Week2: RRT

Zhaoliang Zheng

Review of Week 1

1. Finished the research proposal.

Target-oriented UAV auto-control based on SLAM(Simultaneous localization and mapping)

2. Had a pleasure discussion with professors and decided my beginning direction.

1) Path planning, navigating strategy...
2) RRT: Rapidly-exploring Random Trees
3) Rtab-map: Real-Time Appearance-Based Mapping
4) ….

Works in week 2

1. Understanded the principle of RRT

2. Studied and learnt from the classical RRT algorithm, digested it.

3. Developed a usable RRT algorithm and applied it to 2D path planning.

4. Further comprehension on RRT

RRT-Rapidly-Exploring Random Trees

RRT-Rapidly-Exploring Random Trees is a very common robotic path planning method. Essentially, it is a randomized data structure---Trees.

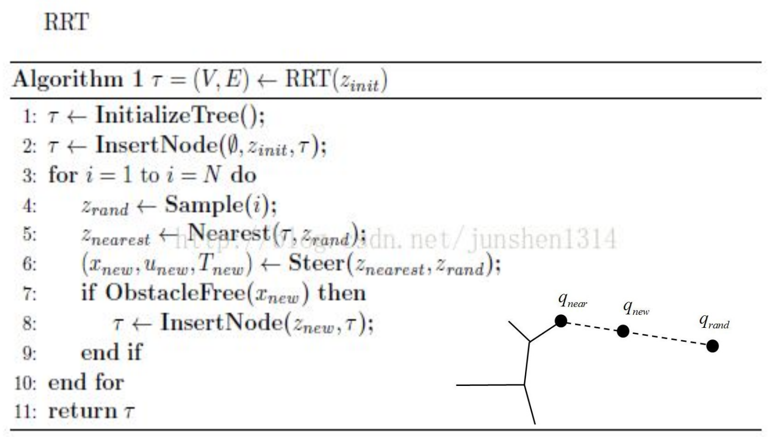
RRT is designed for a broad class of path planning problems. While they share many of the beneficial properties of existing randomize planning techniques, RRTs are specifically designed to handle nonholonomic constraints(including dynamics) and high degrees of freedom.[1]LaValle, S.M., Rapidly-exploring random trees: A new tool for path planning. 1998.

Path planning also known as motion planning is an important task for autonomous robotics. It's aim is to find shortest or at least optimal path between start and goal positions. It is hard and highly demanded task today. [2]Emre Ozan Alkan,Autonomous Robotics RRT Algorithm,2014

Basic RRT algorithm

procedure :

Start: In the environment, we define an initial point: q_init, then we randomly pick a point in the environment, we got q_rand. If q_rand is not in obstacle area, link q_init and q_rand, we got line L, and if L is not in obstacle area, move along the L for a specific distance, we got q_new. Between q_init and q_new, we form a simplest tree.



RRT

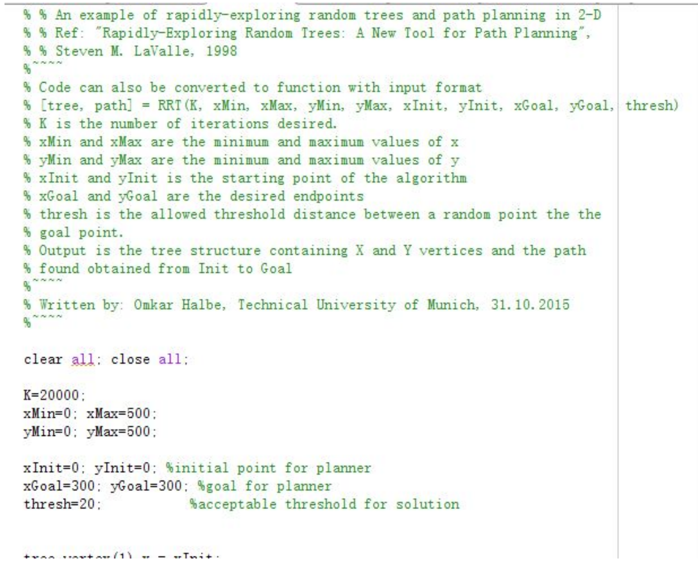
```
Algorithm 1 $\tau = (V, E) \leftarrow \text{RRT}(z_{init})$
1:  $\tau \leftarrow$ InitializeTree();
2:  $\tau \leftarrow$ InsertNode($\emptyset, z_{init}, \tau$);
3:  for $i = 1$ to $i = N$ do
4:      $z_{rand} \leftarrow$ Sample($i$);
5:      $z_{nearest} \leftarrow$ Nearest($\tau, z_{rand}$);
6:      $(x_{new}, u_{new}, T_{new}) \leftarrow$ Steer($z_{nearest}, z_{rand}$);
7:      if ObstacleFree($x_{new}$) then
8:          $\tau \leftarrow$ InsertNode($z_{new}, \tau$);
9:      end if
10: end for
11: return $\tau$
```

