# Using Randomization to Find and Optimize Feasible Trajectories for Nonlinear Systems

Peng Cheng Zuojun Shen Steven M. LaValle

Department of Computer Science
Iowa State University
Ames, IA 50011 USA
{chp, zjshen, lavalle}@iastate.edu

**Abstract**

*We present our current progress on the design and experimentation with trajectory planning and optimization algorithms for nonlinear systems that have significant state-space constraints. An overview of our planning method based on Rapidly-exploring Random Trees (RRTs) is given. We show our current planning results for two challenging sets of nonlinear systems: 1) the determination of automobile trajectories through obstacle courses; 2) the design of re-entry trajectories for a reusable launch vehicle model based on the NASA X33 prototype. We also briefly describe some early results on using randomization to optimize trajectories in the presence of state space constraints.*

## 1 Introduction

The design of open loop trajectories for nonlinear systems that have state space constraints is a challenging task. If algorithms can be designed that are able to efficiently find and optimize such trajectories for broad classes of problems, many application areas would be greatly impacted, including robotics, aeronautics, and automotive design. The successful development of such algorithms will most likely depend on a culmination of ideas from both the algorithmic motion planning and nonlinear systems communities.

The primary interest in the motion planning community has been to compute collision free paths in the presence of complicated constraints on the configuration space of one or more controllable bodies (while ignoring differential motion constraints). It is widely known that the class of problems is NP-hard [33], which has caused the focus of research in this area to move from exact, complete algorithms to randomized (or Monte-Carlo) algorithms that can solve many challenging high-dimensional problems efficiently at the expense of completeness. Within the past decade, randomized versions of earlier ideas were developed. The classic notion of a *roadmap* [6, 20, 31], is a network of collision-free paths that captures the configuration-space topology, and is generated by preprocessing the configuration space independently of any initial-goal query. This evolved into a Monte-Carlo variation termed a probabilistic roadmap (PRM) [16], which is formed by selecting numerous configurations at random, and generating a network of paths by attempting to connect nearby points. In contrast to roadmaps, classical *incremental search* ideas are based heavily on a particular initial-goal query, and include methods such as dynamic programming, $A^*$ search, or bidirectional search. These evolved into randomized approaches such as the randomized potential field approach [1], Ariadne's clew algorithm [28], the planner in [15][1], and Rapidly-exploring Random Trees (RRTs) [18].

Once differential constraints are introduced, a challenging problem emerges that involves both nonlinear control and traditional path planning issues. This problem is often referred to as *nonholonomic planning* [2, 21, 12, 22, 30] or kinodynamic planning [5, 7, 8, 10, 9, 12]. Many of these methods, such as those in [2, 9], follow closely the incremental search paradigm. It is generally more challenging to design a roadmap-based algorithm due to the increased difficulty of connecting numerous pairs of states in the presence of differential constraints (often referred to as the steering problem [22]). The first randomized approach to kinodynamic planning appeared in [24], and was based on Rapidly-exploring

[1] We note that the method introduced here is termed a PRM by the authors. Since the method is based on incremental search, we include it here to help categorize the methods based on their conceptual similarities.

Random Trees [23]. In that paper, RRTs were applied to trajectory design problems for hovercrafts and rigid spacecrafts that move in a cluttered 2D or 3D environment. Theoretical performance bounds are presented in [25]. In [13], RRTs were applied to the design of collision-free trajectories for a helicopter in a cluttered 3D environment. In [34], RRTs were applied to the design of trajectories for underactuated vehicles. Recently, the incremental search method in [17] was extended to the case of kinodynamic planning in time-varying environments. Sections 3-5 present the RRT-based trajectory design method and its application to vehicle dynamics and a reusable launch vehicle.

Given the emerging interest in the use of randomized techniques for trajectory design, a growing issue is that the quality of the resulting trajectories may be poor due to the large number of random fluctuations in inputs. This has led us recently to consider the problem of trajectory refinement as a post-processing step to our planning methods. In Section 6 our current attempts to use randomized ideas in this domain are briefly presented.

## 2 Problem Description

The class of problems considered in this paper can be formulated in terms of six components:

1. **State Space:** A topological space, $X$
2. **Boundary Values:** $x_{init} \in X$ and $X_{goal} \subset X$
3. **Collision Detector:** A function, $D : X \rightarrow \{true, false\}$, that determines whether global constraints are satisfied from state $x$. This could alternatively be a real-valued function.
4. **Inputs:** A set, $U$, which specifies the complete set of controls or actions that can affect the state.
5. **Incremental Simulator:** Given the current state, $x(t)$, and inputs applied over a time interval, $\{u(t') | t \leq t' \leq t + \Delta t\}$, compute $x(t + \Delta t)$.
6. **Metric:** A real-valued function, $\rho : X \times X \rightarrow [0, \infty)$, which specifies the distance between pairs of points in $X$ (however, $\rho$ is not necessarily symmetric).

