Proceedings of the 2002 IEEE
International Conference on Robotics & Automation
Washington, DC • May 2002

Resolution Complete Rapidly-Exploring Random Trees

Peng Cheng Steven M. LaValle

Dept. of Computer Science
University of Illinois
Urbana, IL 61801 USA
{pcheng1, lavalle}@cs.uiuc.edu

Abstract

Trajectory design for high-dimensional systems with nonconvex constraints has considerable success recently; however, the resolution completeness analysis for various methods is insufficient. In this paper, based on Lipschitz condition and the accessibility graph, conditions for resolution completeness are derived. By combining the systematic search with randomized technique, a randomized planner is transformed to be a deterministic resolution complete planner, which shows reliable performance in the simulation.

1 Introduction

Trajectory design for high-dimensional nonlinear systems with nonconvex constraints attracts more and more attention in current research. A challenging problem is shown in Fig. 1. In most of the work [1, 7, 8, 9, 11, 16, 17, 18], dynamic programming methods normally can only find a global optimal solution for low-dimensional problems. Randomized techniques are introduced into trajectory design in [14, 15] to solve the high dimensional problems. Their application in motion planning for autonomous vehicles [10] and nonlinear underactuated vehicles [12, 19] achieves good results. The application of random techniques in trajectory design is also seen in [13]. The general idea in these methods is to incrementally build a search graph from the initial state and extend it toward the goal state. To avoid searching the same state repeatedly and terminate the searching in finite number of iterations, many methods have been used to discretize the configuration or state space such that there are only finite number of nodes in the search graph. In [9], the discretization is based on the grid built from the acceleration bounds and fixed time step. Nonuniform boxes is used to discretize the configuration space in [18]. The configuration space is decomposed into disjoint parallelepipeds of equal size and asymptotic completeness is derived [1]. More generally in [2], a partition is used to discretize the state space, and there is no restriction on the form of sets in the partition. However, how to choose appropriate discretization resolution to guarantee that the existing solution will be found is still unknown.

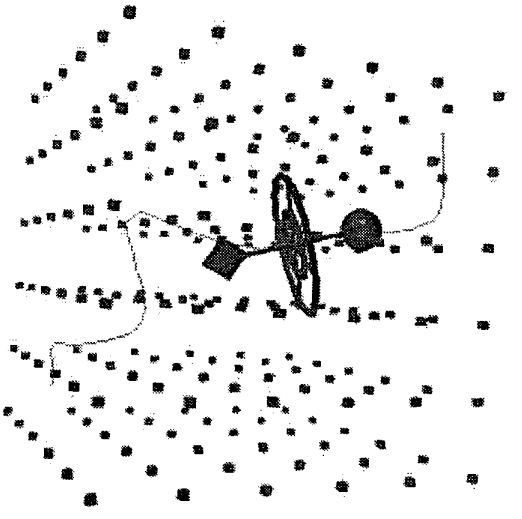

Figure 1: A trajectory design for an underactuated 12-dimensional spacecraft with three thrusters moving in a 3D grid.

In our paper, by considering an accessibility graph and Lipschitz conditions, *resolution completeness* conditions are presented, which means that the planner can find an existing solution only if the discretization resolution is in a specified range. It is different from the asymptotic completeness in [1], in which an exact algorithm for a small-time locally controllable system is complete when the control period is small enough, search depth is big enough and discretization resolution is big enough.

2 Problem Formulation

The trajectory design problem, Δ, considered in this paper is defined as a nine-tuple $(X, D, x_{init}, X_{goal}, U, \delta t, \rho, \tau, f)$: The X is state space, an n-dimensional

compact differentiable manifold. The D is constraint satisfaction function determining whether global constraints are satisfied for a state. The x_{init} and X_{goal} are boundary conditions. The U is the complete set of h-dimensional inputs that might be dependent or independent of the state. The δt is the fixed control period which denotes the period during which a constant $u \in U$ is applied. The ρ is metric function $\rho : X \times X \rightarrow [0, \infty)$ defined on X. The τ is the solution tolerance, a real number $\tau \geq 0$. For an algorithm with analytical results, the end of a trajectory, x_{end}, always lies in X_{goal} so that $\tau = 0$; for a numerical algorithm, the x_{end} always meets the condition $\exists x_{goal} \in X_{goal}$ such that $\rho(x_{end}, x) < \tau$ and $\tau > 0$. f is equation of motion.