Expansion of Trees: On the basis of the start, continue to repeat, sprinkle in the environment, have no point q_rand in obstacle areas, then look for a q_near(the closest point to q_rand) in the Tree, connecting two points, if no obstacles on the line, move along with the line, from q_near to q_rand, get new points, x_new, that point is added to the existing tree.

Planning: repeat the above process, until target point is added to the tree, then we can find a path from the initial point to the target point.

Basic RRT algorithm

There are some source code in Mathwork.com , the most classical one is as follow:



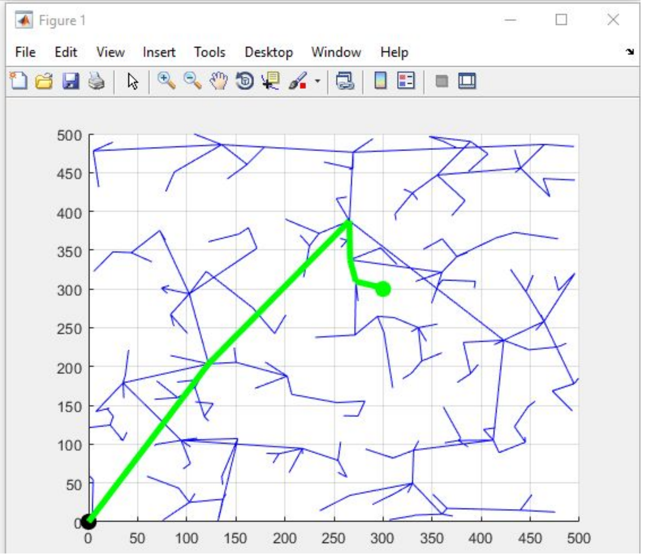
```
% % An example of rapidly-exploring random trees and path planning in 2-D
% % Ref: "Rapidly-Exploring Random Trees: A New Tool for Path Planning",
% % Steven M. LaValle, 1998
%~~~~
% Code can also be converted to function with input format
% [tree, path] = RRT(K, xMin, xMax, yMin, yMax, xInit, yInit, xGoal, yGoal, thresh)
% K is the number of iterations desired.
% xMin and xMax are the minimum and maximum values of x
% yMin and yMax are the minimum and maximum values of y
% xInit and yInit is the starting point of the algorithm
% xGoal and yGoal are the desired endpoints
% thresh is the allowed threshold distance between a random point the the
% goal point.
% Output is the tree structure containing X and Y vertices and the path
% found obtained from Init to Goal
%~~~~
% Written by: Omkar Halbe, Technical University of Munich, 31.10.2015
%~~~~

clear all; close all;

K=20000;
xMin=0; xMax=500;
yMin=0; yMax=500;

xInit=0; yInit=0; %initial point for planner
xGoal=300; yGoal=300; %goal for planner
thresh=20;            %acceptable threshold for solution


tree.vertex(1).x = xInit;
```



---[1]LaValle, S.M., Rapidly-exploring random trees: A new tool for path planning. 1998.

