated. Three initial seeds are planted and two new seeds are generated until solution trajectory is found. Therefore total 5 seeds are tried to obtain the trajectory.



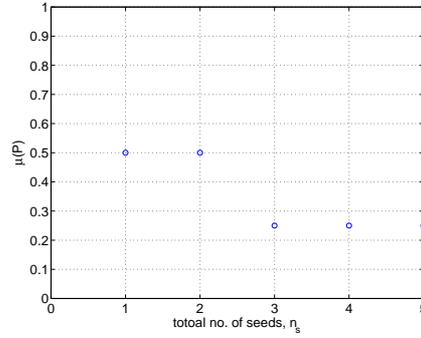
**Fig. 5.** The modes of operation for the aircraft collision avoidance example (left). Testing the aircrafts with  $v = 0.3km/sec$ ,  $\omega = 0.03rad/sec$ , and  $p = [4.5, 5.5]km$  under bounded disturbances  $\|u\| \leq 0.03km/sec$  (right). We define the collision distance as  $1km$ . Circles represent initial positions and rectangles are goal positions. A collision is discovered after exploring about 8,600 nodes with  $p = 5.25km$ .

## 6.2 Example 2: Unmanned blimp control law

In this section, we consider the validation of a feedback control algorithm for waypoint to waypoint navigation of an unmanned outdoor blimp under unpredictable



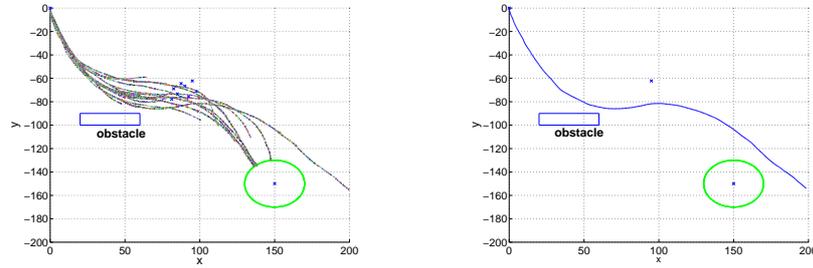
**Fig. 6.** Coverage of the trees. New trees are started when the growth rate slows below a specified threshold ( $\bar{g} = 1 \times 10^{-10}$  used in this example). Solution is discovered in one of the initial seeds.



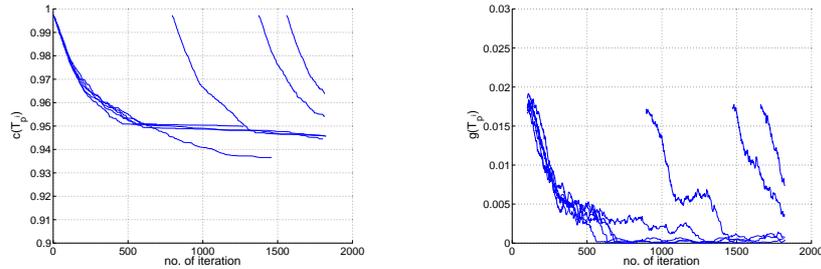
**Fig. 7.** The coverage improves ( $\mu(\bar{P})$  decreases) as new trees are seeded. A collision is discovered at  $n_s=5$ .

but bounded disturbances. The blimp has a 12-dimensional state space. Closed loop control laws use proportional inertial feedback to keep the blimp at the desired altitude with the target speed and to move from one inertial waypoint to the next. Waypoints are generated in the 3-dim space  $(x, y, z)$ . Change of the waypoints can be considered as a system mode change. We bound the input  $u(t)$ , wind disturbance) by limiting the magnitude of the wind gust and the rate of change of wind velocity. The max. change rate of wind direction  $= 0.05(1/s)$  and the max. change rate of wind direction  $= 18^\circ/m$  and the magnitude is bounded as  $\|u\| \leq 0.03 km/sec$ . For detailed description of the feedback control law, sampling strategy, metric design, and the bounded wind disturbance, refer to [15]. In this case the time invariant parameters we will test over is the position of the waypoints. The waypoints are specified by a high level planner which does not account for the blimp's dynamics. We would like to see if it is possible for this planner to specify a waypoint which would cause the blimp to collide with the obstacle.

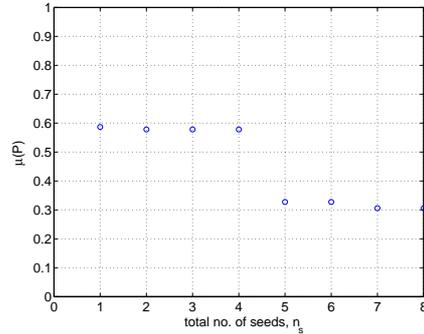
Figure 8 shows the trajectories of the blimp. The initial forward velocity is  $0.5m/sec$ . The target forward velocity is  $1m/sec$ . The starting waypoint is  $[0 \ 0 \ -5]^T$  and the goal waypoint is  $[150 \ -150 \ -10]^T$ . We assume the high level planner can generate intermediate waypoints  $p \in [80 \ 100]^T \times [-80 \ -60]^T \times (-10)$  to avoid a collision. We assume the navigation plan is achieved if the blimp can reach within 20m of the goal waypoint avoiding the obstacle under the wind disturbance. A counter-example is discovered with the intermediate waypoint at  $[95 \ -62.2 \ -10]^T$  after exploring about 9,000 nodes. The solution requires 315 minutes of computation time on 1.4GHz PC. Coverage criteria are shown in Figure 9 and Figure 10. In this application, the RRFT analysis technique allows the designer to efficiently explore the safeness of the blimp closed loop flight control laws for navigation plans in the presence of obstacles.