The objective of the trajectory design problem is to find a *solution control function*, $u : [t_0, t_f] \rightarrow U$, in which t_0 is the starting time and $t_f = t_0 + K\delta t$, and its corresponding *solution path*, $\pi : [t_0, t_f] \rightarrow X_{free}$, which meet the following conditions: $u(t) = u_i \in U, \forall t \in [t_0 + k\delta t, t_0 + (k+1)\delta t), i = 1, 2, \cdots, h, k = 0, 1, \cdots, K-1$. $\pi(t) \in X_{free}, t \in [t_0, t_f]$, in which

$$\pi(t) = \begin{cases} x_{init}, \\ \quad \text{if } t = t_0; \\ \pi(t_0 + k\delta t) + \int_{t_0+k\delta t}^{t} f(x, u, t)dt, \\ \quad \text{if } t \in (t_0 + k\delta t, t_0 + (k+1)\delta t] \\ \quad \text{and } k = 0, 1, \cdots, K-1. \end{cases}$$

$\exists\, x_{goal} \in X_{goal},\ \rho(\pi(t_f), x_{goal}) \leq \tau$. We also represent π by either an input sequence $\pi_u = \{u_1, u_2, \cdots, u_K\}$ in which $u_i = u(t_0 + (i-1)\delta t)$ or a state sequence $\pi_x = \{\, x_0 = x_{init}, x_1, \cdots, x_K\}$ in which $x_i = \pi(t_0 + i\delta t)$.

3 The accessibility graph

The conditions of resolution completeness are based on the accessibility graph, which describes the connectivity between all of the states accessible from x_{init}.

Definition of the accessibility graph Before introducing the accessibility graph, the following definitions are given:

System transition equation, $s(u, x, t, \delta t)$ is a function: $s : U \times X \times [t_0, t_f] \times (0, \infty) \rightarrow X$. The $s(u, x, t, \delta t)$ calculates the new state by applying a constant input $u \in U$ on a state $x(t) \in X$ for a constant time δt. The $s(u, x, t, \delta t) = x + \int_t^{t+\delta t} f(x, u, t)dt$ without considering global constraints.

Violation-free system transition equation, $\tilde{s}(u, x, t, \delta t)$ is a function: $\tilde{s} : \tilde{U} \times X_{free} \times [t_0, t_f] \times (0, \infty) \rightarrow X_{free}$, in which $\tilde{U}$ is a function of $x \in X$. The $\tilde{s}(u, x, t, \delta t)$ is actually the same as $s(u, x, t, \delta t)$ except for constraints on u and x and its result value. For $\forall u \in \tilde{U}$ and $x \in X_{free}$, $\tilde{s}(u, x, t, \delta t) = s(u, x, t, \delta t) = x + \int_t^{t+\delta t} f(x, u, t)dt$ and $\forall t' \in (t, t + \delta t]$ such that $x + \int_t^{t'} f(x, u, t)dt \in X_{free}$. It is a system transition equation considering the global constraints.

Problem Δ corresponds to a multi-stage process. Each stage corresponds to a moment and has a set of accessible states. In control literature, this system is referred to as quantized control systems [5, 6], in which stability of the system is their research interest.

Assume the system begins at time t_0. Let stage k correspond to time $t_0 + (k-1)\delta t$, and let N_k be the accessible state set at stage k. We obtain the following sequence:

Stage 1, $N_1 = \{x \mid x = x_{init}\}$; Stage 2, $N_2 = \{x \mid x = \tilde{s}(u, x', t_0, \delta t), u \in \tilde{U}(x'), x' \in N_1\}$; Stage k, $N_k = \{x \mid x = \tilde{s}(u, x', t_0 + (k-1)\delta t, \delta t), u \in \tilde{U}(x'), x' \in N_{k-1}\}$.

Based on the above sequence, the definition of the accessibility graph follows:

Def 3.1 *Accessibility graph, G_∞:*
For a given Δ, G_∞ is a graph $G_\infty(N_\infty, E_\infty)$, in which $N_\infty = \bigcup_{k=1}^{\infty} N_k$ *and* $E_\infty = \{e(x, x') \mid x, x' \in N_\infty, \exists u \in \tilde{U}(x), x' = \tilde{s}(u, x, t, \delta t)\}$.

