深蓝学院
shenlanxueyuan.com

第三章作业思路讲解

主讲人 汪世玉

纲要

深蓝学院
shenlanxueyuan.com

- 第一题:MATLAB(手撕代码实现，重要)
  - RRT(必做)
  - RRT*
  - Informed-RRT*
- 第二题:ROS(调OMPL库)
- 高频问题

第一题:MATLAB

深蓝学院 shenlanxueyuan.com

Step 1: 在地图中随机采样一个点x_rand

Step 2: 遍历树，从树中找到最近邻近点x_near

Step 3: 扩展得到x_new节点 ,检查collision-free

Step 4: 将x_new插入树T

Step 5:检查是否到达目标点附近

Step 6:将x_near和x_new之间的路径画出来

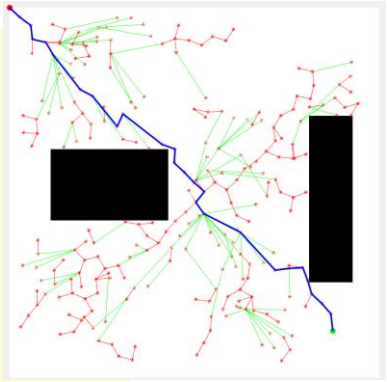
```
for iter = 1:3000
    x_rand=[];
    x_rand(1) = xL*rand;
    x_rand(2) = yL*rand;
    x_near=[];
    min_dist = 1000000;
    near_iter = 1;
    %%=======寻找x_near==========%%
    [~,N]=size(T.v);
    for j = 1:N
       x_near(1) = T.v(j).x;
       x_near(2) = T.v(j).y;
       dist = norm(x_rand - x_near);
       if min_dist > dist
           min_dist = dist;
           near_iter = j;
       end
    end
    x_near(1) = T.v(near_iter).x;
    x_near(2) = T.v(near_iter).y;
    %%========获取x_new============%%
    x_new=[];
    near_to_rand = [x_rand(1)-x_near(1),x_rand(2)-x_near(2)];
    normlized = near_to_rand / norm(near_to_rand) * Delta;
    x_new = x_near + normlized;
    %%=======障碍检测==============%%
    if ~collisionChecking(x_near,x_new,Imp)
       continue;
    end
    count=count+1;
    %%========将X_NEW增加到树中========%%
    T.v(count).x = x_new(1);
    T.v(count).y = x_new(2);
    T.v(count).xPrev = x_near(1);
    T.v(count).yPrev = x_near(2);
    T.v(count).dist= norm(x_new - x_near);
    T.v(count).indPrev = near_iter;
    plot([x_near(1),x_new(1)],[x_near(2),x_new(2)],'-r');
    hold on;
    %========判断是否找到路径==========%%
    if norm(x_new - goal) < Thr
        break;
    end
    pause(0.1);
end
```

