




WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
العربية
Deutsch
 Edit links

Create account Log in

Article **Talk**

Read **Edit** View history

Rapidly exploring random tree

From Wikipedia, the free encyclopedia

A **rapidly exploring random tree** (RRT) is an **algorithm** designed to efficiently search **nonconvex**, high-dimensional spaces by randomly building a **space-filling tree**. The tree is constructed incrementally from samples drawn randomly from the search space and is inherently biased to grow towards large unsearched areas of the problem. RRTs were developed by **Steven M. LaValle** and **James J. Kuffner Jr.** ^[1] ^[2] They easily handle problems with obstacles and differential constraints (**nonholonomic** and kinodynamic) and have been widely used in **autonomous robotic path planning**.

RRTs can be viewed as a technique to generate open-loop trajectories for nonlinear systems with state constraints. An RRT can also be considered as a **Monte-Carlo** method to bias search into the largest **Voronoi regions** of a graph in a configuration space. Some variations can even be considered **stochastic fractals**.^[3]

Contents [hide]

- 1 Description
- 2 Algorithm
- 3 See also
- 4 References
- 5 External links

Description [edit]

A RRT grows a tree rooted at the starting configuration by using random samples from the search space. As each sample is drawn, a connection is attempted between it and the nearest state in the tree. If the connection is feasible (passes entirely through free space and obeys any constraints), this results in the addition of the new state to the tree. With uniform sampling of the search space, the probability of expanding an existing state is proportional to the size of its **Voronoi region**. As the largest **Voronoi regions** belong to the states on the frontier of the search, this means that the tree preferentially expands towards large unsearched areas.

The length of the connection between the tree and a new state is frequently limited by a growth factor. If the random sample is further from its nearest state in the tree than this limit allows, a new state at the maximum distance from the tree along the line to the random sample is used instead of the random sample itself. The random samples can then be viewed as controlling the direction of the tree growth while the growth factor determines its rate. This maintains the space-filling bias of the RRT while limiting the size of the incremental growth.

RRT growth can be biased by increasing the probability of sampling states from a specific area. Most practical implementations of RRTs make use of this to guide the search towards the planning problem goals. This is accomplished by introducing a small probability of sampling the goal to the state sampling procedure. The higher this probability, the more greedily the tree grows towards the goal.

Algorithm [edit]

For a general **configuration space** *C*, the algorithm in **pseudocode** is as follows:

Part of a series on

Probabilistic data structures

Bloom filter · Count–min sketch · Quotient filter · Skip list

Random trees

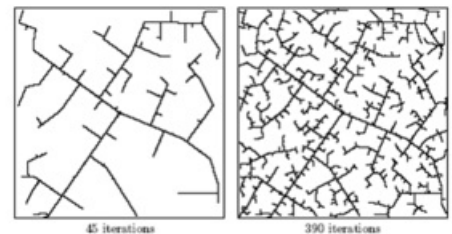
Random binary tree · Treap · Rapidly exploring random tree

Related

Randomized algorithm

 **Computer science portal**

v · t · e



A visualization of a RRT graph after 45 and 390 iterations

An animation of a RRT starting from iteration 0 to 10000

Algorithm BuildRRT

Input: Initial configuration q_{init} , number of vertices in RRT K , incremental distance Δq

Output: RRT graph G

```

G.init( $q_{init}$ )
for  $k = 1$  to  $K$ 
     $q_{rand} \leftarrow \text{RAND\_CONF}()$ 
     $q_{near} \leftarrow \text{NEAREST\_VERTEX}(q_{rand}, G)$ 
     $q_{new} \leftarrow \text{NEW\_CONF}(q_{near}, q_{rand}, \Delta q)$ 
    G.add_vertex( $q_{new}$ )
    G.add_edge( $q_{near}, q_{new}$ )
return  $G$ 

```

- " \leftarrow " is a shorthand for "changes to". For instance, " $largest \leftarrow item$ " means that the value of *largest* changes to the value of *item*.
- "**return**" terminates the algorithm and outputs the value that follows.

In the algorithm above, "**RAND_CONF**" grabs a random configuration q_{rand} in C . This may be replaced with a function "**RAND_FREE_CONF**" that uses samples in C_{free} , while rejecting those in C_{obs} using some collision detection algorithm.

"**NEAREST_VERTEX**" is a straightforward function that runs through all vertexes v in graph G , calculates the distance between q_{rand} and v using some measurement function thereby returning the nearest vertex.

"**NEW_CONF**" selects a new configuration q_{new} by moving an incremental distance Δq from q_{near} in the direction of q_{rand} . (According to [4] in holonomic problems, this should be omitted and q_{rand} used instead of q_{new})

See also

[\[edit\]](#)

- Probabilistic roadmap
- Space-filling tree
- Motion planning
- Randomized algorithm

References

[\[edit\]](#)

- ↑ LaValle, Steven M. (October 1998). "Rapidly-exploring random trees: A new tool for path planning" (PDF). *Technical Report* (Computer Science Department, Iowa State University) (TR 98-11).
- ↑ LaValle, Steven M.; Kuffner Jr., James J. (2001). "Randomized Kinodynamic Planning" (PDF). *The International Journal of Robotics Research (IJRR)* **20** (5). doi:10.1177/02783640122067453 .
- ↑ <http://msl.cs.uiuc.edu/rrt/about.html> About RRTs, by Steve LaValle
- ↑ Rapidly-Exploring Random Trees: Progress and Prospects (2000), by Steven M. Lavalley , James J. Kuffner , Jr. Algorithmic and Computational Robotics: New Directions, <http://eprints.kfupm.edu.sa/60786/1/60786.pdf>

External links

[\[edit\]](#)

- Java visualizer of RRT and RRT* including map editor

Categories: [Search algorithms](#) | [Robot control](#) | [Probabilistic data structures](#)

This page was last modified on 12 April 2015, at 07:41.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

