Preemptive Rapidly-Exploring Random Trees

Zachary Weiss
Boston University
1 Silber Way, Boston, MA 02215, United States
zacharyw@bu.edu

Abstract

Rapidly exploring random trees (RRTs) are a commonly employed method of efficiently searching non-convex and/or high-dimensional spaces. Within the context of robot motion planning, RRTs can handle problems with additional constraints, such as degrees of freedom, kinematic bounds, and obstacles. In all algorithms intended for real-time use in autonomous robots, computation time is an important consideration, and often planning must occur within a predefined time budget, even if at the cost of solution optimality. In doing so, however, one wishes to still provide the most optimal solution discoverable within the time budget; this paper sets out to consider the implementation of a preemptive iteration limited RRT algorithm, taking inspiration from the preemption approach of Preemptive RANSAC.

1. Introduction

1.1. Prior Work

1.1.1 Rapidly-Exploring Random Trees

LaValle's original implementation of the RRT algorithm extended existing progress in randomized approaches to nonholonomic planning problems [4]. It considers path planning as a search in a metric space, X, for a continuous path between an initial state x_i to a final state x_{goal} or goal region X_{goal} within X: in standard problems the state space X corresponds to the configuration space C, in kinodynamic problems this changes to the tangent bundle of the configuration space, detailed in greater depth in LaValle's 2001 follow-on paper with Kuffner [5] as well as in Suh *et al.*'s 2011 paper on Tangent Space RRT [9].

RRTs are constructed by beginning at x_{init}, and then for each vertex up to a defined count of vertices K, a random state within the configuration space is selected (rejecting illegal samples), and the nearest node to the random configuration is found. Here, a new node is created a set incremental distance towards the random configuration from the current node, and the two nodes are connected by an edge. These steps iterate until it meets the total number of nodes allotted. This algorithm is formally described below in the Methods section, including pseudocode in the block for Algorithm 1, taking input arguments x_i for graph's root, K for the number of vertices, and Δt for the incremental distance from the nearest node towards the new random state. It returns an RRT: $\mathcal{T}$.

This entirely stochastic process allows it to search nearly any space with reasonable efficiency, and without concern of entrapment in local minima, tricky constraints upon the system dynamics, or dimensionality of the space.

1.1.2 RRT Variants

Because of the wide array of aforementioned benefits and potential applications of RRTs, several variants have been created since the technique's inception. These notably include variants that improve the processing and memory impacts, create convergence to optimal solutions, improve upon convergence rates with heuristics, and live-rewire trees for real-time implementation: examples being RRT*[8], A*-RRT*[1], Informed RRT*[3], and RT-RRT*[6]. While anytime algorithms enable the use of RRT-variant approaches in real-time applications, this can come at the cost of optimality.

1.1.3 Preemption, as applied to RANSAC

This difficulty, of solution optimality in a real-time situation, is similarly faced with RANSAC [2]. Standing for random sample consensus, RANSAC is an iterative approach to model parameter estimation from data containing outliers; it is commonly applied to computer vision problems within the context of correspondence-based methods: where one assumes correspondences between points, and outliers must be weeded out to generate good hypotheses. The primary disadvantage of RANSAC's iterative design is that there does not exist any upper bound to the computation time, and as such, when limits are externally enforced, results can be far from optimal. Nistér, in his 2005 paper [7], attempts to handle this by employing a preemption scheme

in which observations and hypotheses are iteratively scored against one another. As iterations pass, the number of hypotheses kept decreases, until only one remains: a singular preferred hypothesis. The primary appeal of this approach over others considered within the paper is that we can compare hypotheses against each other throughout the entire process, rather than simply against some unchanging measure of quality; the methods section covers the form of and theory behind the preemption algorithm in greater detail.

2. Methods

2.1. Overview

Seeking to use preemption as a novel approach to RRT optimality in real-time scenarios, the jumping-off point will be LaValle's original RRT algorithm, and a similar preemption scheme as proposed in the context of RANSAC, in which hypotheses are iteratively scored against observations, quitting with the best remaining hypothesis as the preferred one when the time / iteration budget is reached. As RRTs are not *per se* attempting to converge to a hypothesis, but rather simply seeking to fill space in a Monte-Carlo fashion, this concept will need to be modified somewhat. Borrowing concepts from the RRT variants mentioned, namely, the heuristics of Informed RRT*, we can make observations upon the properties of generated motion hypotheses, and comparatively score them in a preemptive manner. To demonstrate this in a proof of concept, a simple RRT will be generated to (cut off at) K nodes, at which point metrics over all paths will be used to hypothesize the best at time of termination.