RRT*

深蓝学院 shenlanxueyuan.com

```
%%========获取x_new============%%
x_new=[];
near_to_rand = [x_rand(1)-x_near(1),x_rand(2)-x_near(2)];
normlized = near_to_rand / norm(near_to_rand) * Delta;
x_new = x_near + normlized;
%%========障碍检测==============%%
if ~collisionChecking(x_near,x_new,Imp)
    continue;
end

%%=======  nearC && chooseParent  =========%%
nearptr = [];
nearcount = 0;
neardist =  norm(x_new - x_near) + T.v(near_iter_tmp).dist;
for j = 1:N
    if j == near_iter_tmp
        continue;
    end
    x_neartmp(1) = T.v(j).x;
    x_neartmp(2) = T.v(j).y;
    dist = norm(x_new - x_neartmp) + T.v(j).dist;
    norm_dist = norm(x_new - x_neartmp);
    if norm_dist < 120
        %nearC
        if collisionChecking(x_neartmp,x_new,Imp)
            nearcount = nearcount + 1;
            nearptr(nearcount,1) = j;
            if neardist > dist
                neardist = dist;
                near_iter = j;
            end
        end
    end
end

x_near(1) = T.v(near_iter).x;
x_near(2) = T.v(near_iter).y;
count=count+1;
```

```
%%========将X_NEW增加到树中========%%
T.v(count).x = x_new(1);
T.v(count).y = x_new(2);
T.v(count).xPrev = x_near(1);
T.v(count).yPrev = x_near(2);
T.v(count).dist= norm(x_new - x_near) + T.v(near_iter).dist;
T.v(count).indPrev = near_iter;

%%========  rewirte  =========%%
[M,~] = size(nearptr);
for k = 1:M
    x_1(1) = T.v(nearptr(k,1)).x;
    x_1(2) = T.v(nearptr(k,1)).y;
    x1_prev(1) = T.v(nearptr(k,1)).xPrev;
    x1_prev(2) = T.v(nearptr(k,1)).yPrev;
    if T.v(nearptr(k,1)).dist >  (T.v(count).dist + norm(x_1-x_new))
        T.v(nearptr(k,1)).dist = T.v(count).dist + norm(x_1-x_new);
        T.v(nearptr(k,1)).xPrev = x_new(1);
        T.v(nearptr(k,1)).yPrev = x_new(2);
        T.v(nearptr(k,1)).indPrev = count;
        plot([x_1(1),x1_prev(1)],[x_1(2),x1_prev(2)],'-w');
        hold on;
        plot([x_1(1),x_new(1)],[x_1(2),x_new(2)],'-g');
        hold on;
    end
end

plot([x_near(1),x_new(1)],[x_near(2),x_new(2)],'-r');
hold on;
plot(x_new(1),x_new(2),'*r');
hold on;
if norm(x_new - goal) < Thr
    break;
end
pause(0.1);
```



- 第一步:RRT
- 第二部:rewire

Informed-RRT*

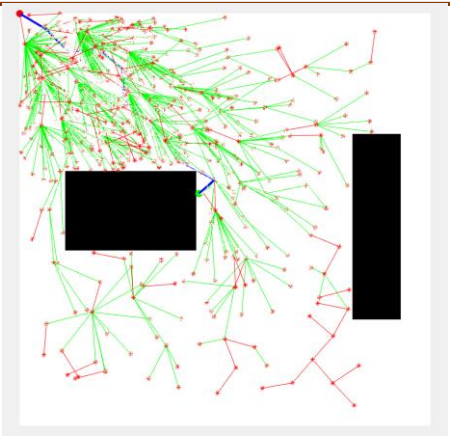
深蓝学院 shenlanxueyuan.com

- main.m

```
for iter = 1:1000
    x_rand=[];
    %%=========采样想x_rand========%%
    if start_goal_dist < 1000000
        while 1
            x_rand(1) = xL*rand;
            x_rand(2) = yL*rand;
            if new_node(x_rand(1),x_rand(2),start_goal_dist)
                break;
            end
        end
    else
        x_rand(1) = xL*rand;
        x_rand(2) = yL*rand;
    end
```

- new_node

```
function feasible = new_node( x,y,path_dist )
%UNTITLED2 此处显示有关此函数的摘要
%   此处显示详细说明
    feasible = 0;
    diff = 350.5;
    alpha = 1/4*pi;
    a = path_dist / 2;
    c = sqrt(2) * 699 / 2;
    if a>c
        b = sqrt(a*a - c*c);
    else
        fprintf("a<c\n");
    end
    u = (x-diff)*cos(alpha) + (y-diff)*sin(alpha);
    v = -(x-diff)*sin(alpha) + (y-diff)*cos(alpha);
    dist = (((x-diff)*cos(1/4*pi) + (y-diff)*sin(1/4*pi))^2) / (a^2) + ((-(x-diff)*sin(1/4*pi) + (y-diff)*cos(1/4*pi))^2 ) / (b^2);

    if dist <= 1
         feasible = 1;
    else
         feasible = 0;
    end
end
```



