Reachability-Guided Sampling for Planning Under Differential Constraints

Alexander Shkolnik, Matthew Walter, and Russ Tedrake

***Abstract*— Rapidly-exploring Random Trees (RRTs) are widely used to solve large planning problems where the scope prohibits the feasibility of deterministic solvers, but the efficiency of these algorithms can be severely compromised in the presence of certain kinodynamics constraints. Obstacle fields with tunnels, or tubes are notoriously difficult, as are systems with differential constraints, because the tree grows inefficiently at the boundaries. Here we present a new sampling strategy for the RRT algorithm, based on an estimated feasibility set, which affords a dramatic improvement in performance in these severely constrained systems. We demonstrate the algorithm with a detailed look at the expansion of an RRT in a swingup task, and on path planning for a nonholonomic car.**

I. Introduction

The problem of robotic motion planning, whether for a wheeled robot, an aerial vehicle, or a quadruped, has motivated the development of powerful tools for planning in high dimensional spaces. Of these techniques, randomized sample-based planning strategies have proven particularly effective in quickly solving for a series of actions that drive the system to a desired state. Perhaps the most widely used single-query sample-based planners are those based upon Rapidly-exploring Randomized Trees (RRTs). RRTs have been used to solve a broad range of planning problems that include motion for manipulators and kinematic chains [1], [2].

It is well-known that these algorithms can quickly become inefficient when planning on systems with complicated kinodynamic constraints [3], and there has been considerable work in attempting to modify the basic RRT algorithm for these situations [3], [4]. The essential symptom of this inefficiency is that nodes at the boundaries of these constraints tend to be sampled and expanded repeatedly, with little progress towards the ultimate goal.

In this paper, we present a modified adaptive sampling strategy for the RRT algorithm which takes into account local reachability, as defined by differential constraints, while building the tree. The algorithm is based on the observation that sampling points at random and checking collisions is relatively cheap, but adding extra nodes to the tree (through the Extend algorithm) is relatively more expensive - both in its instantaneous cost and in the added cost of having a larger tree. Therefore, we attempt to build sparse trees through kinodynamic obstacles, with a simple heuristic that quickly throws away random samples that otherwise would not have the effect of extending the tree into previously unexplored regions of state space. This effectively changes the sampling distribution, so that points are selected uniformly from a potentially small portion of the configuration (or state) space in which the tree is capable of growing. The result of this simple algorithm is a dramatic practical improvement in the total computation time involved with finding a solution in a constrained search problem.

The authors are in the Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology, 32 Vassar St. Room 32-380, Cambridge, MA 02139, USA {shkolnik, mwalter, russt}@mit.edu

In order to illustrate the mechanisms involved with the new sampling strategy, we focus our attention in this paper on a low-dimensional torque-limited pendulum swing-up example, where the Voronoi regions can be illustrated and the entire tree can be visualized effectively. To illustrate the generality of the approach, we also discuss the planning for a nonholonomic vehicle through a tight corridor. In both cases, the algorithm makes a substantial enough improvement in planning speed to consider real-time planning. Finally, we discuss the applicability of the algorithm to high dimensional spaces.

II. RRT Planning Under Differential Constraints

In this section, we analyze the performance of the RRT in the presence of kinodynamic constraints, with an emphasis on planning for underactuated systems. Before proceeding with the analysis, we first review the basic operation of the general RRT algorithm.

A. Basic RRT Operation

The fundamental process by which the vanilla RRT operates is relatively simple. Given an initial state, $x_{\text{init}} \in \mathcal{X}$ and a goal state, $x_{\text{goal}} \in \mathcal{X}$, the planner incrementally builds a set of trajectories through the state space, $\mathcal{X}$, in search of a path that connects the initial state with the goal. These trajectories are modeled as a set, $\mathcal{T}$, of inter-connected points that form a tree-like structure over a region of the state space. At each iteration of the algorithm, one draws a random sample, x_{rand}, from the state space according to a predefined sampling distribution, $p_{\text{rand}}(x)$. The algorithm then identifies the node in the tree that is closest to the sample, x_{near}, as defined by the specified distance metric, $\rho(x)$. This point is expanded towards the random sample based upon the best known input that drives the state towards the sample and the resulting state, x_{new}, is added to the tree.

The pattern by which nodes are selected for expansion is a function of both the sampling distribution and the nearest-neighbor distance metric. Typically, samples are drawn uniformly over the state space while the most common metric for the nearest-neighbor selection is the Euclidean distance between points. In this case, the expansion pattern of the tree is modeled by the Voronoi diagram over the nodes within the tree. The probability of a node being expanded is directly proportional to the volume of its corresponding Voronoi region. Nodes that have a larger Voronoi region are