If $\exists x_i \in N_i$ and $\exists x_j \in N_j$ such that $i \neq j$ and $x_i = x_j$, then we say G_∞ has *cyclic paths* and the corresponding Δ is *cyclic*; otherwise, Δ is *acyclic*. If $\exists m > 1$ such that $N_m - \bigcup_{k=1}^{m-1} N_k = \varnothing$, then N_∞ is finite; otherwise, N_∞ is infinite.

For a given Δ, a state $x \in X_{free}$ is Δ locally accessible if and only if $\exists r > 0$ and $\forall x', \rho(x, x') < r$ such that $\exists \pi, \pi(t_0) = x$ and $\pi(t_f) = x'$. If there exists a Δ locally accessible state in X, the N_∞ is infinite and discretization techniques should be used to guarantee the completeness.

Relation between the accessibility graph and the planner From the above description, it is clear that once Δ is given, G_∞ is fixed. If there exists a solution path from x_{init} to x_{goal} under the control period δt and tolerance τ, it must exist in G_∞. To illustrate this relation, we define the following class of planners.

Def 3.2 *Graph search planners, Γ:*
The class of all planners for a given Δ, in which a graph $G_{sub}(N_{sub}, E_{sub}) \subseteq G_\infty$.

4 Algorithm description

The resolution complete RRT algorithm, which our analysis is based on, is given here. The basic RRT algorithm is presented in [14]. Initially, the x_{init} is the only node of G_{sub}. For each iteration, a random state $x_{rand} \in X$ is chosen, and $x_{near} \in N_{sub}$ is selected as the nearest state to x_{rand} according to a metric function ρ. For x_{near}, a best input u_{best} is chosen to generate a

new state x_{new} which is closest to x_{rand} of all states generated by applying one step control from x_{near}. If x_{new} satisfies the global constraints, the x_{new} will be added to G_{sub}. The performance of RRTs degrade when ρ poorly approximate the real path cost [4]. Much time will be wasted in choosing states destined to collision numerous times, and the probability of expanding the solution path will decrease with more and more iterations. In [4], we presented the basic RC-RRT algorithm, RC-RRT$_1$, in which exploration information and the constraint violation frequency (CVF) are collected and used during the exploration to reduce metric sensitivity. Exploration information records whether an input has been applied on a state. The CVF approximates the constraint violation tendency (CVT). Because it is impractical to calculate the CVT, the CVF is used to approximate the CVT in a way that the CVF will never exceed the CVT. To choose x_{near}, exploration information, the CVF and ρ are used. If all of the inputs have been applied on a state x, the x will be discarded; otherwise, the probability of not choosing x equals to the CVF of x. To choose u_{best}, exploration information and ρ are used. If $u \in U$ has been considered for x_{best}, the u will be discarded; otherwise, it will be applied to x_{best} to generate x'_{new}. The collision free x'_{new} with the shortest distance to x_{rand} will be added to G_{sub}. If x'_{new} is in the collision region, the CVF information will be collected. RC-RRT$_1$ shows reliable performance in [4]; however, its completeness is limited to *acyclic* G_∞ with finite N_∞. To provide a resolution complete planner for general G_∞, two extensions of RC-RRT$_1$ are presented in this paper.

Using neighborhoods to exclude repeated states The RC-RRT$_1$ employs only exploration information to exclude repeated states; however, if G_∞ is *cyclic* or has infinite N_∞, the planner might continue running even though the entire state space has been explored. To ensure that the planner terminates in finite time for any Δ, a covering of X, $\Pi(X)$, is used to discretize X. The $\Pi(X)$ divides X into n_p sets, called neighborhoods in the following description, $S^1, S^2, \cdots, S^{n_p}$, which satisfy the following conditions: $X = \bigcup_{i=1}^{n_p} S^i$; $\exists \kappa > 0, \forall i \in [1,2,\cdots,n_p], \mu(S^i - \bigcup_{j\in[1,2,\cdots,i-1,i+1,\cdots,n_p]} S^j) > \kappa$, in which μ is the Lebesgue measure. These neighborhoods can be in different forms such that various $\Pi(X)$ will be generated. Without losing generality, a ball neighborhood, $B(x, r_b) = \{x' \mid \rho(x, x') < r_b, x' \in X\}$, is used in our methods. Instead of adding the new state, x_{new}, directly into G_{sub} in the original RRT, we check if x_{new} is in $B(N_{sub}, r_b) = \bigcup_{x\in N_{sub}} B(x, r_b)$. If $x_{new} \in B(N_{sub}, r_b)$, x_{new} will be discarded; otherwise, it will be added to G_{sub}. We call this extension RC-RRT$_2$.

