RRT-Connect path solving

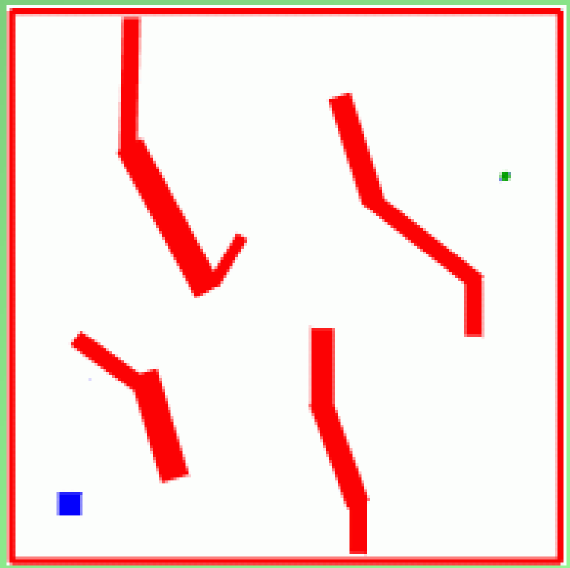
J.J. Kuffner and S.M. LaValle

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 place in the space.
- C_{free} : set of configuration where the body does not collide with obstacles

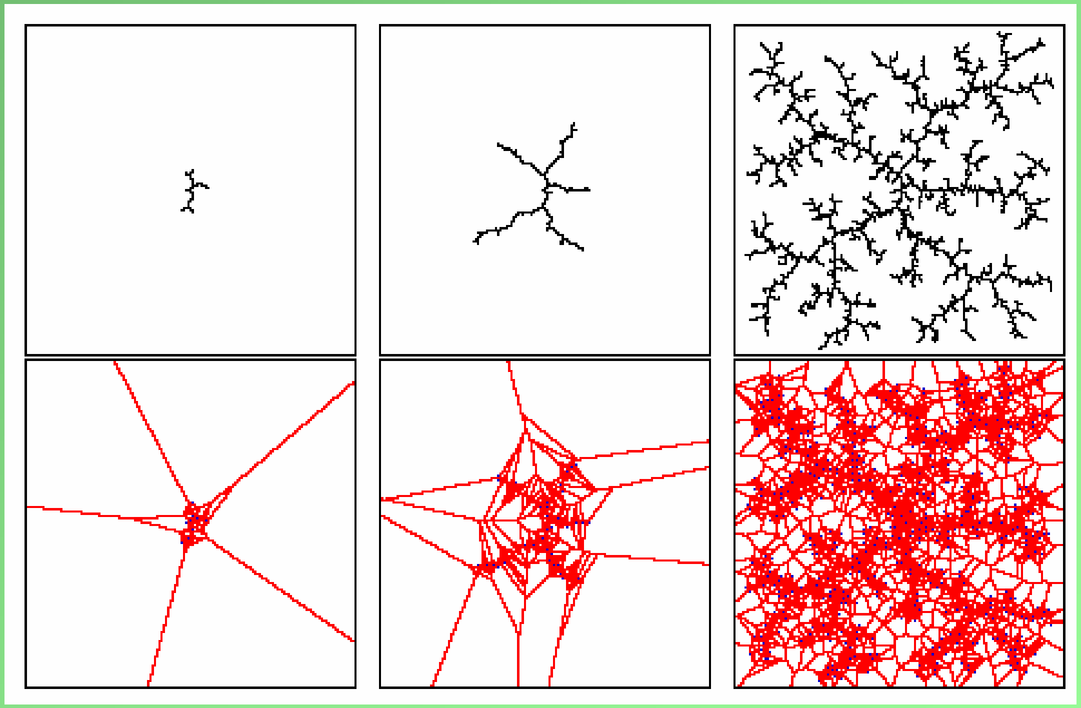
Previous Work

BUILD_RRT(q_{init})
$\mathcal{T}$.init(q_{init});
for $k = 1$ **to** K **do**
    $q_{rand} \leftarrow$ RANDOM_CONFIG();
    EXTEND($\mathcal{T}, q_{rand}$);
Return $\mathcal{T}$