Trajectory planning will generally be viewed as a search in a state space, $X$, for a control $u$ that brings the system from an initial state, $x_{init}$ to a goal region $X_{goal} \subset X$ or goal state $x_{goal} \in X$. It is assumed that a complicated set of global constraints is imposed on $X$, and any solution path must keep the state within this set. A collision detector reports whether a given state, $x$, satisfies the global constraints. We generally use the notation $X_{free}$ to refer to the set of all states that satisfy the global constraints. Local, differential constraints are imposed through the definition of a set of inputs (or controls) and an incremental simulator. Taken together, these two components specify possible changes in state. The incremental simulator can be defined by numerical integration of a *state transition equation* of the form $\dot{x} = f(x, u)$, or can simply be achieved by a simulation software package. Finally, a metric is defined to indicate the closeness of pairs of points in the state space. This metric will be used in Section 3, when the RRT is introduced.

## 3 Trajectory Planning Method

The Rapidly-exploring Random Tree (RRT) was introduced in [23] as a planning algorithm to quickly search high-dimensional spaces that have both algebraic constraints (arising from obstacles) and differential constraints (arising from kinematics and dynamics). The key idea is to bias the exploration toward unexplored portions of the space by sampling points in the state space, and incrementally "pulling" the search tree toward them.

The basic RRT construction algorithm is given in Figure 1. A simple iteration is performed in which each step attempts to extend the RRT by adding a new vertex that is biased by a randomly-selected state, $x \in X$. The EXTEND function, illustrated in Figure 2, selects the nearest vertex already in the RRT to $x$. The "nearest" vertex is chosen according to the metric, $\rho$. The function NEW_STATE makes a motion toward $x$ by applying an input $u \in U$ for some time increment $\Delta t$. This input can be chosen at random, or selected by trying all possible inputs and choosing the one that yields a new state as close as possible to the sample, $x$ (if $U$ is infinite, then an approximation or analytical technique can be used).

```
BUILD_RRT(x_init)
  1  T.init(x_init);
  2  for k = 1 to K do
  3      x_rand ← RANDOM_STATE();
  4      EXTEND(T, x_rand);
  5  Return T
```

```
EXTEND(T, x)
  1  x_near ← NEAREST_NEIGHBOR(x, T);
  2  if NEW_STATE(x, x_near, x_new, u_new) then
  3      T.add_vertex(x_new);
  4      T.add_edge(x_near, x_new, u_new);
```

Figure 1: The basic RRT construction algorithm.

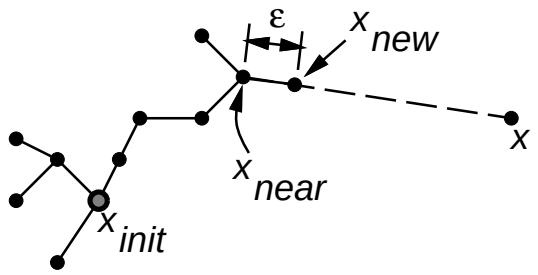



Figure 2: The EXTEND operation.

NEW_STATE implicitly uses the collision detection function to determine whether the new state (and all intermediate states) satisfy the global constraints. For many problems, this can be performed quickly ("almost constant time") using incremental distance computation algorithms [14, 26, 29] by storing the relevant invariants with each of the RRT vertices. If NEW_STATE is successful, the new state and input are represented in $x_{new}$ and $u_{new}$, respectively. The left column of Figure 3 shows an RRT for a 2D holonomic (no differential constraints) planning problem, constructed in a 2D square space. The incremental construction method biases the RRT to rapidly explore in the beginning, and then converge to a uniform coverage of the space [25].

One issue is the size of the step that is used for RRT construction. This could be chosen dynamically during execution on the basis of a distance computation function that is used for collision detection. If the bodies are far from colliding, then larger steps can be taken. Aside from following this idea to obtain an incremental step, how far should the new state, $x_{new}$ appear from $x_{near}$? Should we try to connect $x_{near}$ to $x_{rand}$? Instead of attempting to extend an RRT by an incremental step, EXTEND can be iterated until the random state is reached or until the collision detector reports a collision. The best choice often depends on the type of problem.

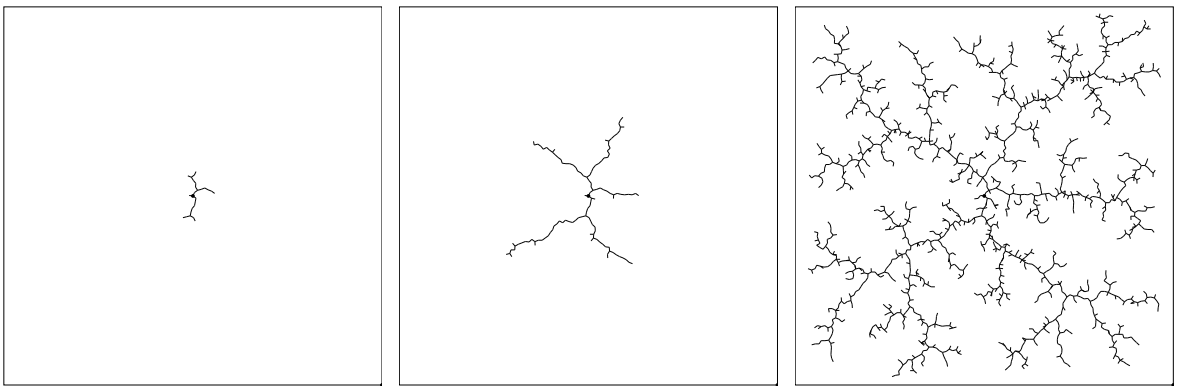


Figure 3: The RRT rapidly explores in the beginning, before converging to the sampling distribution.