**Fig. 8.** RRFT of the blimp under wind disturbance  $\|u\| \leq 0.03km/sec$  and uncertain intermediate waypoints (left). Solution trajectory is obtained after exploring about 9000 nodes (right)



**Fig. 9.** Coverage of the trees. New trees are started when the growth rate slows below a specified threshold ( $\bar{g} = 1 \times 10^{-10}$  used in this example). Solution is discovered in one of the initial seeds.



**Fig. 10.** The coverage improves ( $\mu(\bar{P})$  decreases) as new trees are seeded. A solution is discovered at  $n_s=8$ .

## 7 Conclusion

The RRT method is a powerful technique to explore high-dimensional configuration spaces and find motion plans for systems with kinematic and dynamic constraints. In this paper, we presented two enhancements to this method and a novel application. First, we showed how sets of time-invariant parametric uncertainties can be explored with this method to generate a forest of trees. Second, we developed an on-line measure of dispersion that allows us to adapt the growth of the forest to the growth rate of the tree. We presented the application of both methods to the testing and validation of hybrid robot control systems, systems that do not lend themselves to proofs of convergence and stability. In both these examples, because the controller is fixed, the resulting trees do not expand to fill the configuration space. Instead, they fill a "tube" of configuration space that is defined by allowable disturbances and external inputs. The first example showed the ability to analyze multiple-agent systems with uncertainties, while the second example addressed the generation of worst case disturbances for the analysis of full dynamic models of aerial vehicles. The adaptation of the growth of the individual tree to coverage in configuration or state space is a direction of current research and is reported in a forthcoming publication [14].

## References

1. Thomas A. Henzinger, Benjamin Horowitz, Rupak Majumdar, and Howard Wong-Toi. Beyond HyTech: Hybrid systems analysis using interval numerical methods. In *Hybrid Systems : Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 130–144. Springer Verlag, 2000.
2. R. Arkin and T. Balch. *Artificial Intelligence and Mobile Robots*, chapter Cooperative Multiagent Robot Systems. MIT Press, 1998.
3. Eugene Asarin, Thao Dang, and Oded Maler. d/dt: A tool for reachability analysis of continuous and hybrid systems. In *5th IFAC Symposium on Nonlinear Control Systems*, St. Petersburg, Russia, 2001.

4. 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*, 2003.
5. Oleg Butchkarev and Stavros Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In *Hybrid Systems : Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 73–88. Springer Verlag, 2000.
6. J. F. Canny. *The complexity of robot motion planning*. MIT Press, Cambridge, MA, 1988.
7. 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.
8. Peng Cheng, Zuojun Shen, and Steven M. LaValle. RRT-Based trajectory design for autonomous vehicles and spacecraft. *Archives of control science*, 11(3-4):51–78.
9. Alongkritt Chutinam and Bruce Krogh. Computational techniques for hybrid system verification. *IEEE transactions on automatic control*, 48(1):64–75, 2003.
10. E. Frazzoli, M.A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. In *AIAA Conference on guidance, navigation and control*, August 2000.
11. R. W. Ghrist and D. E. Koditschek. Safe cooperative robotic patterns via dynamics on graphs. In *Proceedings of the 8th International Symposium on Robotics Research*, Shonan Village, Japan, oct 1997.
12. J. H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numer. Math.*, (2):84–90, 1960.
13. Lydia E. Kavraki, P. Svestka, J.C.Latombe, and M.H.Overmars. Probabilistic roadmaps for path planning in high dimensional configuration space. *International Transactions on Robotics and Automation*, 12:566–580, 1996.
14. Jongwoo Kim and Joel M. Esposito. Adaptive biasing of RRTs for testing hybrid systems.
15. 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.
16. V. Kumar, M. Zefran, and J. Ostrowski. Intelligent motion planning and control. In *Handbook of Industrial Robotics*. John Wiley and Sons, 1999.
17. George Lafferriere, George J. Pappas, and Sergio Yovine. Symbolic reachability for families of linear vector fields. *Journal of Symbolic Computation*, 32:231–253, 2001.
18. Steven LaValle. From dynamic programming to rrt: Algorithmic design of feasible trajectories. In A.Bicchi, H.I.Christensen, and D.Prattichizzo, editors, *Control Problems in Robotics*, pages 19 – 37. Springer Verlag, 2002.
19. Steven M. LaValle and James J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, 2001.
20. 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.
21. Joshua A. Levine. Sampling-based planning for hybrid systems. Master’s thesis, Case Western Reserve University, September 2003.
22. John Lygeros and George Pappas. A tutorial on hybrid systems: Modeling, analysis and control. 14th IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics, September 1999.
23. M. Mataric. Issues and approaches in the design of collective autonomous agents. *Robotics and Autonomous Systems*, 16(2-4):321–331, Dec 1995.

24. 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.
25. P.Chen and Y.K.Hwang. SANDROS: A dynamic graph search algorithm for motion planning. *IEEE Transactions on Robotics and Automation*, 14(3):390–403, June 1996.
26. Q.Zhao, B.H.Krogh, and P.Hubbard. Generating test inputs for embedded control systems. *IEEE Control Systems Magazine*, 23(4):49–57, 2003.
27. R.Alur, T.A.Henzinger, and P.-H.Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22:181–201, 1996.
28. J.T. Schwartz and M.Sharir. On the 'Piano Movers' problem: II General techniques for computing topological properties of real algebraic manifolds. *Advances in applied mathematics*, 4:298–351, 1983.