RRT-Connect path solving

J.J. Kuffner and S.M. LaValle
Talk Overview

• Introduction
• Previous work
• Algorithm
• Examples
• Performance
• Conclusions
Introduction

• Given a “configuration space” we need to find a path between a start point and a destination

Paper impact: about 200 citations, over 200000 google query results
Introduction
Previous Work

Single Query Planning:
- one path needs to be computed for the environment fast and without preprocessing
- popular method: “randomized potential field”

Multiple Query Planning:
- Many paths will be computed for the same environment and thus the environment model can be preprocessed
- popular method: “probabilistic roadmap approach”
Previous Work

RRT (rapidly exploring random trees)

- $C$: configuration space where $q$ belongs to $C$ and describes the position and orientation of a body placed in the space.
- $C_{\text{free}}$: set of configuration where the body does not collide with obstacles.
Previous Work

```plaintext
BUILD_RRT(q_{init})
1  T.init(q_{init});
2  for k = 1 to K do
3    q_{rand} ← RANDOM_CONFIG();
4    EXTEND(T, q_{rand});
5  Return T

EXTEND(T, q)
1  q_{near} ← NEAREST_NEIGHBOR(q, T);
2  if NEW_CONFIG(q, q_{near}, q_{new}) then
3    T.add_vertex(q_{new});
4    T.add_edge(q_{near}, q_{new});
5    if q_{new} = q then
6      Return Reached;
7    else
8      Return Advanced;
9  Return Trapped;
```
Previous Work

• Why are RRT’s rapidly exploring?

- the probability of a node to be selected for expansion is proportional to the area of its Voronoi region
The Algorithm

• RRT-connect is a variation of RRT
  – grows two trees from both the source and destination until they meet
  – grows the trees towards each other (rather then towards random configurations)
  – the greediness becomes stronger by growing the tree with multiple epsilon steps instead of a single one
The Algorithm

CONNECT(\(T, q\))
1 repeat
2 \(S \leftarrow \text{EXTEND}(T, q);\)
3 until not (\(S = \text{Advanced}\))
4 Return \(S;\)

RRT_CONNECT_PLANNER(\(q_{init}, q_{goal}\))
1 \(T_a.\text{init}(q_{init}); T_b.\text{init}(q_{goal});\)
2 for \(k = 1\) to \(K\) do
3 \(q_{rand} \leftarrow \text{RANDOM_CONFIG();}\)
4 if not (\(\text{EXTEND}(T_a, q_{rand}) = \text{Trapped}\)) then
5 if (\(\text{CONNECT}(T_b, q_{new}) = \text{Reached}\)) then
6 Return PATH(\(T_a, T_b\));
7 SWAP(\(T_a, T_b\));
8 Return Failure
The Algorithm

• The approach is flexible:
  – single epsilon step instead of multiple ones
  – single tree but with multiple epsilon steps
  – only add the last $q_{new}$ to minimize the number of nodes
Examples
Examples
Examples
Examples
Performance

• much faster than common RRT methods for uncluttered environments and slightly faster in very cluttered environments
• 2D cases are solved in <= 1 second depending on the complexity of the situation
• 3D piano scene required 12 seconds
• 6 DOF robot arm required 4 seconds
Conclusions

• Improved version of RRT for faster convergence
• Finds paths in high dimensional spaces at interactive time rates
• Experiments showed it to be consistent
• Drawback: a lot of nearest neighbor searches are performed