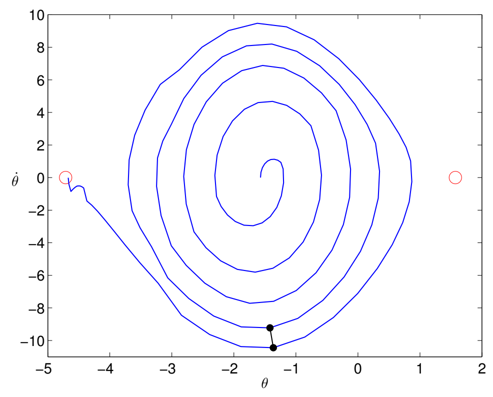


Fig. 1. Example state space trajectory for the torque-limited, single link pendulum. Due to the torque limits at the shoulder, a swing up phase is necessary to drive the pendulum from the vertical downward position to the stationary upright position. The two black dots connected by a segment shows that the Euclidean metric is not a good measure of cost-to-go

more likely to be chosen for expansion and are referred to as *major* nodes. In the case of the Euclidean metric, these nodes tend to lie on the outside of the tree during the initial exploration. Conversely, *inner* or *minor* nodes have smaller Voronoi regions and often lie on the inside of the tree. Once the tree has explored the state space, these become major nodes as the algorithm begins to fill in the space. This phenomenon of favoring some nodes over others is referred to as the *Voronoi bias*, and yields an initial preference towards the exploration of the state space. Over time, the Voronoi regions become more uniform in volume and, in the case of planning for holonomic systems, the likelihood of expanding a node tends toward the sampling distribution [5].

B. Performance with Kinodynamic Constraints

The efficiency by which the RRT algorithm is able to grow the tree and explore the space is highly sensitive to the distance metric. In the presence of kinematic (eg, joint limits, obstacles, or non-holonomic constraints) or dynamic (eg, torque limits, or underactuation constraints), widely-used metrics like the Euclidean distance are a very poor representation for the true distance between points in the space. As a result, the algorithm often repeatedly attempts to extend the same nodes without growing the tree any closer to the sampled region.

Consider the swing-up task for a torque-limited pendulum. The two-dimensional state space consists of the joint angle and the angular rate, $x = \begin{bmatrix}\theta, & \dot{\theta}\end{bmatrix}^\top$. The task is to bring the pendulum from a stationary down position, $x_{\text{init}} = \begin{bmatrix}-\pi/2, & 0\end{bmatrix}^\top$, to the fixed upright goal pose, $x_{\text{goal}} = \begin{bmatrix}\pi/2, & 0\end{bmatrix}^\top$. If the torque-limits are severe, then a swing up phase is required in order to drive the system to most states in the state space, including the goal pose. The spiral state trajectories in Figure 1 demonstrate this behavior as the pendulum is swung back and forth to reach the goal.

The Euclidean distance metric is ignorant to the fact that the differential constraints on the pendulum dynamics require a swing up phase to reach most of the state space. Consider the pair of points connected with a black line, shown in Figure 1. The Euclidean distance metric would suggest that this pair is very close to each other, but the pendulum is unable to transition between these two parts of state space without swing up. While RRT-based planners have successfully solved swing up control for the underactuated pendulum [6] using the Euclidean metric, it comes at a cost of expanding a large number of fruitless nodes. Often times, this is tolerable for low-dimensional systems like the single-link pendulum, but not so for higher-dimensional systems for which sample-based planners are typically well-suited.

C. RRT Modifications

The motion planning literature contains a number of attempts to improve the performance in constrained systems. An obvious solution is to use a metric other than the Euclidean distance that is better-suited to the problem. Perhaps an ideal metric would be the optimal cost-to-go in terms of the time or energy required to drive the system between a pair of states [7]. Unfortunately, computing the ideal cost is intractable for most higher-order systems since it is equivalent to optimally solving the original motion planning problem [2]. Instead, RRT-based planners often compromise and utilize a sub-optimal heuristic function that has been tuned to the particular problem as the distance measure. The metric is not as accurate as the optimal cost-to-go, but typically performs much better than the Euclidean distance. Unfortunately, these metrics rely upon domain-specific knowledge and are not generalizable across different systems.

Rather than design a system specific metric for planning, another option is to adaptively learn a suitable metric during planning. Such is the approach of Cheng [3, Ch. 6], who scores nodes according to their consistency with constraints as well as that of their children in the tree. These values are updated while building the tree and used to select nodes during exploration. The RRT extension proposed in [8] is similar, in which the algorithm keeps the history of expansion attempts on nodes, considering failed attempts as both collisions, as well as repetition of previous expansions. Furthermore, in place of the Euclidean distance metric, this algorithm calculates the cost-to-go from each node in the tree toward the sample using a linear approximation of the dynamics for each node. The approach alleviates some of the problems with the Euclidean distance metric, but can be computationally expensive to compute, and is sensitive to how well the linearization approximates the actual cost-to-go.

The dynamic-domain RRT [9] combats futile oversampling by altering the size of the Voronoi regions for those nodes that are close to obstacles. By lowering the bias associated with these nodes, the dynamic-domain RRT reduces the likelihood of drawing samples from regions of the state space that are more likely to induce collisions. More specifically, the algorithm identifies boundary nodes as those for which a collision occurred while expanding a sample. The domain of samples that can then be matched with a boundary node is then restricted to a sphere of fixed radius in the state space. One limitation of the dynamic-domain RRT is that it must first encounter collisions before modifying the Voronoi bias. Additionally, it currently supports only spherical domains centered about the boundary nodes, which do not necessarily reflect the best sampling regions.