EXTEND($\mathcal{T}, q$)
$q_{near} \leftarrow$ NEAREST_NEIGHBOR($q, \mathcal{T}$);
if NEW_CONFIG(q, q_{near}, q_{new}) **then**
    $\mathcal{T}$.add_vertex(q_{new});
    $\mathcal{T}$.add_edge(q_{near}, q_{new});
    if $q_{new} = q$ **then**
        Return *Reached*;
    else
        Return *Advanced*;
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

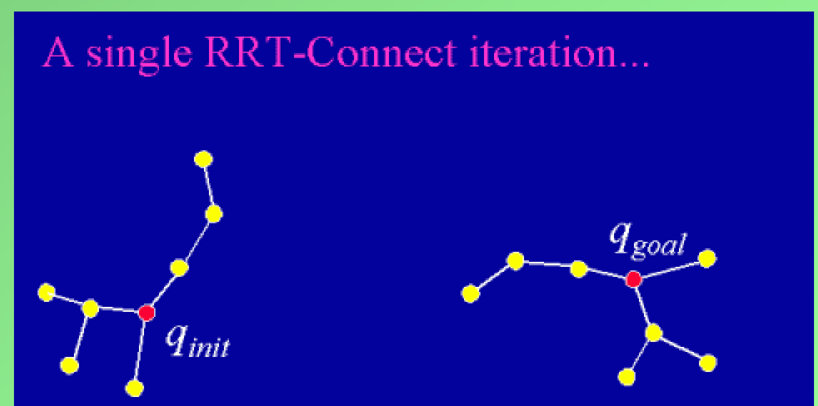