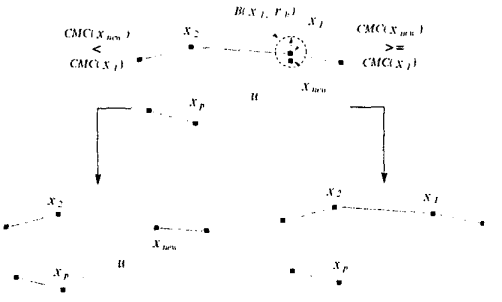

Figure 2: The method to keep the minimum cost path.

Returning an optimal solution in the explored state space Dynamic programming explores all of the state space and returns a global optimal solution; however, randomized methods sacrifice global optimality by only exploring part of the state space. In RC-RRT$_3$, we combine dynamic programming with randomized methods to return an optimal path in the explored state space. Assume the current minimum path cost, $CMC(x)$, is calculated for $\forall x \in N_{sub}$. In Fig. 2, if x_{new} is successfully generated from x_p, the following procedure will be used instead of adding x_{new} directly into G_{sub}. If $\forall x \in N_{sub}, x_{new} \notin B(x, r_b)$, it will be added to the graph, and its path cost $CMC(x_{new}) = CMC(x_p) + C(x_p, u, \delta t)$, in which $u \in U$ is the input leading x_p to x_{new} and $C(x_p, u, \delta t)$ is a cost function; otherwise, if $x_{new} \in B(x_1, r_b)$ and $CMC(x_{new}) \geq CMC(x')$, x_{new} will be discarded and the graph is unchanged. If $CMC(x_{new}) < CMC(x_1)$, the x_{new} replaces node x_1 in G, the link between x_p and x_{new} is added, and the link between x_2 and x_1 is removed. Under these procedures, if a solution is found, it will be the minimum cost path in the explored state space.

It is apparent that the size of neighborhoods (i.e., the discretization resolution) will affect the completeness of the algorithm. It needs to be small enough to avoid missing a solution for a given tolerance. If there exists a computation error, the discretization resolution should be big enough to avoid returning a wrong solution. We call a method *resolution complete* if the completeness of the method holds only for a specified resolution range.

5 Algorithm analysis

The first lemma shows that CVF will not affect resolution completeness by preventing the path from expanding. Its proof can be seen in [3].

Lemma 5.1 *For a given Δ, suppose there exists a path, π, from x_{init} to $x_{goal} \in X_{goal}$; for $\forall x \in \pi_x$, RC-RRTs using the constraint violation frequency will choose it with a strictly positive probability in a finite number of iterations.*

Theorem 5.2 *(Resolution Completeness) Suppose the computer precision (round-off) error is considered and* $\exists \eta > 0$ *such that* $\forall x \in X, \varsigma(x) < \eta$. *If there exists a solution* π *of length* K *with tolerance* $\frac{\tau}{2}$ *for a given* Δ, *then RC-RRT*$_1$, *RC-RRT*$_2$ *and RC-RRT*$_3$ *using* $B(x,\epsilon)$ *will find a solution with tolerance* τ *if the following conditions are satisfied:*

1. *The system transition equation* $s(u,x,t,\delta t)$ *meets the Lipschitz condition:* $\forall u \in U, x_1, x_2 \in X, \rho(s(u,x_1,t,\delta t), s(u,x_2,t,\delta t)) < L_s \rho(x_1,x_2)$.
2. *The computer precision error* $\eta < \frac{\tau(L_s-1)}{2({L_s}^{K+1}-1)}$.
3. *The ball region radius* $\frac{{L_s}^{K_c+1}-1}{L_s-1}\eta < \epsilon < \min\{\frac{\tau(L_s-1)}{2(L_s^K-1)} - \frac{{L_s}^{K_c+1}-1}{L_s^K-1}\eta, \min_{k=0,\cdots,K}(\rho(\pi(t_0+k\delta t), \pi(t_0+(k+1)\delta t)))\}$, *in which* K_c *is the length of the largest cyclic path in the search graph* G_∞.
4. *There exists a collision-free "tunnel" around* π *such that* $D(x) = true$ *for* $\forall x, \rho(x,\pi(t)) \le \tau$ *and* $t \in [t_0, t_f]$. *(The "tunnel" here is different from the "tube" in [9]. The "tunnel" is used to guarantee the completeness and only related to* π *and* τ; *however, the "tube" is used for the safety consideration and is related to robot sensors and velocity error correction rate.)*