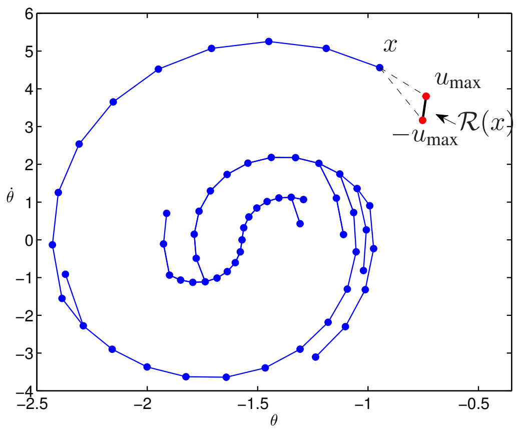


Fig. 2. The reachable set, $\mathcal{R}(x)$, for the underactuated pendulum consists of an approximately linear segment, the endpoints of which are defined by the points in state space that are achieved by applying the minimum and maximum torques at the shoulder.

Other modifications to the RRT assume a discretized set of actions. In the RRT-Blossom algorithm [10], whenever a node is chosen for expansion, all possible (discrete) actions are attempted, while sparse expansion is encouraged by weeding out any expansions of the tree if the expanded node is closer to other nodes in the tree other than the parent. Once a node is visited once, it never needs to be visited again, so redundant exploration is a non-issue. However, in some problems (for example the pendulum) it does not make sense to remove child nodes in this way since child nodes will be near other nodes in the tree, unless the integration time step Δt is very small, which will increase the size of the resulting tree exponentially. Another variant that assumes discrete actions, but was only demonstrated on path planning problems without differential constraints, is the Rapidly-Exploring Random Leafy Tree (RRLT) [11]. Leaf nodes, corresponding to nodes that are reachable in one time step from the current tree are stored in the tree. The algorithm picks the closest leaf node to a random sample, converts it to a main node in tree, and then generates new reachable leaf nodes from this new node.

III. Reachability-Guided RRT

In the previous section, we presented an analysis of the shortcomings of existing RRT-based planners in dealing with systems with differential constraints. In particular, we emphasized the sensitivity of RRTs to the nearest neighbor metric and the implications that this has for building the tree.

We now present the Reachability-Guided Rapidly-exploring Random Tree (RG-RRT) as an alternative planning strategy for general systems that are subject to differential constraints. The RG-RRT takes the form of a modified RRT that explicitly accounts for the limitations of the system dynamics to shape the Voronoi bias so as to emphasize nodes within the tree that exhibit the greatest contribution towards exploring the state space. The RG-RRT alleviates the sensitivity to the distance metric and, in turn, does not require a system-specific metric heuristic. The result is an expansion of the tree that makes efficient use of the system dynamics to more rapidly reach the goal.

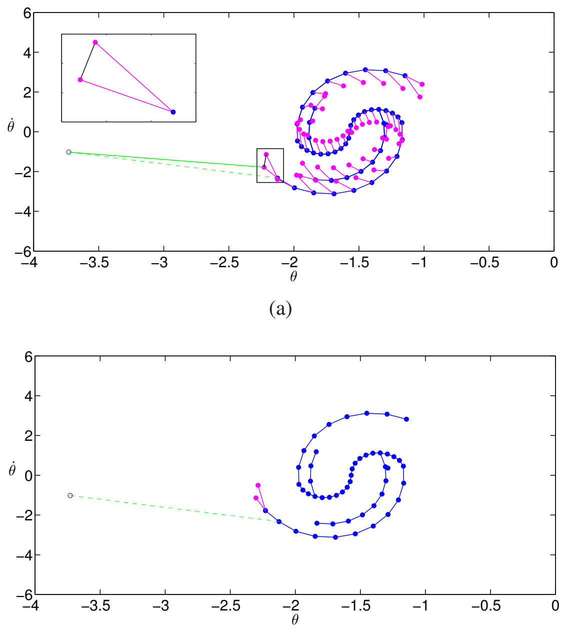


Fig. 3. Two consecutive steps in a single iteration of the RG-RRT, as demonstrated for the underactuated pendulum. In (a), a random sample is drawn from the state space and paired with the nearest node in the tree and the closest point in its reachable set according to the Euclidean distance metric. Shown in (b), the algorithm then expands the node towards the point in the reachable set and adds that point to the tree.

Additionally, we strictly ensure that any nodes added to the tree must make progress towards a given sample. A primary idea here is that for a given tree, particularly in a system subject to dynamic constraints, it is often not able to expand into certain regions of state space. We implement what is effectively an adaptive sampling strategy, which constantly changes the allowed regions of samples for which expansions are attempted, based on the local capabilities of expansion of the current tree.