- 如何确定椭圆：
  - 焦点：起点终点
  - 长半轴：路径长度/2

第二题：ROS

深蓝学院
shenlanxueyuan.com

OMPL调用流程：

1.构建状态空间 RealVectorStateSpace(3)；

2.设置状态空间边界 realVectorBounds(3)；

3.构建状态信息 SpaceInformation；

4.构建问题实例 ProblemDefinition；

5.构造起点和终点并设置值ScopedState；

6.设置优化目标，这里用路径长度；

7.构建规划器 Planner，例如RRTstar；

8.规划器求解 Planner->solve()；

9.若求解成功，则调用getSolutionPath()获取路径点

第二题：ROS


```
// Set our robot's starting state to be the bottom-left corner of
// the environment, or (0,0).
ob::ScopedState<> start(space);
start->as<ob::RealVectorStateSpace::StateType>()->values[0] = start_pt(0);
start->as<ob::RealVectorStateSpace::StateType>()->values[1] = start_pt(1);
start->as<ob::RealVectorStateSpace::StateType>()->values[2] = start_pt(2);
// Set our robot's goal state to be the top-right corner of the
// environment, or (1,1).
ob::ScopedState<> goal(space);
goal->as<ob::RealVectorStateSpace::StateType>()->values[0] = target_pt(0);
goal->as<ob::RealVectorStateSpace::StateType>()->values[1] = target_pt(1);
goal->as<ob::RealVectorStateSpace::StateType>()->values[2] = target_pt(2);

// Create a problem instance
ob::ProblemDefinitionPtr pdef(new ob::ProblemDefinition(si));
// Set the start and goal states
pdef->setStartAndGoalStates(start, goal);

pdef->setOptimizationObjective(getPathLengthObjective(si));
//pdef->getThresholdPathLengthObj(getPathLengthObjective(si));
// Construct our optimizing planner using the RRTstar algorithm.
ob::PlannerPtr optimizingPlanner(new og::RRTstar(si));
// Set the problem instance for our planner to solve
optimizingPlanner->setProblemDefinition(pdef);
optimizingPlanner->setup();

// attempt to solve the planning problem within one second of
// planning time
ob::PlannerStatus solved = optimizingPlanner->solve(1.0);

if (solved)
{
    // get the goal representation from the problem definition (not the same as the goal state)
    // and inquire about the found path
    og::PathGeometric* path = pdef->getSolutionPath()->as<og::PathGeometric>();

    vector<Vector3d> path_points;
    for (size_t path_idx = 0; path_idx < path->getStateCount (); path_idx++)
    {
        const ob::RealVectorStateSpace::StateType *state = path->getState(path_idx)->as<ob::RealVectorStateSpace::StateType>();
        Vector3d position;
        position[0] = state->values[0];
        position[1] = state->values[1];
        position[2] = state->values[2];
        path_points.push_back(position);
    }
    visRRTstarPath(path_points);
}
```

```
bool isValid(const ob::State* state) const
{
    // We know we're working with a RealVectorStateSpace in this
    // example, so we downcast state into the specific type.
    const ob::RealVectorStateSpace::StateType* state3D =
        state->as<ob::RealVectorStateSpace::StateType>();
    // Extract the robot's (x,y,z) position from its state
    double x = state3D->values[0];
    double y = state3D->values[1];
    double z = state3D->values[2];

    return _RRTstar_preparatory->isObsFree(x, y, z);
}
```