Proof: For a given Δ, G_∞ is fixed. If every parameter is represented algebraically, an exact geometric computation method [20] can be taken and no computation error will exist. RC-RRTs generate $G_{sub} \subset G_\infty$. The other computation method uses floating point numbers to represent the parameters resulting in the numerical computation error. Let $\breve{G}_{sub}$ denote G_{sub} generated by using the floating point computation. The $\breve{G}_{sub}$ approximates G_∞ with the computation error. It is possible that the algorithm will report a wrong solution when the computation error causes a non-solution path to enter the goal region. The computation error is related to the computer architecture, algorithm, and state transition equation. It is difficult to analyze all of these computation error factors. To show the effect of the computation error on the completeness, a simple computation error model is used. In this model, only $\varsigma(x)$ is considered, and all other computations are accurate. Let $U_s = (u_1, u_2, \cdots, u_p)$ denote an input sequence. By applying U_s on a state x at time t_0, the final state is $\Phi(U_s,x,t_0) = s(u_p, (s(u_{p-1}, \cdots (s(u_1,x,t_0,\delta t))\cdots, t_0 + (p-2)\delta t, \delta t)), t_0 + (p-1)\delta t, \delta t)$. Considering exact computation and the Lipschitz condition, the error $\rho(\Phi(U_s,x_1,t_0), \Phi(U_s,x_2,t_0)) < {L_s}^q \rho(x_1,x_2)$. Considering floating point methods and the computer precision error in each stage, the error $\rho(\Phi(U_s,x,t_0), \Phi(U_s,x',t_0)) < \frac{{L_s}^{q+1}-1}{L_s-1}\eta$.

Because of different G_∞, we will provide completeness arguments for RC-RRTs in the following three parts:

Part I. Accessibility graph with finite N_∞ and no cyclic paths All RC-RRTs have completeness in this case. Only completeness conditions of RC-RRT$_1$ are presented here because conditions for RC-RRT$_2$ and RC-RRT$_3$ for this G_∞ is same as those for other G_∞, which will be discussed in Part II.

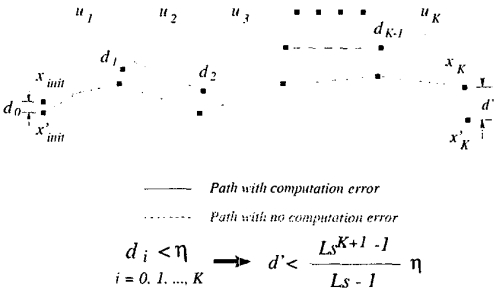

Figure 3: Accumulated computation error resulted from the computer precision error.

If the exact computation method is used, the completeness can be guaranteed by exploring until $G_{sub} = G_\infty$.

If floating point computation is used, the analysis is shown in Fig. 3. For the solution path $\pi_u = \{u_1, u_2, \cdots, u_K\}$, the initial state precision error $\eta = \rho(x_{init}, x'_{init})$ will lead to the error $\rho(\Phi(\pi_u, x_{init}, t_0), \Phi(\pi_u, x'_{init}, t_0)) < \frac{{L_s}^{K+1}-1}{L_s-1}\eta$ at the final state. If $\eta < \frac{\tau(L_s-1)}{2({L_s}^{K+1}-1)}$, the final computation error $\rho(f(P_s,x_{init}), f(P_s,x'_{init})) < \tau$.

Part II. Accessibility graph with finite N_∞ and cyclic paths Because only exploration information is used in RC-RRT$_1$, it might go into an infinite loop on the cyclic path, but RC-RRT$_2$ and RC-RRT$_3$ are complete for this G_∞.