A. Problem Formulation

We consider the problem of path planning for a system subject to differential constraints. Let $\mathcal{X}$ denote the state space that applies to kinodynamic systems and let $\mathcal{C}$ represent the general configuration space. We are concerned with finding a time-varying control input, $u(t) \in \mathcal{U}$ for $t \in [0, T]$ that drives the system from some initial state, x_{init}, to an arbitrary goal state, x_{goal}, in finite time, T. The resulting trajectory, $x(t)$, must be free of collisions, i.e. $x(t) \in \mathcal{X}_{\text{free}}$ where $\mathcal{X}_{\text{free}}$ denotes free space, and satisfy the differential constraints according to the state transition function,

$$\dot{x} = f(x, u). \quad (1)$$

The problem becomes even more challenging when considering systems that are additionally constrained to be underactuated or nonholonomic.

B. The RG-RRT Algorithm

The Reachability-Guided RRT is derived from the premise that sampling is relatively inexpensive compared to the process of searching for collision-free trajectories from a node while satisfying differential constraints. The

RG-RRT provides a means for choosing nodes that have a better chance of yielding consistent trajectories without having to redefine the metric function. Instead, the RG-RRT actively constrains the set of nodes under consideration for nearest-neighbor pairing with a sample to those that are actually able to expand towards the given sample. Effectively, it applies the Euclidean metric to nodes for which it is a valid indication of the approximate cost-to-go under the differential constraints. The RG-RRT does so by considering the reachable region of the state space associated with each node.

Definition 1: For a state $x_0 \in \mathcal{X}$ and a finite local integration time step Δt, we define its *reachable set*, $\mathcal{R}_{\Delta t}(x_0)$, to be the set of all points that can be achieved from x_0 in finite time, Δt, according to the state equations (1) and the set of available control inputs, $\mathcal{U}$.

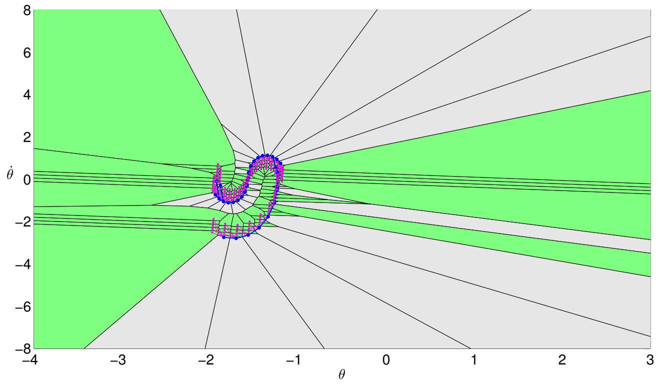
In the case of the torque-limited, single-link pendulum, the reachable set for a state is an approximately linear segment, the bounds of which are found by integrating the dynamics while applying the maximum negative and positive torque, $-u_{\text{max}}$ and u_{max}. Figure 2 depicts the reachable set for a node in the tree. Any state along the segment can be achieved in time Δt by an allowable control input, $|u| \leq u_{\text{max}}$.

Algorithm 1 $\mathcal{T} \leftarrow$ BUILDRRT(x_{init})

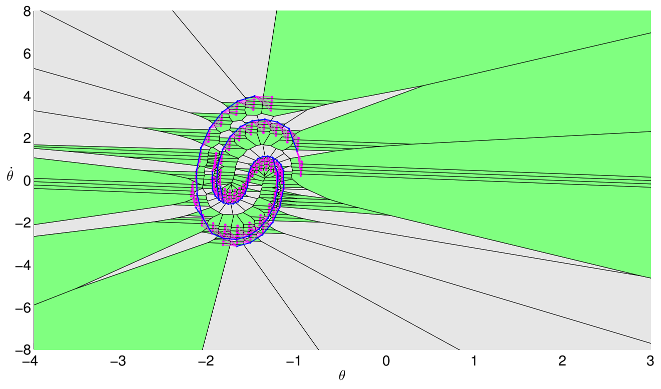
```
1:  $\mathcal{T} \leftarrow$ INITIALIZETREE();
2:  $\mathcal{T} \leftarrow$ INSERTNODE($x_{\text{init}}, \mathcal{T}$);
3:  for $k = 1$ to $k = K$ do
4:      $x_{\text{rand}} \leftarrow$ RANDOMSTATE();
5:      $(x_{\text{near}}, x^r_{\text{near}}) \leftarrow$ NEARESTSTATE($x_{\text{rand}}, \mathcal{T}$);
6:      while $x_{\text{near}} = \{\}$ do
7:          $x_{\text{rand}} \leftarrow$ RANDOMSTATE();
8:          $(x_{\text{near}}, x^r_{\text{near}}) \leftarrow$ NEARESTSTATE($x_{\text{rand}}, \mathcal{T}$);
9:      end while
10:     $u \leftarrow$ SOLVEINPUT($x_{\text{near}}, x^r_{\text{near}}, x_{\text{rand}}, \mathcal{T}$);
11:     $x_{\text{new}} \leftarrow$ NEWSTATE($x_{\text{near}}, u$);
12:     $\mathcal{T} \leftarrow$ INSERTNODE($x_{\text{new}}, \mathcal{T}$);
13: end for
14: return $\mathcal{T}$
```

The RG-RRT algorithm builds and maintains a standard RRT, $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, consisting of a set of vertices, $\{x_i\}$ connected by edges. Associated with each node is a representation of its reachable set, $x_i.\mathcal{R}$. At a high level, the RG-RRT algorithm possesses much the same structure as standard RRT-based planners. The primary differences lie in the use of a node's reachable set to focus sampling on regions of the state space that are most likely to promote expansion under the differential constraints.