The Algorithm

- 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)
    repeat
        S ← EXTEND(T, q);
    until not (S = Advanced)
    Return S;
```

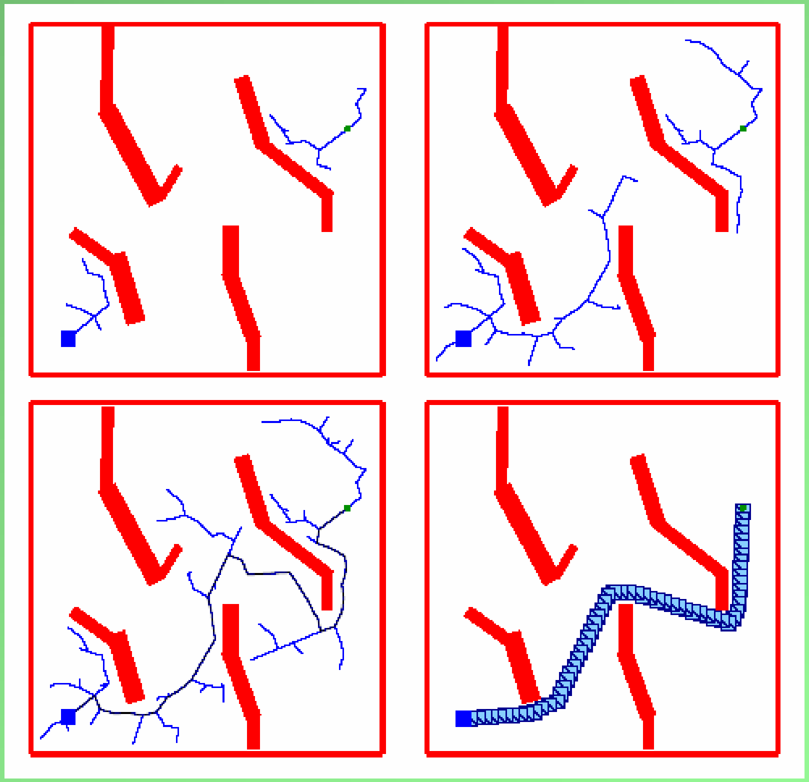
```
RRT_CONNECT_PLANNER(q_init, q_goal)
    T_a.init(q_init); T_b.init(q_goal);
    for k = 1 to K do
        q_rand ← RANDOM_CONFIG();
        if not (EXTEND(T_a, q_rand) = Trapped) then