Using exact computation, G_{sub} generated by RC-RRT$_2$ and RC-RRT$_3$ using $B(x,0)$ is always a subset of G_∞. Because after at most $|U|$ iterations a new node will be added to G_{sub} and there are finite number of nodes in N_∞, the G_{sub} will equal to G_∞ after finite number of iterations and the completeness is achieved. If $B(x,\epsilon), \epsilon > 0$ is used to exclude repeated states, a set of nodes, $S_x \subset N_\infty$ might be replaced by only one state node in N_{sub}. If ϵ is too large, a solution might be missed by exceeding the tolerance τ. Let $S_x = \{x_1, x_2, \cdots, x_k\} \subset N_\infty$, and $x' = x_i, x_i \in S_x$ will represent S_x in G_{sub} if x_i is the first state node visited in S_x during the search and $\forall x \in S_x, \rho(x,x_i) < \epsilon$. Assume a solution $\pi_x = (x_0, x_1, x_2, \cdots, x_p, \cdots, x_K)$ exists, in which x_p is the first state in π_x replaced by state, x_i, such that $x_p \in S_x$ and $x_p \neq x_i$. Let $x_K = \Phi(u_p, x_p, t)$, in which $x_p = \pi(t)$ and $u_p = \{u_{p+1}, u_{p+2}, \cdots, u_K\}$ is the input sequence leading x_p to x_K. The error resulting from ϵ in $x_i \in \pi_x$ will be $L_s^{(K-p)}\epsilon$. Under the worst-case when $x_1 \in \pi_x$ is replaced, $\epsilon < \frac{\tau(L_s-1)}{2(L_s^K-1)}$ will guarantee a solution with tolerance τ will be found.

Considering the computation error, The $B(x,0)$ cannot be applied because the two identical states resulting from a cyclic path might appear different because of the computation error. The $B(x,\epsilon), \epsilon > 0$ must be used to control the computation error. If ϵ is too small, a single state may result in two different state nodes in N_{sub}. Assume that K_c is the maximum length of all of cyclic paths, if ball radius $\epsilon > \frac{L_s{}^{K_c+1}-1}{L_s-1}\eta$, then two identical states will stay in the same ball region for any cyclic path in G_∞. Also, the $\epsilon < \frac{\tau(L_s-1)}{2(L_s^K-1)} - \frac{L_s{}^{K_c+1}-1}{(L_s-1)L_s^{(K-1)}}\eta$ guarantees a solution with tolerance τ will be found.

Part III. Accessibility graph with infinite N_∞ Only RC-RRT$_2$ and RC-RRT$_3$ using $B(x,\epsilon), \epsilon > 0$ are suitable for this problem. It is easy to prove that RC-RRT$_2$ and RC-RRT$_3$ using $B(x,\epsilon), \epsilon > 0$ will only generate G_{sub} with finite N_{sub} to approximate G_∞ by considering that $\mu(X_{free})$ is a finite constant, and $\exists c > 0$ such that $\forall x \in N_{sub}, \mu(B(x,\epsilon) - B(N_{sub} - \{x\}, \epsilon)) > c$, in which c is a constant and $\mu(S)$ is the Lebesgue measure function defined on X. Similar to above proof: For exact computation, the $\epsilon < \frac{\tau(L_s-1)}{2(L_s^K-1)}$ will guarantee the solution is found; For floating point computation, the complete condition is $\frac{L_s{}^{K_c+1}-1}{L_s-1}\eta < \epsilon < \frac{\tau(L_s-1)}{2(L_s^K-1)} - \frac{L_s{}^{K_c+1}-1}{(L_s-1)L_s^{(K-1)}}\eta$.

Collision-free "tunnel" Because of the computation error or the discretization, the path generated by the planner will be around the neighborhood of π. The path might be lost when it collides with obstacles in the neighborhood of π. To prevent this from happening, we require a collision-free "tunnel" around π exists such that $D(x) = true$ for $\forall x, \rho(x,\pi(t)) \leq \tau$ and $t \in [t_0, t_f]$.

Keep the tree expanding Because $B(x,\epsilon)$ is used to discretize the state space, new nodes in the explored neighborhood will be discarded. In order to guarantee that π can be found, we require that $\epsilon < \min_{k=0,\cdots,K}(\rho(\pi(t_0+k\delta t), \pi(t_0+(k+1)\delta t)))\}$. ∎

Corollary 5.3 *If there exists a solution π of length K with tolerance $\frac{\tau}{2}$ for a given Δ, and π is optimal in $G_{sub} \subseteq G_\infty$, RC-RRT$_3$ will find π with tolerance τ if resolution completeness conditions are satisfied and G_{sub} are generated by RC-RRT$_3$ when π is found.*

Since the above proof is only related to the accessibility graph and has no special restriction on the form of the covering sets, so it is applicable to other incremental planners and any form of covering sets can be used to discretize the state space. Similarly, there is no assumption about the input set, the conditions will still be true for different input space.