The structure of the RG-RRT algorithm is outlined in Algorithm 1. Given an initial point in state space, x_{init}, the first step is to add the state as a node in the tree. Given the point x_{init}, the INSERTNODE() function solves for the set of reachable points in the state space that are consistent with the differential constraints (1). For many systems of interest, the approximate bounds of the reachable set can be generated by discretizing the action space, and then integrating the corresponding dynamics forward using any favored integration method. In some cases, with a short enough Δt, taking actions between the limits results in an integrated state that lies between the states produced when taking the actions at their limits. In such cases it is sufficient to consider only the bounds of the action set. The reachable set may also be approximated or learned. When the dimension of the action space increases, in particular, it may become more efficient to approximate the reachable set with a simple geometric function, for example an ellipsoid, if such a function can approximate the actual reachable volume. One benefit of generating the reachable set using discretized actions is that points that comprise the reachable set can also be efficiently tested for collisions before they are added to the Reachable set, in order to reduce the likelihood of trajectories leaving free space as part of the exploration phase.



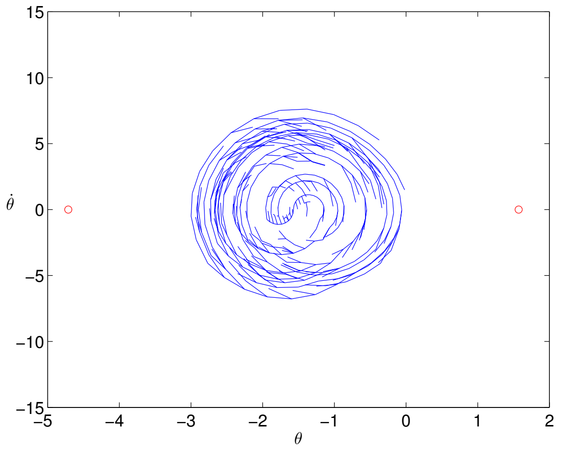
(a) 30 nodes



(b) 60 nodes

Fig. 4. RG-RRT Voronoi diagrams for a pendulum. The blue diagram in (a) corresponds to the tree after 30 nodes have been added while that in (b) corresponds to the tree with 60 nodes. Magenta dots are discretely sampled reachable points affiliated with the tree. Green regions are areas where samples are 'allowed', and correspond to Voronoi areas associated with the reachable set. Samples that fall in any grey areas are discarded. Note that these regions are constantly changing as the tree grows.

With the node and its corresponding reachable set added to the tree, we draw a random sample, x_{rand}, from the state space, typically according to a uniform distribution, and use it to grow the tree. We do so through a variation on the nearest-neighbor matching strategy, that restricts its domain to nodes that are more likely to promote the expansion of the state space. More specifically, the NEARESTSTATE($x_{\text{rand}}, \mathcal{T}$) function compares the distance from the random sample not only to the nodes, but also to the points within their reachable


(a) Standard RRT

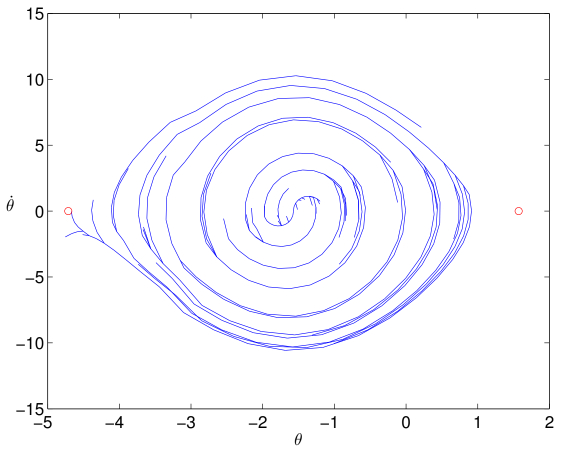

(b) RG-RRT

Fig. 5. The trees for the underactuated pendulum after adding 360 nodes for (a) the standard RRT-based kinodynamic planner and (b) the RG-RRT planner. While the RG-RRT algorithm has reached the goal state, the RRT-based planner has yet to explore much of the state space. The RG-RRT converged in 3 seconds.

sets. If the closest Reachable point is closer to the sample than the closest node of the tree, then both this reachable point, x^r_{near}, and its corresponding parent node, x_{near}, are returned. Otherwise, if the closest node of the tree is nearer to the sample than any Reachable point, the function returns an empty point pair, in which case the RG-RRT throws this sample away, and draws a new sample from the state space and repeats the process.

Figure 3 demonstrates this process for the pendulum example. The figure shows a tree grown from the initial state corresponding to the pendulum down with zero rotational velocity. The nodes and edges comprising the tree are in blue while the magenta points that emanate from each node correspond to the bounds of its reachable set. In Figure 3(a), we have drawn a sample from the state space and computed the Euclidean distance to each of their nodes and the two points that bound each of their reachable sets. Of these nodes and their reachable set, the NEARESTSTATE$(x_{\text{rand}}, \mathcal{T})$ function identifies one of the two extreme points (indicated by the solid green line) within the reachable set of a leaf node (dashed green line) as a suitable point for expansion.

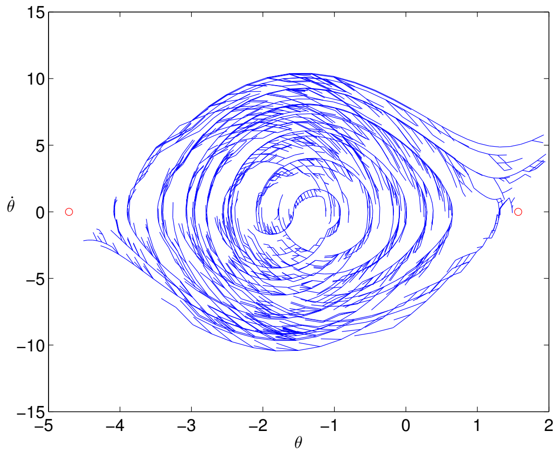


Fig. 6. The final tree generated with the standard kinodynamic RRT-based planner. The tree consists of over 2300 nodes and required 75 seconds to generate.

The result of this policy of throwing out samples for which the nearest node is closer than its reachable set is a change in the Voronoi bias. The RG-RRT then allows samples only from Voronoi regions for which the differential constraints permit the expansion of the node towards the sample. While samples may be drawn from anywhere within the state space, only a subset of regions are actually used to grow the tree. This has the effect of modifying the Voronoi bias to emphasize nodes that are better suited to explore the state space while growing the tree. Figure 4 depicts this phenomenon for the pendulum. The plot presents the Voronoi regions for each of the nodes within the tree and identifies in green the regions for which drawn samples are deemed as valid. Many of the large Voronoi regions associated with outer nodes have appropriately been identified as not suitable for expansion, since it is impossible to grow the tree into these regions in time Δt under the differential constraints. The nodes may still be expanded upon, but towards samples drawn towards the inside of the tree. These nodes, which would otherwise serve as major nodes with standard RRT-based planners using the Euclidean distance, are instead treated as minor nodes.

Upon identifying a suitable node for expansion, the RG-RRT extends the tree from the node. The SOLVEINPUT$(x_{\text{near}}, x^r_{\text{near}}, x_{\text{rand}}, \mathcal{T})$ function can either search for a consistent action that drives the system from the state towards the sample or returns the control associated with the reachable point, x^r_{near}. In either case, lines 10 and 11 in Algorithm 1 identify a collision-free trajectory to a state within $\mathcal{R}(x_{\text{near}})$ that obeys the system's differential constraints. Figure 3(b) demonstrates this step where we have extended the node to the point, $x_{\text{new}} = x^r_{\text{near}}$, in its reachable set that was the nearest to the sample. The new point, x_{new}, is added as a node to the tree together with its reachable set, $\mathcal{R}(x_{\text{new}})$, as before. The RG-RRT then continues with the next iteration of the algorithm.

IV. RESULTS

We implemented the RG-RRT to perform kinodynamic planning for two different systems in simulation. The first application is to find swing up trajectories for the single-link, torque-limited pendulum. We then applied the algorithm to solve for collision-free trajectories for a simple nonholonomic car driving amongst obstacles. For comparison, we also applied the standard RRT-based kinodynamic planner within both simulations. The two planners used a uniform distribution over the state space to generate samples and the Euclidean metric to identify nodes for expansion.

A. *Swing Up Control for an Underactuated Pendulum*

We first applied the RG-RRT to solve for a swing up controller for the underactuated pendulum. The control authority at the shoulder was limited in magnitude. The system parameters were set to a mass of $m = 1$, a length of $l = 0.5$, a damping coefficient of $b = 0.1$, gravitational constant of 9.8, and a maximum torque of $|u| \leq 0.1$.