官方文档

http://ompl.kavrakilab.org/geometricPlanningSE3.html

高频问题1：RRT算法

如何采样：

方法1：直接在地图范围内采样x、y两个随机值作为采样点坐标；

方法2：在(0，1)范围内采样一个随机值，然后映射放大到地图范围内；

方法3：智能采样，RRT的改进变种方向之一，这里作为了解不再多说了。

如何找x_near：直接遍历树，求最近点

如何向树插入x_new：matlab中是通过设置节点值、设置其父节点来实现的，通常树的数据结构实现中会有插入操作的，无需重建树。

高频问题2：kd tree

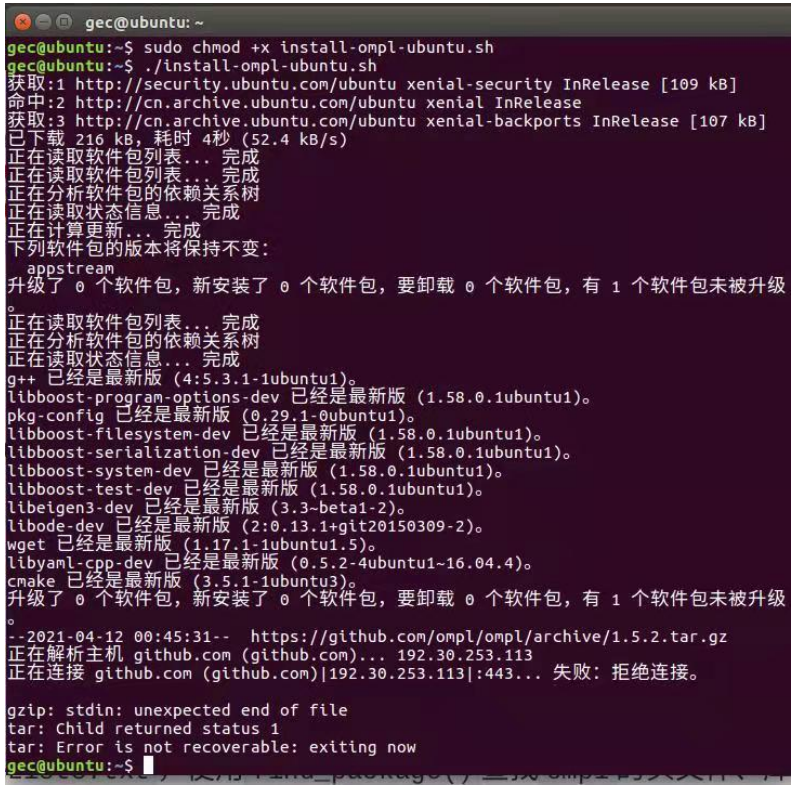
回想RRT算法步骤，时间复杂度比较高的地方在哪一步呢？如何优化？
没错就是找x_near这一步，需要遍历整个树的节点去寻找与采样点最近的节点，单次O(n)

Kd 树的作用：多维的平衡二叉查找树，咱们这里是二维的，单次时间复杂度O(logn)

Kd 树的实现：涉及较深的数据结构与算法，这里不展开讲，留给大家拓展。

Kd 树的开销：额外的空间去建立kd树，插入新节点时除了设置父节点还需旋转树以维护查找树的有序结构。

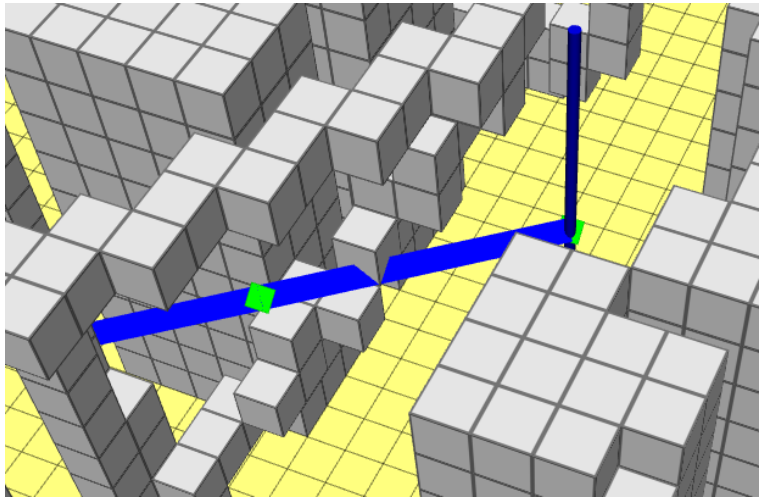
高频问题3：OMPL安装出错



一般都是网络问题导致安装失败，可尝试以下方法：

1.通过手机热点、挂vpn等方式解决下载问题。(推荐)
2.自己找源码手动编译安装，规避下载问题。
3.利用sudo apt-get install ros-kinetic-ompl命令安装(推荐)

高频问题4：路径与障碍物干涉



路径穿过障碍物

路径是由OMPL输出的路径点连接后显示在rviz上的，点间连线穿过障碍物是可能的，正常情况。

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening