Practical Exercise Rapidly Exploring Random Trees (RRT)

Goals:

In this practical exercise you should implement a sampling based motion planner based on Rapidly Exploring Random Trees (RRT). The same robot and environment of the homework PRM is used for task 2 and the voluntary task. It is highly recommended, that you use the same framework.

Introduction

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.

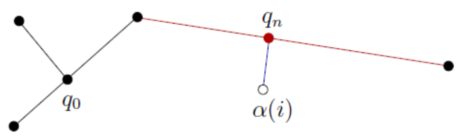
They can also handle problems with obstacles and differential constraints (nonholonomic and kinodynamic) but this is out of the scope of this exercise.

1. Task

You should implement a simple **two-dimensional** RRT **without** any obstacles in the unit square [0,1]x[0,1]. The start point is (0.5, 0.5). Document several trees as a graphic for different resolutions (number of nodes k).

The following algorithm (from figure 5.16, p. 229 LaValle: Planning Algorithms) needs to be implemented:

SIMPLE_RDT(q_0)
1 $\mathcal{G}$.init(q_0);
2 **for** $i = 1$ **to** k **do**
3 $\quad\mathcal{G}$.add_vertex($\alpha(i)$);
4 $\quad q_n \leftarrow$ NEAREST($S(\mathcal{G}), \alpha(i)$);
5 $\quad\mathcal{G}$.add_edge($q_n, \alpha(i)$);



If in step 4 the nearest point in S lies in an edge (instead of a node), then the edge is split into two, and a new vertex is inserted into G.

<u>Hint:</u> if you use the framework given for the PRM exercise, only use a Random Number Generator (RNG) for the first two degrees of freedom, so do NOT use a RNG for the additional (rotational) degrees of freedom.

The following requirements must be fulfilled:

- The step Nearest Neighbour must be done efficiently, for example based on a k-d tree approach.
- Generating the metrics: number of nodes, number of edges, …
- The random number generator is seeded with 0

2. Task

You should implement a RRT for the 5-dof robot and environment given in the homework PRM.

The following algorithm (from figure 5.21, p. 232 LaValle: Planning Algorithms) needs to be implemented:

RDT(q_0)
$\mathcal{G}$.init(q_0);
for $i = 1$ **to** k **do**
  $q_n \leftarrow$ NEAREST($S, \alpha(i)$);
  $q_s \leftarrow$ STOPPING-CONFIGURATION($q_n, \alpha(i)$);
  if $q_s \neq q_n$ **then**
    $\mathcal{G}$.add_vertex(q_s);
    $\mathcal{G}$.add_edge(q_n, q_s);

The same test cases as in the homework PRM needs to be addressed.

The following requirements must be fulfilled:

- The step Nearest Neighbour must be done efficiently, for example based on a k-d tree approach.
- Generating the metrics: number of nodes, number of edges, …
- The random number generator is seeded with 0

Voluntary Task

You should implement a balanced bidirectional RRT for the 5-dof robot and environment given in the homework PRM.

The following algorithm (from figure 5.24, p. 236 LaValle: Planning Algorithms) needs to be implemented:

```
RDT_BALANCED_BIDIRECTIONAL(q_I, q_G)
T_a.init(q_I); T_b.init(q_G);
for i = 1 to K do
    q_n ← NEAREST(S_a, α(i));
    q_s ← STOPPING-CONFIGURATION(q_n, α(i));
    if q_s ≠ q_n then
        T_a.add_vertex(q_s);
        T_a.add_edge(q_n, q_s);
        q'_n ← NEAREST(S_b, q_s);
        q'_s ← STOPPING-CONFIGURATION(q'_n, q_s);
        if q'_s ≠ q'_n then
            T_b.add_vertex(q'_s);
            T_b.add_edge(q'_n, q'_s);
        if q'_s = q_s then return SOLUTION;
    if |T_b| > |T_a| then SWAP(T_a, T_b);
return FAILURE
```

The same test cases as in the homework PRM needs to be addressed.

The following requirements must be fulfilled:

- The step Nearest Neighbour must be done efficiently, for example based on a k-d tree approach.
- Generating the metrics for **both trees**: number of nodes, number of edges, …

Preparation after

You should prepare (pairwise) a document, which

- Describes the task
- Describes the solution in detail including pictures
- Comprises a well documented code
- Describes the lessons learned
- Is uploaded to Moodle in time.

You check in your document and your code as a .zip file into moodle. Provide only the .sln solution and source files, so that the overall size keeps small.