2.2. RRT Implementation

The RRT algorithm was first implemented in MATLAB as per the specifications in LaValle's original paper, listed as pseudocode in Algorithm 1 below. This was then modified to implement collision checking with the environment when building x_{new}, and to gather and store the information for observations within the preemption component.

Algorithm 1 GENERATE_RRT($x_{init}, K, \Delta t$)

$\mathcal{T}$.init(x_{init})
for $i = 1$ **to** K **do**
  $x_{rand} \leftarrow$ RANDOM_STATE()
  $x_{near} \leftarrow$ NEAREST_NEIGHBOR($x_{rand}, \mathcal{T}$)
  $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near})
  $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$)
  $\mathcal{T}$.addVertex(x_{new})
  $\mathcal{T}$.addEdge(x_{near}, x_{new}, u)
end for
return $\mathcal{T}$

2.2.1 Observation Primitives

To gather basic metrics on each path, a set of observation "primitives" were gathered and computed on a node-by-node level. These consisted of the number of nodes in the path leading to the current node (self-inclusive), the path distance as calculated by the sum of the Euclidean norm of the edges, and the number of collisions along the path. Collision count was determined by incrementing a node's collision counter whenever an attempted child node failed: child nodes inherit parent node collision values when first created, and all children have their counter incremented if their parent's counter is incremented.

2.3. Preemption Design

2.3.1 Preliminaries and Implementation in RANSAC

A preemption scheme consists of multiple sub-functions. Given a set of observations, denoted as o_1 to o_N, and a set of hypotheses, denoted h_1 through h_M, we can use a scoring function $\rho(o_i, h_i)$ that takes the observation and hypothesis indices and returns a scalar value corresponding to the log-likelihood of the hypothesis in relation to the observation it was scored against. From this, the log-likelihood of a hypothesis at any stage along the preemption, and the overall log-likelihood, are respectively given by:

$$L_i(h) = \sum_{o=1}^{i} \rho(o, h) \tag{1}$$

$$L(h) = \sum_{o=1}^{N} \rho(o, h) \tag{2}$$

To apply this, an order rule and a preference rule are used to determine the order of the scoring sequence (which pairs of hypotheses and observations are to be scored next, according to prior scores) and to select the current preferred hypothesis, respectively. Altogether, this fully defines a preemption scheme. Depending on the scoring sequence order, a preemption scheme can be said to be depth-first, breadth-first, or hybrid (neither depth- nor breadth-first).

Within Preemptive RANSAC, a hybrid, though functionally breadth-first, scheme is employed. A decreasing preemption function $f(i)$ is defined that takes the iteration of the preemption scheme and returns the number of hypotheses to be kept for the current stage, discarding those with the lowest $L_i(h)$. The preemption function employed is:

$$f(i) = \left\lfloor M 2^{-\lfloor i/B \rfloor} \right\rfloor \tag{3}$$

With M denoting the number of hypotheses, and B denoting a block size, during which the number of hypotheses to keep will remain the same. Overall, the preemption takes the form of Algorithm 2, detailed below. This yields a pro-

Algorithm 2 RANSAC Preemption

1. Randomly permute the observations.
2. Generate the hypotheses indexed by $h = 1, ..., f(1)$.
3. Compute the scores $L_1(h) = \rho(1, h)$ for $h = 1, ..., f(i)$. Set $i = 2$.
4. Reorder the hypotheses so that the range $h = 1, ..., f(i)$ contains the best $f(i)$ remaining hypotheses according to $L_{i-1}(h)$.
5. If $i > N$ or $f(i) = 1$, quit with the best remaining hypothesis as the preferred one. Otherwise, compute the scores $L_i(h) = \rho(i, h) + L_{i-1}(h)$ for $h = 1, ..., f(i)$, increase i and go to Step 4.

cess that allows for comparison of hypotheses against one another throughout the entire preemption period, avoiding the use of solely some absolute quality measure. Additionally, the hybrid design avoids the pitfall of a solely breadth-first approach, which may take significant time before finding a decent hypothesis among the set. For these reasons, the preemption design is mirrored in this paper, the inputs to which the following sections describe.