            if (CONNECT(T_b, q_new) = Reached) then
                Return PATH(T_a, T_b);
        SWAP(T_a, T_b);
    Return Failure
```



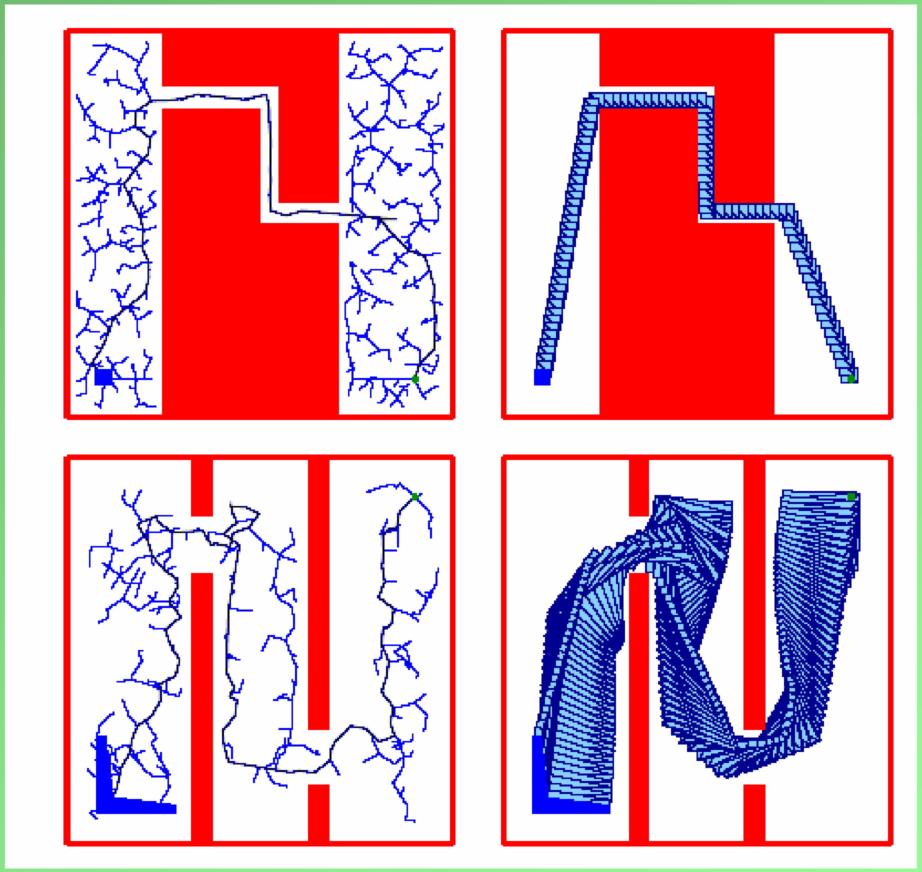
The Algorithm

- 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

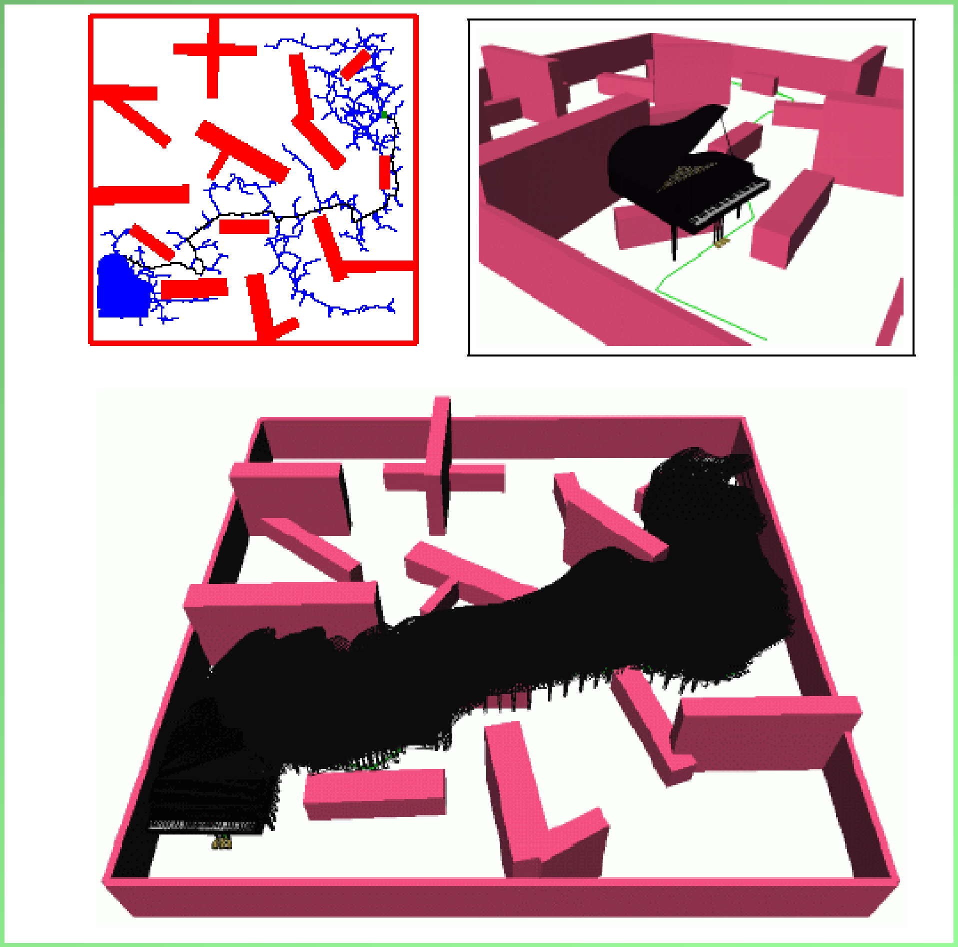
Examples



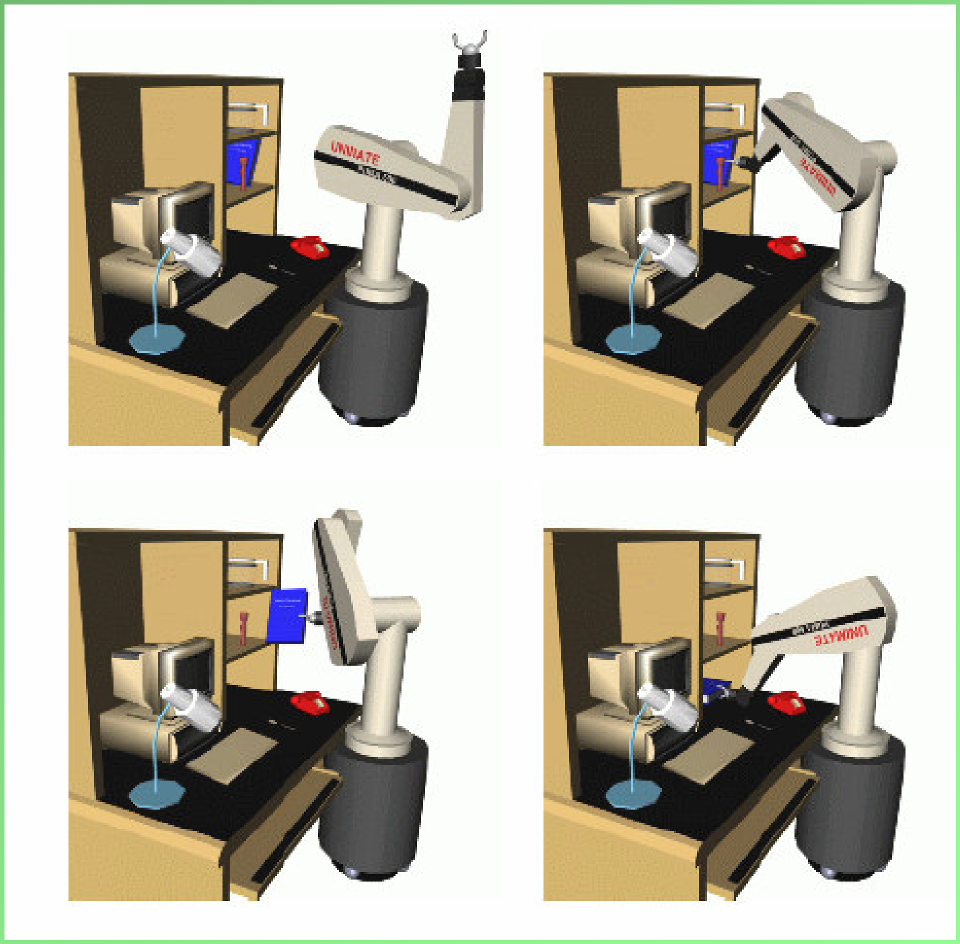
Examples



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