The simple RRT can be used in a variety of ways to form path planning methods [25]. The recommended choice depends on several factors, such as whether differential constraints exist, the type of collision detection algorithm, or the efficiency of nearest neighbor computations. By changing the sampling distribution from uniform a biased RRT can be constructed. For example, an RRT can be started at $x_{init}$, and samples from $X_{goal}$ can be occasionally chosen in the place of random samples. Alternatively, a gradual bias of samples around the goal region can be performed. Inspired by classical bidirectional search techniques [32], it seems reasonable to expect that improved performance can be obtained by growing two RRTs, one from $x_{init}$ and the other from $x_{goal}$; a solution is found if the two RRTs meet. This divides the computation time between two processes: 1) exploring the state space; 2) trying to grow the trees into each other. Two trees, $\mathcal{T}_a$ and $\mathcal{T}_b$ are maintained at all times until they become connected and a solution is found. In each iteration, one tree is extended, and an attempt is made to connect the nearest vertex of the other tree to the new vertex. Then, the roles are reversed by swapping the two trees.

## 4 Trajectory Design for Automobiles

Simulators are used extensively to evaluate vehicle performance in the automotive industry. Most of these simulations involve close interaction between a human driver and a simulation system. The experiments presented in this section can be considered as a component that might be used in virtual prototyping of automobile designs. As opposed to requiring human interaction, the RRT-based planner serves as a kind of "virtual stunt driver" that attempts to achieve specified conditions. The nine-dimensional model used here (and others we have used in our experiments) are minor variations of the models considered in [3].

**Variables and constants** The nine-dimensional state vector is $(x, y, r, \psi, \phi, q, \nu, s, \beta)$. The 3D coordinate frame is designed with the $x$ coordinate increasing from left to right, the $y$ coordinate increasing from top to bottom, and the $z$ coordinate inward (to form a right-handed coordinate system). Let $\beta$ be the steering angle. Let $a$ and $b$ be the distance from the front and rear axles to the car center, respectively. Let $\psi$ be the yaw angle of the car. Let $s$ be the forward speed of the car, and let $\nu$ be the sideways speed (arising from slipping). Let $r$ be the angular velocity. Let $\phi$ be the roll (which describes the sideways tilting of the car). Let $q$ be the roll angle rate. Let $\alpha_f$ and $\alpha_r$ be the slipping angle of the front and rear wheels, respectively. These are expressed as