2.3.2 Observations

To provide the preemption scheme with observations by which we may compare hypotheses, we take the metrics collected in the Observation Primitives section and compute multiple normalized statistics that describe a given node. While ultimately the factors we wish to minimize for optimality of path are the path length and distance to the goal, explicitly and/or solely selecting these would cause the favoring of paths that are more likely to be unsuccessful, especially in non-convex problems, where it may lead to entrapment in local minima. As such, the selected observations were a measure of "straightness", the collisions, the distance to the goal, and the node count. The straightness measure was inspired by the aforementioned Informed RRT* algorithm, which notes that the most optimal path in the absence of obstacles is a straight-line path between the start and the goal, and that when obstacles are included, this changes to the closest feasible path to the straight-line path, in sets of expanding ellipses. To compute a measure that would act as a proxy for this, the progress towards the goal for distance traveled was used, essentially the projection of the actual path onto the straight line, normalized by the path's length.

$$o_{i,str} = \frac{d_{start \to goal} - d_{node \to goal}}{d_{path}} \tag{4}$$

Collisions were similarly normalized by the total number of attempted expansions along the path to the current node, so as to not overly weight towards paths which simply have explored less of the environment. Goal distance and number of nodes received similar treatments.

$$o_{i,col} = \frac{n_{col}}{n_{col} + n_{nodes}} \tag{5}$$

$$o_{i,d_{goal}} = \frac{d_{node \to goal}}{d_{start \to goal}} \tag{6}$$

$$o_{i,n_{node}} = \frac{n_{node,i}}{\max n_{node,i}} \tag{7}$$

For ease of later use during scoring, each was remapped to the range of $[0, 1]$, and similarly had their directionality flipped to all align such that 0 is considered least likely to return a feasible and optimal path, and 1 is considered most likely, yielding:

$$o_{i,str} = \left(\frac{d_{start \to goal} - d_{node \to goal}}{d_{path}} + 1 \right) / 2 \tag{8}$$

$$o_{i,col} = 1 - \frac{n_{col}}{n_{col} + n_{nodes}} \tag{9}$$

$$o_{i,d_{goal}} = 1 - \frac{d_{node \to goal}}{d_{start \to goal}} \tag{10}$$

$$o_{i,n_{node}} = \frac{n_{node,i}}{\max n_{node,i}} \tag{11}$$

While a higher or lower value is not an assurance of the path to which it is attributed being better, it represents a prior belief that paths with those traits are more likely to succeed, and succeed optimally.

2.3.3 Hypothesis Generation

A hypothesis within the context of RRTs is the statement that a path will succeed and do so most optimally of the paths under consideration. Each leaf node in an RRT represents a unique motion path to that point from the starting position. As such, we build the RRT and gather all nodes without children, and similarly return the observation primitives to be turned into observations about each path as per the Observations section above. If the RRT is yet to reach the goal as one would expect within a scenario where preemption is required, we can consider solely leaf nodes, but if it can reach the goal within the limit, one would wish to either consider non-leaves in the preemption, or simply implement a check to see if any nodes reach the goal or goal region, and return the relevant path.

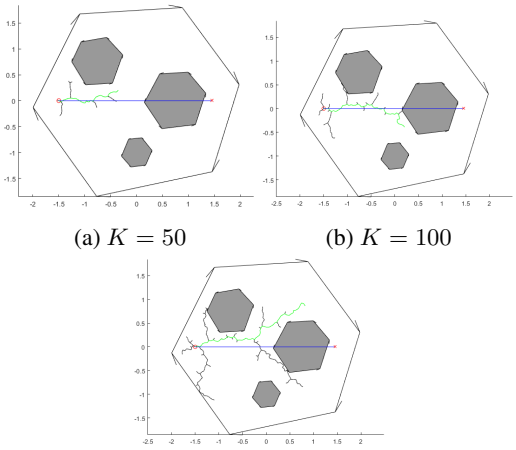
2.3.4 Scoring

To score the observations and hypotheses, we feed indices into a scoring function $\rho(o, h)$. For each hypothesis and observation pair, we find how much more or less likely the hypothesis is relative to the observation, based on their relative likelihoods within a probability distribution associated with each observation type. At present, this is modeled by the same beta distribution for all observation types; this is imperfect but functional for the scope of this work, with alternatives enumerated in the Future Work section below. Individual likelihoods by observation type are compounded into a single likelihood based on an array of weighting coefficients, enabling tuning of a given sub-observation's importance (or complete disregard of a given metric). This was built in to make the algorithm flexible while testing observation types, but stands to enable easy generalization and modularity of the approach.