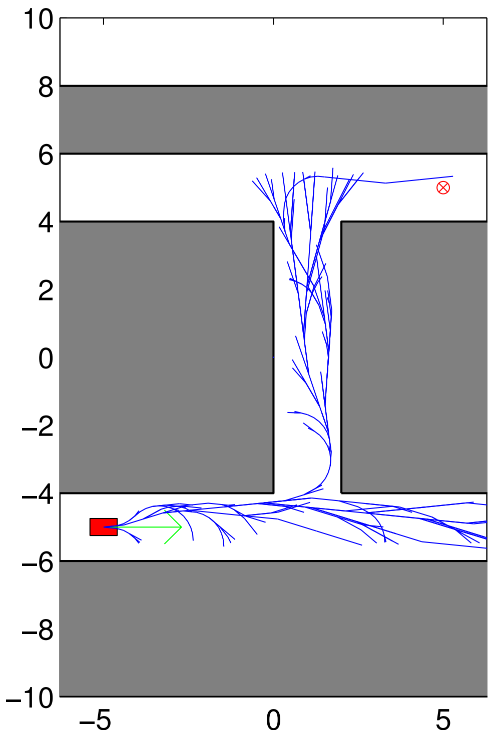
Figure 5 compares the trees for the standard RRT and the RG-RRT after 360 nodes have been expanded. At this point, the RG-RRT algorithm has identified a path to the goal that is consistent with the differential constraints. In contrast, the planner based upon the standard RRT has yet to explore much of the state space. Instead, we see the aforementioned sensitivity to the Euclidean metric, which has resulted in the expansion of many major (outer) nodes for which the torque limits allow the tree to only grow inwards. This is evident in the large number of overlapping branches that extend inward rather than extending towards the larger Voronoi regions. Referring back to the Voronoi diagrams in Figure 4(a), note that the only valid Voronoi regions for these nodes would be on the inside of the tree where sampling is consistent with the ability to grow the tree inwards.

The RG-RRT was able to solve for a swing up controller for the pendulum after expanding the tree 360 times. The overall search took 3 seconds in MATLAB. Meanwhile, the standard RRT-based planner converged to the goal after adding 2300 nodes to the tree, a process that required 75 seconds to complete. All simulations in this paper were run on a 3ghz Intel Xeon X5450 CPU. Figure 6 presents the final tree generated by the standard RRT.

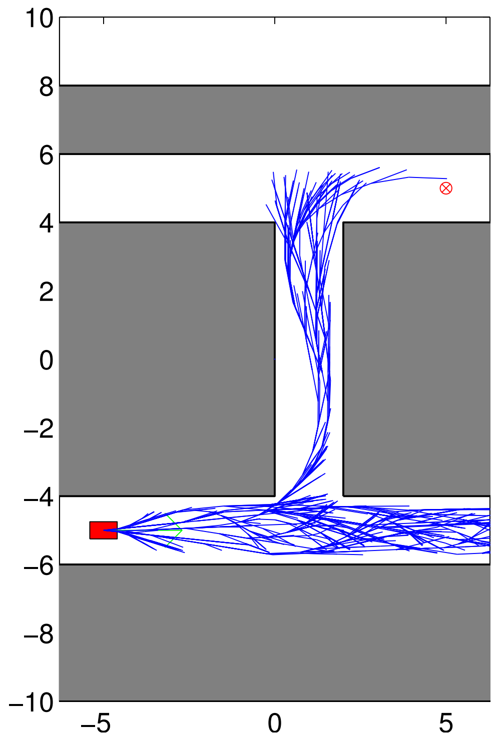
While it's not obvious from the tree, the RG-RRT results in an approximately bang-bang trajectory for swing up control. This behavior is more evident in Figure 3(a) where each node corresponds to one of the extreme points, i.e. $|u| = u_{\max}$, that bound the reachable set for its parent node. This control policy agrees with the bang-bang control strategies that have been proposed elsewhere for the inverted pendulum [12].

B. *Motion Planning for the Acrobot*

For comparison in a similar problem with a higher dimensional search space, we implemented a modified bidirectional version of the RG-RRT algorithm, and applied it on a simulated torque-limited Acrobot [13], a two link pendula with an actuator at the elbow, and a passive joint at the base. In our simulation, each link weighed 2kg and was .5m in length with the COM in the center of the link. The reachable set was approximated by discretizing actions into three possible torques values. A standard bidirectional RRT was also run for comparison using the same code base. We found that the standard RRT worked fastest when the control was sampled from a uniform distribution between the torque limits. The bi-directional RG-RRT was run 10 times and took



(a) RG-RRT



(b) Standard RRT

Fig. 7. A comparison of the final collision-free trajectories that lead the simple car model from its initial position in the lower left-hand corner of the environment to the goal at the upper right. The final tree for (a) the RG-RRT consists of 292 nodes and was generated in under 3 seconds. In contrast, the tree built with the standard RRT-based planner includes 1350 nodes and required 43,000 expansion attempts and 73 seconds to compute.

an average of 38 seconds to run, compared to 112 seconds for the standard RRT.

C. Motion Planning for a Simple Nonholonomic Car

Another class of problem that we consider is a nonholonomic car navigating through a set of corridors bounded by obstacles as shown in Figure 7. We represent the state of the vehicle by its position, heading, and forward velocity, i.e. (x, y, θ, v). The control inputs to the system consist of the angular rate and the forward acceleration, $u = \left[\dot{\theta},\ \dot{v}\right]$, both of which are bounded in magnitude. For our simulation, the vehicle is restricted to forward motion. The task is to find a collision-free trajectory that brings the rectangular-shaped vehicle through the narrow corridors from the lower-left to the upper-right corners of the environment, both with an Eastward heading. The setup of this problem is challenging because the only feasible paths require the vehicle to substantially slow down and make a wide turn in order to avoid collision around the tight turns.