$$\alpha_f = \frac{\nu + ar}{s} - \beta$$

and

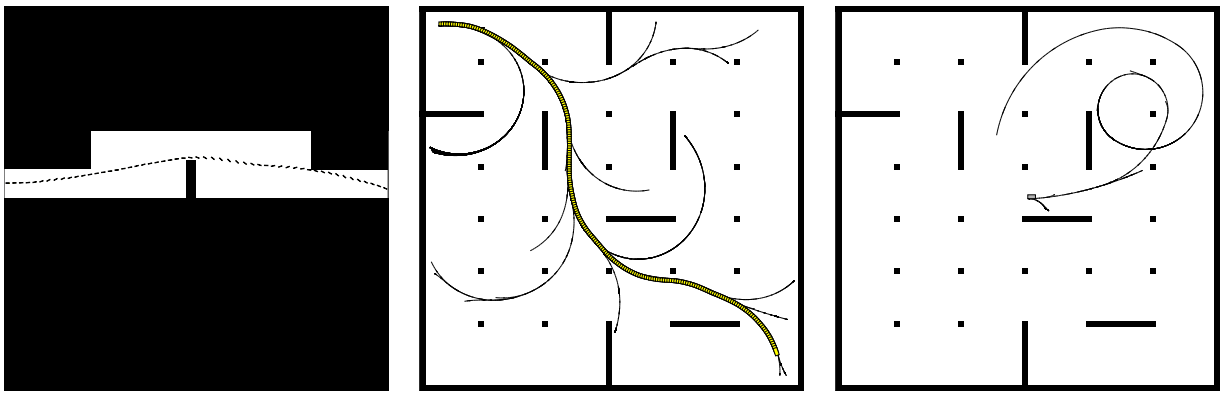
$$\alpha_r = \frac{\nu - br}{s}.$$

Let $C_{\alpha f}$ and $C_{\alpha r}$ be the cornering stiffness between the forces along the $y$ axis, $F_{yf}$ and $F_{yr}$, respectively, on the front and rear wheels. Under some conditions, it is possible for the car to slip sideways. If $N_f\mu/2 > C_{\alpha f}\tan(|\alpha_f|)$, the calculated friction force is less than the maximum possible friction, then $F_{yf} = -C_{\alpha f}\alpha_f$; otherwise, $F_{yf} = \mu N_f Sgn(\alpha_f)(1 - x_f/2)$, in which $Sgn$ denotes the sign function, $\mu$ is a constant, and $x_f = N_f\mu/2C_{\alpha f}\tan(|\alpha_f|)$. Similarly, if $N_r\mu/2 > C_{\alpha r}\tan(|\alpha_r|)$, then $F_{yr} = -C_{\alpha r}\alpha_r$; otherwise, $F_{yr} = \mu N_r Sgn(\alpha_r)(1 - x_r/2)$, in which $x_r = N_f\mu/2C_{\alpha r}\tan(|\alpha_r|)$. Let $M$ be the car mass, and let $I$ be the yaw moment of inertia. Let $H_2$ be the distance from the joint connecting the chassis with the car frame (the chassis and frame are flexibly attached to model a simple suspension system). The constants $K$, $c$, and other details are described an in [3].

**Equations of motion** In the equations, let $h = (-(K - MgH_2)\phi - cq - (F_{yf} + F_{yr})H_2)/I$. The following represent the nine equations of motion: $\dot{x} = s\cos\psi - \nu\sin\psi$, $\dot{y} = s\sin\psi + \nu\cos\psi$, $\dot{r} = (F_{yf}a - F_{yr}b)/I$, $\dot{\psi} = r$, $\dot{\phi} = q$, $\dot{q} = h$, $\dot{\nu} = (F_{yf} + F_{yr})/M - sr - H_2h$, $\dot{s} = u_1$, $\dot{\beta} = u_2$.

The inputs are $u_1$, which is linear acceleration, and $u_2$, which is the change in the steering angle.

**Experiments** Figure 4 shows three separate experiments that were performed for the car model using a simple RRT with a goal bias (the goal was chosen with probability 1/20, instead of a random sample). Figure 4.a involves a lane-changing problem, which is referred to in the automotive industry as the Consumer Union Short Course. The car is moving at 60 m.p.h. from left to right, and planner generates



(a) (b) (c)

Figure 4: a) Double lane change; b) fast driving through an obstacle coursel c) forcing the tires to lift.

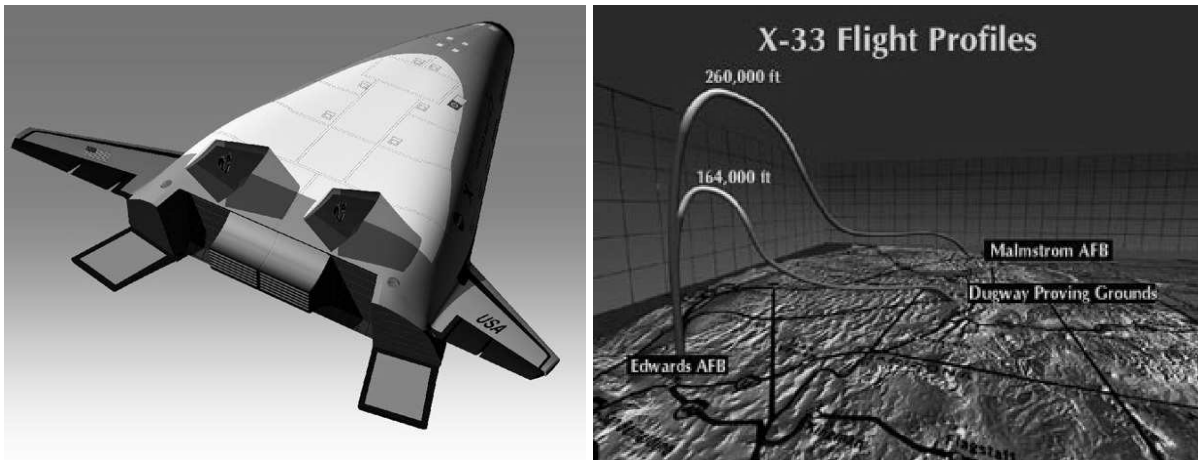



Figure 5: The NASA X-33 prototype reusable launch vehicle.

a steering input that avoids collision. Figure 4.b shows an example in which a car must travel at 108 k.p.h. through an obstacle course. To further complicate the problem, the load on all four tires was computed using the models in [3], and a state constraint was defined that prevents any tire from lifting off of the ground. In Figure 4.c, the car starts at 18 k.p.h., and is requested to reach a state that causes a tire to lift from the ground. The car executes a spiraling path to generate sufficient speed, and then turns while going 190 k.p.h. to achieve the goal. Such a test could be used, for example, in a vehicle safety study.