3. Results

Even without more intelligent probability distribution scoring, or any tweaks to the weighting of sub-observations, both convex and non-convex cases yield excellent results. Two simple polygonal worlds were tested with multiple start positions, capping the vertex count of the tree at different stages of growth to simulate external time limits, at which point the preemption ran to determine the current, most-likely best path. The figures below show three different stages of tree growth for the same start position in the convex world and in the non-convex world. The red circle shows the start position, the red cross shows goal location, the blue line is the optimal path in the absence of obstacles (straight line) for ease of comparison, the obstacles are plotted in gray or represented by the bounding black lines, and the best path found is highlighted in green.

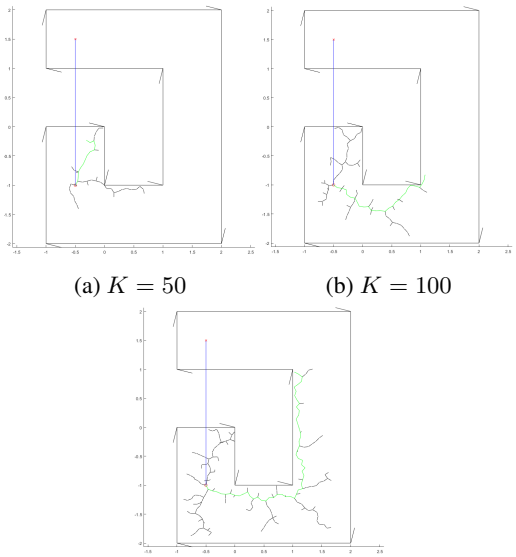
Across all iteration counts and start positions within the convex world, the returned path matched the path closest to the optimal by path length. Within the non-convex world, at low iteration counts, the path returned favored those that entered the local minima, as seen in Figure 2a. As the RRT grew however, this quickly changed, as the collision counts increased on the trapped paths, and the path length metric favored the longer, feasible path that was being constructed around the corner, even at the expense of initially increased distance to the goal and deviation from the straight-line path.



(a) $K = 50$ (b) $K = 100$

(c) $K = 200$

Figure 1: Convex world paths



(a) $K = 50$ (b) $K = 100$

(c) $K = 200$

Figure 2: Nonconvex world paths

4. Future Work

Key areas for future work include modification to scoring and weighting, as well as integration of the preemption scheme with other RRT variants. Regarding scoring and weighting, more accurate models of probability distributions across each of the observations can either be constructed *a priori* through analysis or experimentally by simulation through a wide variety of randomly generated environments. The same is true of the relative weighting of each sub-observation.

Integration with other anytime RRT variants, namely RT-RRT*, is the primary piece of future work that would be of value. Doing so would enable direct optimality comparisons between the preemptive and non-preemptive versions, in place of the more informal testing of Preemptive RRT seen above, for lack of a benchmark to compare to.

References

[1] M. Brunner, B. Brüggemann, and D. Schulz. Hierarchical rough terrain motion planning using an optimal sampling-based method. *2013 IEEE International Conference on Robotics and Automation*, pages 5539–5544, 2013. 1

[2] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. 1

[3] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2997–3004, 2014. 1

[4] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Iowa State University, 1998. 1

[5] S. M. LaValle and J. J. K. Jr. Randomized kinodynamic planning. *The International Journal of Robotics Research (IJRR)*, 20(5):378–400, 2001. 1

[6] K. Naderi, J. Rajamäki, and P. Hämäläinen. RT-RRT*: A real-time path planning algorithm based on RRT*. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, MIG '15, page 113–118, New York, NY, USA, 2015. Association for Computing Machinery. 1

[7] D. Nistér. Preemptive RANSAC for live structure and motion estimation. *Machine Vision and Applications*, 16:321–329, 2005. 1

[8] L. I. Reyes Castro, P. Chaudhari, J. Tůmová, S. Karaman, E. Frazzoli, and D. Rus. Incremental sampling-based algorithm for minimum-violation motion planning. In *52nd IEEE Conference on Decision and Control*, pages 3217–3224, 2013. 1

[9] C. Suh, T. T. Um, B. Kim, H. Noh, M. Kim, and F. C. Park. Tangent space RRT: A randomized planning algorithm on constraint manifolds. In *2011 IEEE International Conference on Robotics and Automation*, pages 4968–4973, 2011. 1