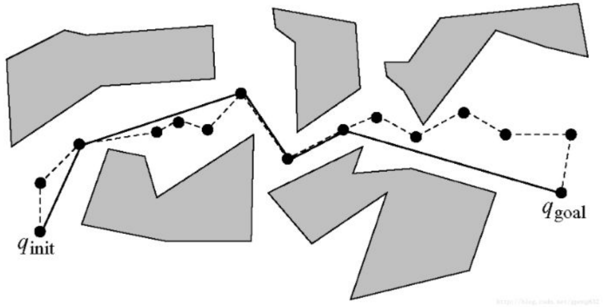
Little Improvement on RRT

When we consider obstacles on the map, things will become more complicated. After you retrieve the path, you cannot always expect UAV to follow that path. Since RRT algorithm creating random points, tend to build tree towards random points, mostly solutionpath is not optimal and very noisy. Hence, we need some kind of noise reduction method for our path.

https://www.youtube.com/watch?v=E-IUAL-D9SY

As for UAV, in a small area, The Greedy algorithm will be good enough. Schematic image:

Therefore, we can implement smoothing method on path to get shorter and less noisy path. I used 'Greedy approach' where each time we trying our connect our latest point with the start position and so on.



Experiment(1)

To implement RRT algorithm, we could start from some simple models : 2D map

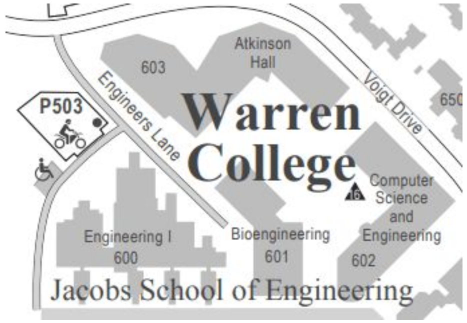
Procedure:

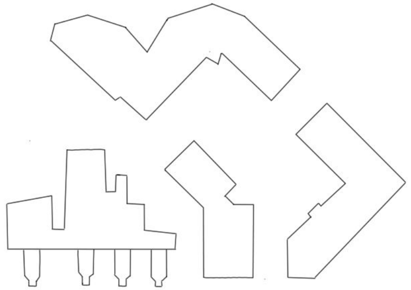
1. Create a 2D map, load into Matlab and binarize the map, get everything prepared.
2. Design a puso-code and the each function that in the main program:

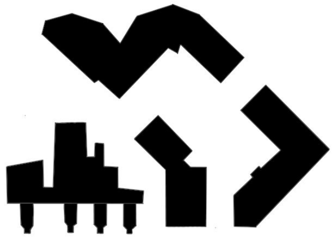
1) Main function
2) Test if goal arrived function
3) Get q_near and q_new function
4) Add into vertices function
5) Retrieve path function
6) Smooth path function

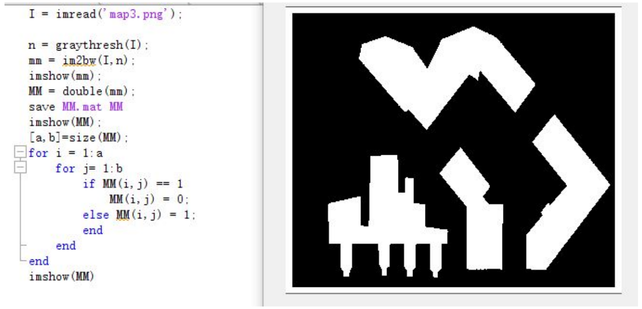
3. Implement it to our map and debugging.

Experiment (2)









```
I = imread('map3.png');

n = graythresh(I);
mm = im2bw(I,n);
imshow(mm);
MM = double(mm);
save MM.mat MM
imshow(MM);
[a,b]=size(MM);
for i = 1:a
    for j= 1:b
        if MM(i,j) == 1
            MM(i,j) = 0;
        else MM(i,j) = 1;
        end
    end
end
imshow(MM)
```

