# Rapidly-exploring Random Trees (RRTs)

| | | |
|---|---|---|
| MENTOR | Professor N. S. Kumar | |
| MENTEE | Arya Rajiv Chaloli | PES1201700253 |
| SUBJECT | Design and Analysis of Algorithms | UE17CS251 |

## AIM

To create an implementation of the Rapidly-exploring Random Trees (RRT) approach for path planning applications in the C-Programming language and to delve further on to improving the efficiency during run-time by parallelizing the operation of generating the tree.

## OVERVIEW

### PATH PLANNING

Path or Motion planning is a widely used in robotics for the process of breaking down a desired movement task into discrete motions that satisfy movement constraints and possibly optimize some aspect of the movement. A fundamental need in robotics is to have algorithms that convert high-level specifications of tasks from humans in to low-level descriptions of how to move: motion planning and trajectory planning.

The process of path planning primarily consists of a configuration space, the set of all possible configurations that the system can be in, which comprises of the free space ($C_{free}$), obstacle space and the target space.

Motion planning has several robotics applications, such as autonomy, automation, and robot design in CAD software, as well as applications in other fields, such as animating digital characters, video game, artificial intelligence, architectural design, robotic surgery, and the study of biological molecules.

### POSSIBLE APPROACHES

Motion or path planning can be implemented using any of the following approaches.

1. Grid-based search algorithms that overlay a grid on top of configuration space on which the system is allowed to make basic, step-wise incremental moves within the free space.

2. Interval-based search algorithms that generate a paving covering the entire configuration space. The paving is then decomposed into 2 subpavings X and X' such that $X<C_{free}<X'$. The system is allowed to move freely in X but is constrained to be within X'. To both the subpavings, a neighbour graph is built and paths can be found using algorithms such as Dijkstra or A*.
3. Geometric algorithms that computes the move towards the direction of the longest ray from a given bundle of rays around the current position attributed with their length hitting a wall.
4. Reward-based algorithms that computes the path based on a system of rewards and punishments, where the robot is rewarded for moves towards the successful completion of the path.
5. Potential-field algorithms treat the robot's configuration as a point in a potential field that combines attraction to the goal, and repulsion from obstacles. The resulting trajectory is output as the path.
6. Sampling-based algorithms represent the configuration space with a roadmap of sampled configurations. Sampled points undergo a collision detection test to decide the inclusion of the point in $C_{free}$. To find a path that connects the start and the target, these sampled points are incrementally are appended to the roadmap.

Low-dimensional problems can be solved with grid-based algorithms or geometric algorithms that compute the shape and connectivity of $C_{free}$. Exact motion planning for high-dimensional systems under complex constraints is computationally intractable. Potential-field algorithms are efficient, but fall prey to local minima (an exception is the harmonic potential fields). Sampling-based algorithms avoid the problem of local minima, and solve many problems quite quickly. They are unable to determine that no path exists, but they have a probability of failure that decreases to zero as more time is spent.

Sampling-based algorithms are currently considered state-of-the-art for motion planning in high-dimensional spaces, and have been applied to problems which have dozens or even hundreds of dimensions (robotic manipulators, biological molecules, animated digital characters, and legged robots).

## CHOSEN APPROACH: RAPIDLY EXPLORING RANDOM TREES

Rapidly-exploring Random Trees is an incremental sampling and searching approach, which incrementally constructs a search tree that gradually improves in resolution. A dense sequence of samples is used as a guide in the incremental construction of the tree. If this sequence is random, the resulting tree is called a Rapidly-exploring Random Tree (RRT).

The search tree initially starts with a single node. For the next k-iterations, a tree is iteratively grown by connecting a randomly generated sample-points to the nearest point in the tree. In every iteration, the sampled point becomes a vertex. Therefore, the resulting tree is dense.

If the nearest point to the sampled point in the tree is a vertex of the tree, then an edge is made from the vertex to the sampled point. However, if the nearest point lies in the interior of an edge, then the existing edge is split such that the nearest point appears as a new vertex, and an edge is made from this new vertex to the sampled point. Due to this edge splitting property, the total number of edges may increase by one or two in each iteration, providing greater density to the tree, which results in a smoother path from the start point to the destination.

### EXPLORATION IN THE PROJECT

This project intends to implement Rapidly-exploring Random Trees in the C-Programming Language for optimal path planning from one point to another; scoped by motion in a two-dimensional space with obstacles where there is at least one finite solution to the path that can be constructed as the solution.
In addition to this, the project, aims at parallelizing this implementation to improve the run-time of the algorithm.

## REFERENCES

1. Introduction to Autonomous Robots
   By Nikolaus Correll,
   https://github.com/correll/Introduction-to-Autonomous-Robots/releases/

2. Planning Algorithms
   By Steven M. LaValle,
   http://planning.cs.uiuc.edu/

3. https://en.wikipedia.org/wiki/Motion_planning/

4. http://www.centropiaggio.unipi.it/sites/default/files/course/material/srd_cap4_mp_0.pdf