## 5 Trajectory Design for a Reusable Launch Vehicle

One of the current goals in space transportation is to design a new generation of launch vehicles to dramatically lower the costs of putting payloads in space. Instead of the complex systems in use today, the emphasis is being placed on a simple, fully reusable vehicle. Since July 1996, NASA has commissioned Lockheed Martin Skunk Works to design, build and test the X-33 experimental vehicle, which is depicted in Figure 5. The purpose of the vehicle is to serve as a prototype for demonstrating the feasibility of the new concept. The craft is capable of reaching speeds beyond Mach 13. A formidable challenge is to design a trajectory that will guide the X-33 safely to earth. Skilled engineers sometimes spend weeks designing good trajectories for this system. In this section, we describe our early, but promising results in the design of trajectories for the X-33.

**Variables and constants** Our model is a minor variation of the one presented in [27]. The equations of motion will be expressed in terms of dimensionless variables. In the basic model there are six state

variables, $r, \theta, \phi, V, \gamma$, and $\psi$. Let $r$ be the radial distance from the center of the earth to the flying spacecraft, normalized by the radius of the earth, $R_0 = 6378km$. The longitude and latitude are $\theta$ and $\phi$, respectively. Derivatives of these variables will be taken with respect to dimensionless time $\tau$, in which $\tau = \frac{t}{\sqrt{R_0 g_0}}$ and $g_0 = 9.81m/sec^2$. Let $V$ be the earth-relative velocity, normalized by $\sqrt{R_0 g_0}$. Let $\gamma$ be the flight path angle measured from horizon surface downward. Let $\psi$ be the velocity azimuth angle, measured from the north in a clockwise direction.

The equations of motions will include the variables $D$, $L$, and $\Omega$, which are all functions of state. The quantities $D$ and $L$ are aerodynamic accelerations in g's, which are expressed as

$$L = \frac{1}{2}\rho V^2 S C_L \tag{1}$$

$$D = \frac{1}{2}\rho V^2 S C_D, \tag{2}$$

in which $S$ is a constant based on surface area, and $C_L = h_1(M, \alpha)$ and $C_D = h_2(M, \alpha)$ are the lift and drag coefficients, respectively. These are each functions of Mach number, $M$, and the angle of attack, $\alpha$. The quantity $\rho$ is the air density, which is a function of $r$. The quantity $\alpha$ could be pre-scheduled to be a function of $M$, or $\alpha$ could alternatively be used as a control variable. For simplicity, we assume it is a function of $M$. The functions, $h_1$ and $h_2$ are defined using table lookup and interpolation.

Let $\Omega$ be the rotation rate of the earth normalized by $\sqrt{R_0 g_0}$. Let $\sigma$ be the bank angle, which can be considered as the principle control input of the re-entry flight (the actual control surface inputs on the craft are computed in terms of bank angle).

**Equations of motion** The dimensionless equations of three dimensional motion over a spherical rotating earth are:

$$\dot{r} = V \sin\gamma \tag{3}$$

$$\dot{\theta} = \frac{V \cos\gamma \sin\psi}{r \cos\phi} \tag{4}$$

$$\dot{\phi} = \frac{V \cos\gamma \cos\psi}{r} \tag{5}$$

$$\dot{V} = -D - \left(\frac{\sin\gamma}{r^2}\right) + \Omega^2 r \cos\phi(\sin\gamma\cos\phi - \cos\gamma\sin\phi\cos\psi) \tag{6}$$

$$\dot{\gamma} = \frac{1}{V}\left\{L\cos\sigma + \left(V^2 - \frac{1}{r}\right)\left(\frac{\cos\gamma}{r}\right) + 2\Omega V\cos\phi\sin\psi + \Omega^2 r\cos\phi(\cos\gamma\cos\phi - \sin\gamma\cos\psi\sin\phi)\right\} \tag{7}$$

$$\dot{\psi} = \frac{1}{V}\left[\frac{L\sin\sigma}{\cos\gamma} + \frac{V^2}{r}\cos\gamma\sin\psi\tan\phi - 2\Omega V(\tan\gamma\cos\psi\cos\phi - \sin\phi) + \frac{\Omega^2}{\cos\gamma}\sin\psi\sin\phi\cos\phi\right]. \tag{8}$$

**Boundary conditions** Initial state is given by orbit flight conditions. The goal state is termed the TAEM (Terminal Area Energy Management) point, which is:

$$r(\tau_f) = r_f, \qquad \theta(\tau_f) = \theta_f, \qquad \phi(\tau_f) = \phi_f \tag{9}$$

$$V(\tau_f) = V_f, \qquad \gamma_{min} \le \gamma(\tau_f) \le \gamma_{max}, \qquad \psi(\tau_f) = \psi_f \tag{10}$$

**State constaints** The following constraints define $X_{free}$. Although the X-33 is not in "collision" in the physical sense, failure to satisfy these state-space constraints will result in a collision that must be reported by the collision detector, as described in Section 2.