One can visualize the state of the system in two dimensions where the location of a point denotes the vehicle's (x, y) position. The direction of a vector extending from the point represents the heading, θ, while the length of the vector denotes the forward velocity, v. In this space, the reachable set, $\mathcal{R}$, is bound by a quadrilateral positioned in front (per the orientation vector) of the point in space where the four corners correspond to the four pairs of saturated control inputs. Any point within this quadrilateral is reachable in time Δt according to the differential constraints.

We applied both the RG-RRT as well as the standard RRT planner to solve for a sequence of control inputs that drive the vehicle to the goal pose while satisfying the differential constraints and avoiding obstacles. Both algorithms utilized the Euclidean distance metric, and shared much of the same code base. Figure 7 compares examples of trees resulting from both algorithms. Each was run a total of 20 times, with the same start and goal poses and obstacle configuration as shown in the figure. On average, the RG-RRT found a path in 2.3 seconds, with a mean of 405 nodes in the tree, and a mean total of 2150 integrations. Five integrations were performed for each expansion, including four required to generate a reachable set for each new node. On the other hand, the standard RRT required 51 seconds on average, with a mean of 1700 nodes in the tree, and 35,000 integrations required. The large number of integrations resulted from a substantial proportion of expansion attempts that failed due to collision.

V. Discussion

The RG-RRT algorithm alleviates the sensitivity of randomized sampling for systems with differential constraints to the metric that is employed to expand the tree. Essentially, the RG-RRT utilizes the metric only for regions of the state space for which it is valid. The result is a modified Voronoi bias that emphasizes nodes that are more likely to promote exploration given the constraints on the system dynamics. The RG-RRT takes advantage of the fact that sampling is cheap compared to the process of expanding a node and, therefore, is willing to resample until a point is drawn from a region that yields better exploration.

The size distribution for the modified Voronoi regions under the RG-RRT varies as the tree explores the state space. Initially, there is a large variability in the size of the different Voronoi regions but, as the tree expands, the size tends to become more uniform. The lower bound on the size of the regions, typically corresponding to the inside of the tree, is inversely proportional to the resolution of the reachable set. In turn, the sampling of the space will be probabilistically complete so long as the resolution of the reachable set is sufficient relative to the smallest "gap" in the state space.

As we investigate higher dimensional problems, if the reachable set cannot be concisely parameterized, then we must employ a sampling strategy to describe the set. This has the drawback that the sampling will grow exponentially on the dimension of the action space (not the state space). Clever parameterizations may be possible, but we have not addressed this challenge yet.

Overall, the algorithm presented is a clean way of eliminating metric and sampling specificity in the RRT algorithm implementation, which is particularly useful when there are differential constraints, or when reachable sets are otherwise non-spherical. The algorithm has shown significant time improvements in planning on two different example problems.

Acknowledgment

This work was supported by the DARPA Learning Locomotion program (AFRL contract # FA8650-05-C-7262)

References

[1] J. Cortés and T. Siméon, "Sampling-based motion planning under kinematic loop-closure constraints," in *Proceedings of the 6th International Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2004, pp. 59–74.

[2] S. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[3] P. Cheng, "Sampling-based motion planning with differential constraints," Ph.D. dissertation, University of Illinois, Urbana-Champaign, Urbana, IL, August 2005.

[4] B. Burns and B. O., "Utility-guided random trees," Computer Science Department, University of Massachusetts Amherst, Tech. Rep. 06-29, June 2006.

[5] J. Kuffner, J.J. and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, San Francisco, CA, April 2000, pp. 995–1001.

[6] M. Branicky and M. Curtiss, "Nonlinear and hybrid control via RRTs," in *Proceedings of the International Symposium on Mathematical Theory of Networks and Systems*, South Bend, IN, August 2002.

[7] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-Time Motion Planning for Agile Autonomous Vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 116–129, JanuaryFebruary 2002.

[8] J. Kim, J. M. Esposito, and V. Kumar, "An rrt-based algorithm for testing and validating multi-robot controllers," in *Robotics: Science and Systems I*. Robotics: Science and Systems, June 2005.

[9] A. Yershova, L. Jaillet, T. Siméon, and S. M. LaValle, "Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, April 2005, pp. 3856–3861.

[10] M. Kalisiak, "Toward More Efficient Motion Panning with Differential Constraints," Ph.D. dissertation, University of Toronto, 2008.

[11] Morgan, S., Branicky, and M.S., "Sampling-based planning for discrete spaces," *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 2, pp. 1938–1945 vol.2, Sept.-2 Oct. 2004.

[12] K. Furuta, M. Yamakita, and S. Kobayashi, "Swing up control of inverted pendulum," in *Proceedings of the International Conference on Industrial Electronics, Control, and Instrumentation*, vol. 3, Kobe, Japan, October 1991, pp. 2193–2198.

[13] M. W. Spong, "Swing up control of the acrobot," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1994, pp. 2356–2361.