Experiment(3)--demo

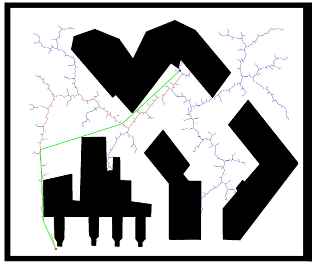
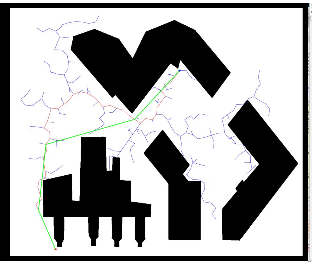
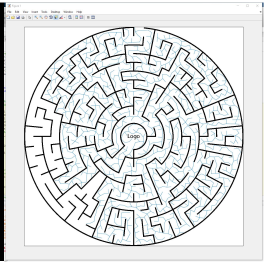
Further understanding to RRT

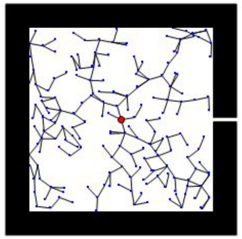
Core of RRT: [tree,path] = RRT(K,xMin,xMax,yMin,yMax,XInit,YInit,)

Deficiency:

1.It is a kind of pure random search algorithm that is not sensitive to environmental types. When there are a lot of obstacles or narrow channel constraints in space, the convergence speed of the algorithm is slow and the efficiency will be greatly reduced.

2. Another weakness of RRT is the difficulty of finding a path in an environment with narrow channels. Because narrow channel area is small, be touched probability is low.



Bugs

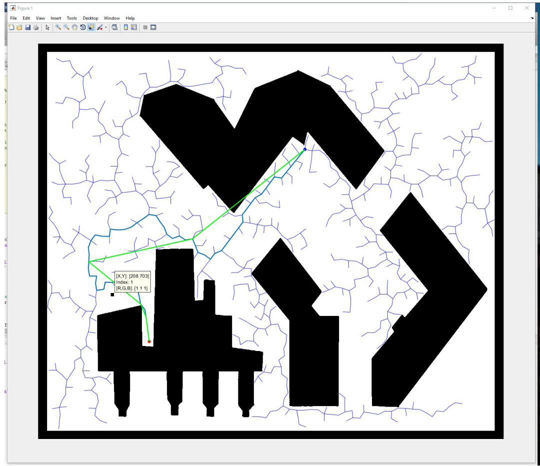


Figure 1
File Edit View Insert Tools Desktop Window Help
[X,Y]: [208 703]
Index: 1
[R,G,B]: [1 1 1]

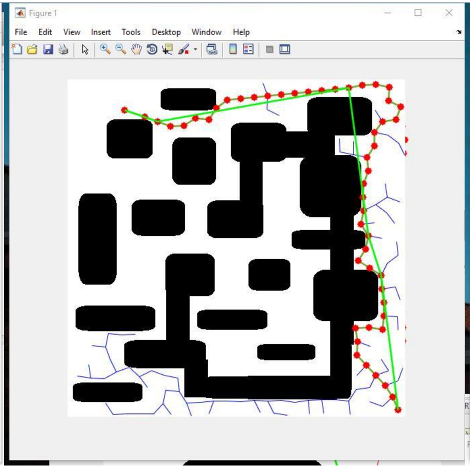


Figure 1
File Edit View Insert Tools Desktop Window Help

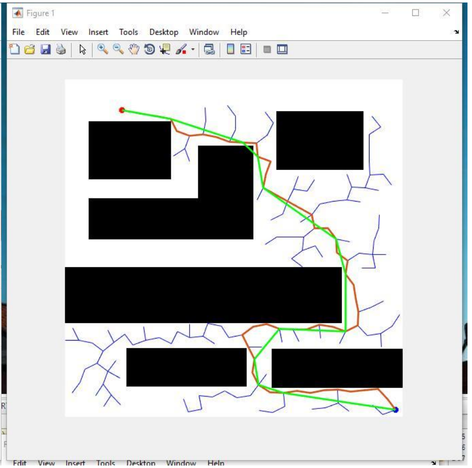


Figure 1
File Edit View Insert Tools Desktop Window Help