Normal load constraint:

$$|L\cos\alpha + D\sin\alpha| \leq n_{z_{max}} \tag{11}$$

Dynamic pressure constraint:

$$q \leq q_{max} \tag{12}$$

Heat rate constraint:

$$\dot{Q}_s \leq \dot{Q}_{max} \tag{13}$$

Equilibrium glide constraint:

$$\left[\frac{1}{r} - V^2\right]\left(\frac{1}{r}\right) - L \leq 0 \tag{14}$$

Both $\dot{Q}_s$ and $q$ are complicated functions of state. A slice of the resulting constraints is shown in Figure 6.a.

**Varying models** We have performed experiments with several variations of the model. Each is progressively more complicated than the previous one, due to a difference in the input.

A The state is $(r, \theta, \phi, V, \gamma, \psi)$, and the input is the bank angle, $\sigma$.

B The state is $(r, \theta, \phi, V, \gamma, \psi, \sigma)$, and the input is bank angle velocity, $\dot{\sigma}$.

C The state is $(r, \theta, \phi, V, \gamma, \psi, \sigma, \nu)$, and the input is bank angle acceleration, $\ddot{\sigma}$. The new state variable is defined as $\nu = \dot{\sigma}$.

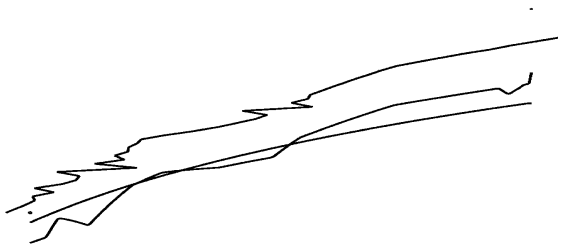
Model A allows instantaneous setting of the bank angle, which is unrealistic in practice. Model B allows the bank angle velocity to be changed. Model C is the most realistic (and most difficult), because only the acceleration of the bank angle is provided by the input. The limits on the bank angle derivatives are $|\dot{\sigma}| \leq 10^o/sec$ and $|\ddot{\sigma}| \leq 5^o/sec^2$.

**Experiments** In our experiments to date, we have experienced considerable success design trajectories using RRT-based planners on Models A and B. In both cases we are able to reach the TAEM. Currently, we are experimenting with Model C. Figure 6 shows a computed trajectory for Model B. The initial state is $r = 1.0086$, $\theta = 4.257$, $\phi = 0.6400$, $V = 0.3630$, $\gamma = 0.01098$, $\psi = 0.6138$, $\sigma = 0.0$. The goal state is $r = 1.0045$, $\theta = 4.305$, $\phi = 0.6967$, $V = 0.1156$, $\gamma = -0.1062$, $\psi = 0.7522$, $\sigma = -0.2250$. A weighted Euclidean metric was used in the RRT algorithm. Figure 6.a shows the projection of the state space constraints, plotted as speed vs. range. The state must remain below the jagged upper curve, and above both of the two lower curves (although the initial state may not initially be below the upper curve). The initial and goal states are indicated with black dots on the right and left of the figure, respectively. Figure 6.b shows a projection of the random samples that were used in the RRT. Instead of using uniform random samples, the samples were concentrated in an elliptical region based on the flight corridor, with some bias towards the goal state. Figures 6.c-f show the paths generated by the RRT. In each case, the initial and goal states are shown as dots. The resulting trajectory comes sufficiently close to the goal state in each of the perspectives shown.

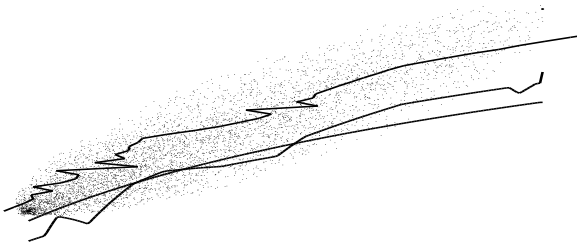
## 6 Path Smoothing and Optimization

Although randomization is helpful in the design of feasible trajectories, it often leads to jagged paths or useless fluctuations in the inputs. This motivations our investigation into the problem of refining trajectories that are computed by our planning algorithms. Existing optimization methods, such as first-order gradient descent [4], are usually not designed to handle complicated, implicit state-space constraints that usually emerge in motion planning, and also make differentiability assumptions on the model (which would might prohibit their use in the models used in Sections 4 and 5.