To prove RC-RRTs are deterministically resolution complete, the following theory shows that the planner will answer the query in finite number of iterations. For a Γ method, x_{init} is initially the only node in N_{sub}. New nodes might be added in each iteration. Eventually, after a finite number of iterations, no more new nodes can be added because of the discretization. We use $\bar{G}_{sub}(\bar{N}_{sub}, \bar{E}_{sub})$ to represent the final G_{sub}. Because nodes in N_∞ might be added to G_{sub} in different order, there will be different $\bar{G}_{sub}$.

Theorem 5.4 *(Deterministic Completeness) For a given Δ, assume each node in N_∞ costs one unit of space for storing. If the resolution completeness conditions are satisfied, let m be the number of inputs in U, n be number of nodes in $\bar{G}_{sub}$, $n_{max} = \max_{\forall \bar{G}_{sub}} n$, and let n_p be the number of sets in $\Pi(X)$ which corresponds to the chosen neighborhood used in RC-RRT$_2$ and RC-RRT$_3$. An RC-RRT$_2$-based planner or RC-RRT$_3$-based planner needs at most $n_{max}m \leq n_p m$ iterations and $n_{max} \leq n_p$ units of space to determine whether there exists a solution for Δ.*

Proof: To bound the number of iterations in the worst-case, we first need to find the upper bound for the number of nodes in $\bar{N}_{sub}$. n_∞ represents the number of nodes in N_∞.

Upper bound for number of nodes in $\bar{N}_{sub}$: For a given Δ, whose G_∞ has a finite N_∞, a $\Pi(X)$ can be constructed, in which $\forall S^i \in \Pi(X)$, if $\exists n_1 \in N_\infty \cap S^i$, then $\forall n_2 \in N_\infty - \{n_1\}, n_2 \notin S^i$. Thus, $n_\infty \leq n_p$. Because the resolution completeness conditions are satisfied, using this $\Pi(X)$, any $\bar{N}_{sub}$ has the same number of nodes as N_∞. Thus $n_{max} = n_\infty \leq n_p$.

For Δ, whose G_∞ has infinite N_∞, a $\Pi(X)$ is required to meet the resolution completeness conditions. For the $\Pi(X)$, $\exists S^i \in \Pi(X)$ and $\exists N_i = \{n_{i1}, n_{i2}, \cdots, n_{ij} \in N_\infty, j \geq 2\}$ such that $N_i \subset S^i$. According to RC-RRT$_2$ and RC-RRT$_3$ algorithm, only $n_{ik} \in N_i$ will be in $\bar{N}_{sub}$ when n_{ik} is the first node in N_i to be added to G_{sub}. $\forall n \in N_i - \{n_{ik}\}$ will not be in $\bar{N}_{sub}$. Because $\forall \bar{G}_{sub}$ and $\forall S^i \in \Pi(X)$, if $\exists n_1 \in N_\infty \cap S^i$, then $\forall n_2 \in N_\infty - \{n_1\}, n_2 \notin S^i$, $n_{max} \leq n_p$.

Space requirement: Because we assume each node costs one unit of space, RC-RRT$_2$ and RC-RRT$_3$ need at most $n_{max} \leq n_p$ units of space.

Number of iterations: RC-RRT$_2$-based planner and RC-RRT$_3$-based planner will stop searching when all of the inputs for all state nodes existing in N_{sub} are EXPANDED. Marking one input for a state node as EXPANDED needs at most one iteration. Thus at most $n_{max}m \leq n_p m$ iterations are required to determine whether a solution exists for a given Δ. ∎

From here we can see that the resolution completeness and the asymptotic completeness are about how the resolution affects the completeness of a planner; however, the deterministic completeness and probabilistic completeness are about how the computing time affects the completeness. Different completeness combination will result different planners.