We are currently exploring the use of randomization to refine trajectories in the presence of state space constraints. Some experiments are shown here for the Dubins car [11]. In this model, there are three state variables (position and orientation). The car moves forward-only at constant speed; the

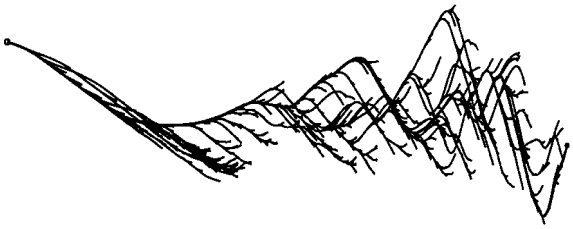


(a) Flight corridor: Altitude vs. speed



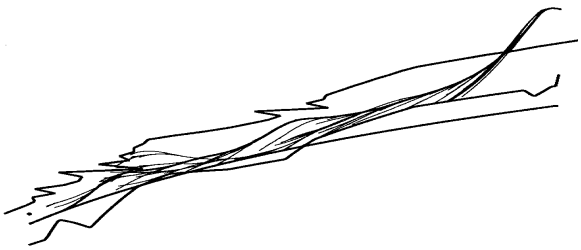
(b) Biased sampling along flight corridor

Altitude vs. range



Heading angle vs. range

Latitude vs. longitude



Altitude vs. speed

Figure 6: Trajectory design results for the X-33 reusable launch vehicle.

only input is the steering angle. The maximum steering angle induces a bound on the curvature of the trajectory.

Two methods are currently being evaluated. For the first method, consider the classical first-order gradient method presented in [4]. For a given control history, $u(t)$ for $t \in [0, t_f)$, this iterative method yields a perturbation, $u(t) + \delta u(t)$, that locally (in the trajectory space) reduces a performance criterion. Eventually, convergence to a locally-optimal solution is obtained; however, the basic method becomes quickly trapped if obstacles exist. One way to avoid this problem is to iteratively select a segment of the path at random, perform the classical optimization on it, and then insert the new segment into the original path. Figure 7.a shows a path computed by an RRT-based planner, and Figure 7.b shows the resulting path after performing a few hundred iterations of the classical gradient descent method on randomly-selected path segments. We note that Dubins' shortest path curves [11, 19] could be used instead of the method in [4]; however, this approach cannot be generalized to many systems. An alternative method is to iteratively compute $\delta u(t)$, and use any perturbation $u(t) + \delta u(t)$ that satisfy collision constraints and reduce the performance criterion. This method can be particularly useful if the equations of motion are not differentiable or not even explicitly known. Figure 7.c shows an initial path, two intermediate paths, and a final optimized path that was obtained using random perturbations. We have yet to consider obstacles in this method, and to compare both methods. Nevertheless, it appears so far that randomization may by useful for path smoothing and optimization.

## 7 Discussion

The experiments presented here show good promise for the use of randomized planning and optimization methods for challenging problems that involve nonlinear systems and complicated state-space constraints. However, many challenges remain. One of the greatest difficulties in motion planning of trajectories is designing a metric that yields good performance. For some systems is may be possible to utilize cost-to-go or Lyapunov functions; however, it also appears valuable to develop randomized

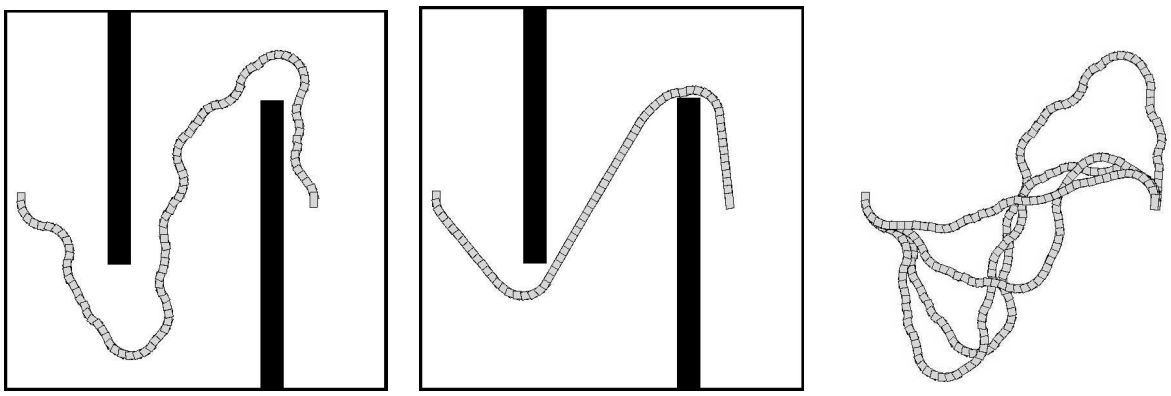



Figure 7: a) A path computed using an RRT-based planner for a Dubins car; b) a refined path obtained by iteratively optimizing random intervals using the first-order gradient method; c) path refinement by random perturbations.

planning methods that have less sensitivity when presented with a poor metric. A substantial amount of work remains on the problem of trajectory optimization in the presence of obstacles. The work presented here represents a step in that direction be considering ways to combine classical optimization methods with modern randomized algorithms.

## Acknowledgments

This work was funded in part by NSF CAREER Award IRI-9875304 (LaValle). We thank Jim Bernard for many helpful discussions on vehicle dynamics, and his suggestions of models. We thank Ping Lu for his suggestions regarding the modeling of the X-33. We thank Francesco Bullo for helpful discussions regarding trajectory optimization.

## References

[1] J. Barraquand, B. Langlois, and J. C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Trans. Syst., Man, Cybern.*, 22(2):224–241, 1992.

[2] J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10:121–155, 1993.

[3] J. Bernard, J. Shannan, and M. Vanderploeg. Vehicle rollover on smooth surfaces. In *Proc. SAE Passenger Car Meeting and Exposition*, Dearborn, Michigan, 1989.

[4] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control.* Hemisphere Publishing Corp., New York, NY, 1975.

[5] J. Canny, A. Rege, and J. Reif. An exact algorithm for kinodynamic planning in the plane. *Discrete and Computational Geometry*, 6:461–484, 1991.

[6] J. F. Canny. *The Complexity of Robot Motion Planning.* MIT Press, Cambridge, MA, 1988.

[7] M. Cherif. Kinodynamic motion planning for all-terrain wheeled vehicles. In *IEEE Int. Conf. Robot. & Autom.*, 1999.

[8] C. Connolly, R. Grupen, and K. Souccar. A Hamiltonian framework for kinodynamic planning. In *Proc. of the IEEE International Conf. on Robotics and Automation (ICRA'95)*, Nagoya, Japan, 1995.

[9] B. R. Donald and P. G. Xavier. Provably good approximation algorithms for optimal kinodynamic planning for Cartesian robots and open-chain manipulators. *Algorithmica*, 14(6):480–530, 1995.

[10] B. R. Donald, P. G. Xavier, J. Canny, and J. Reif. Kinodynamic planning. *Journal of the ACM*, 40:1048–66, November 1993.

[11] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497–516, 1957.

[12] Th. Fraichard and C. Laugier. Kinodynamic planning in a structured and time-varying 2d workspace. In *IEEE Int. Conf. Robot. & Autom.*, pages 2: 1500–1505, 1992.

[13] E. Frazzoli, M. A. Dahleh, and E. Feron. Robust hybrid control for autonomous vehicles motion planning. Technical Report LIDS-P-2468, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 1999.

[14] L. J. Guibas, D. Hsu, and L. Zhang. H-Walk: Hierarchical distance computation for moving convex bodies. In *Proc. ACM Symposium on Computational Geometry*, pages 265–273, 1999.

[15] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *Int. J. Comput. Geom. & Appl.*, 4:495–512, 1999.

[16] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. & Autom.*, 12(4):566–580, June 1996.

[17] R. Kindel, D. Hsu, J.-C. Latombe, and S. Rock. Kinodynamic motion planning amidst moving obstacles. In *IEEE Int. Conf. Robot. & Autom.*, 2000.

[18] J. J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, 2000.

[19] J.-C. Latombe. A fast path planner for a car-like indoor mobile robot. In *Proc. Am. Assoc. Artif. Intell.*, pages 659–665, 1991.

[20] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[21] J.-P. Laumond. Finding collision-free smooth trajectories for a non-holonomic mobile robot. In *Proc. Int. Joint Conf. on Artif. Intell.*, pages 1120–1123, 1987.

[22] J. P. Laumond, S. Sekhavat, and F. Lamiraux. Guidelines in nonholonomic motion planning for mobile robots. In J.-P. Laumond, editor, *Robot Motion Plannning and Control*, pages 1–53. Springer-Verlag, Berlin, 1998.

[23] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State University. <http://janowiec.cs.iastate.edu/papers/rrt.ps>, Oct. 1998.

[24] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 473–479, 1999.

[25] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Workship on the Algorithmic Foundations of Robotics*, 2000.

[26] M. C. Lin and J. F. Canny. Efficient algorithms for incremental distance computation. In *IEEE Int. Conf. Robot. & Autom.*, 1991.

[27] P. Lu and J. M. Hanson. Entry guidance for the X-33 vehicle. *J. Spacecraft and Rockets*, 35(3):342–349, 1998.

[28] E. Mazer, G. Talbi, J. M. Ahuactzin, and P. Bessière. The Ariadne's clew algorithm. In *Proc. Int. Conf. of Society of Adaptive Behavior*, Honolulu, 1992.

[29] B. Mirtich. V-Clip: Fast and robust polyhedral collision detection. Technical Report TR97-05, Mitsubishi Electronics Research Laboratory, 1997.

[30] R. M. Murray and S. Sastry. Nonholonomic motion planning: Steering using sinusoids. *Trans. Automatic Control*, 38(5):700–716, 1993.

[31] C. O'Dunlaing and C. K. Yap. A retraction method for planning the motion of a disc. *Journal of Algorithms*, 6:104–111, 1982.

[32] I. Pohl. Bi-directional and heuristic search in path problems. Technical report, Stanford Linear Accelerator Center, 1969.

[33] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. of IEEE Symp. on Foundat. of Comp. Sci.*, pages 421–427, 1979.

[34] G. J. Toussaint, T. Başar, and F. Bullo. Motion planning for nonlinear underactuated vehicles using hinfinity techniques. Coordinated Science Lab, University of Illinois, September 2000.