6 Simulation results

To test the performance of new planners, some modifications are made to the experiments in [4]. For each

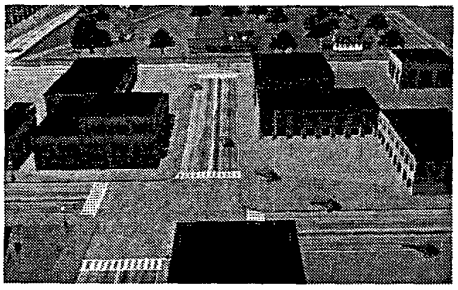

Figure 4: The virtual driving experiment.

experiment, we first set up the problem to make sure a solution exists, then ran the planners for 50 trials on the problem. The experiment will keep running until all trials are completed and solutions are found 50 times. If one of the trials does not find the solution after 48 hours, the experiment is aborted.

For the virtual driving experiment (Fig. 4), the goal biased RC-RRT$_1$-based planner found the solution in each trial with an average of 1629.62 seconds, in which "goal biased" means choosing x_{goal} as x_{random} with a given probability. For the goal biased RRT-based planner, we did fifty trials twice. In the first fifty trials, the first trial failed after 48 hours; in the second fifty trials, the eighth trial failed after 48 hours. For the trajectory design for a floating spacecraft (Fig. 1, the system equation is in [4]). The Dual-RC-RRT$_1$-based planner solved the problem fifty times with an average of 8059.31 seconds, in which "Dual" means that two trees are extended from x_{init} and x_{goal}, respectively, to obtain a solution. We did the fifty trials on the goal biased Dual-RRT-based planner twice. They both failed in the first trial after 48 hours. From experimental results, we can see that our new methods solved those challenging problems with reasonable average running time.

7 Conclusion

In this paper, based on the accessibility graph and Lipschitz conditions, resolution complete RRTs are designed and the conditions for resolution completeness are derived. The new planner shows reliable performance in the simulation.

Acknowledgments

Thanks James Bernard and Andrew Olson for their help. This work was funded in part by NSF CAREER Award IRI-9875304 (LaValle) and NSF IIS 0118146.

References

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

[2] D. P. Bertsekas. Convergence in discretization procedures in dynamic programming. *IEEE Trans. Autom. Control*, 20(3):415–419, June 1975.

[3] P. Cheng. Reducing RRT metric sensitivity for motion planning with differential constraints. Master's thesis, Iowa State University, Ames, IA, 2001.

[4] P. Cheng and S. LaValle. Reducing metric sensitivity in randomized trajectory design. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2001.

[5] D. F. Delchamps. Extracting state information from a quantized output record. *Systems and Control Letters*, 13:365–371, 1989.

[6] D. F. Delchamps. Stabilizing a linear system with quantized output record. *IEEE Trans. AUtom. Control*, 35(8):916–926, 1990.

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

[8] B. R. Donald and P. G. Xavier. Provably good approximation algorithms for optimal kinodynamic planning: Robots with decoupled dynamics bounds. *Algorithmica*, 14(6):443–479, 1995.

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

[10] 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.

[11] G. Heinzinger, P. Jacobs, J. Canny, and B. Paden. Time-optimal trajectories for a robotic manipulator: A provably good approximation algorithm. In *IEEE Int. Conf. Robot. & Autom.*, pages 150–155, Cincinnati, OH, 1990.

[12] T. Karatas and F. Bullo. Randomized searches and nonlinear programming in trajectory planning. In *IEEE Conference on Decision and Control*, 2001.

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

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

[15] S. M. LaValle and J. Kuffner Jr. Randomized kinodynamic planning. In *IEEE Int. Conf. Robot. & Autom.*, 1999.

[16] S. M. LaValle and P. Konkimalla. Algorithms for computing numerical optimal feedback motion strategies. *Int. J. Robot. Res.* To appear.

[17] K. M. Lynch and M. T. Mason. Stable pushing: Mechanics, controllability, and planning. *Int. J. Robot. Res.*, 15(6):533–556, 1996.

[18] J. Reif and H. Wang. Non-uniform discretization approximations for kinodynamic motion planning. In J.-P. Laumond and M. Overmars, editors, *Algorithms for Robotic Motion and Manipulation*, pages 97–112. A K Peters, Wellesley, MA, 1997.

[19] 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.

[20] C. Yap and T. Dubé. *The exact computation paradigm.* World Scientific Press, 1995.