深蓝学院
shenlanxueyuan.com

移动机器人运动规划
——第三章作业讲解

主讲人 武庆斌

作业目录

深蓝学院
shenlanxueyuan.com

- MATLAB作业
  - RRT算法
  - 拓展—RRT*算法

- ROS作业
  - RRT*算法

MATLAB作业-RRT算法

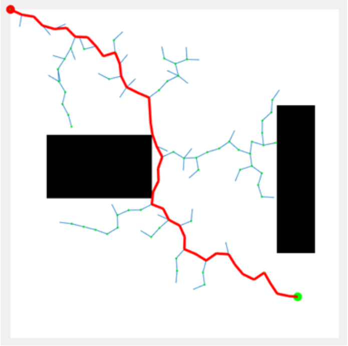
RRT算法原理与实现

Step 1: 在地图上随机采样一个点x_rand
Step 2: 遍历树， 从树中寻找最邻近点x_near
Step 3: 扩展得到x_new节点， 并检查是否碰撞
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之间的路劲画出来

实现

```
for iter = 1:3000
    %x_rand=[];
    %Step 1: 在地图中随机采样一个点x_rand
    %提示: 用 (x_rand(1),x_rand(2)) 表示环境中采样点的坐标
    x_rand=Sample(Imp,goal);
    %x_near=[];
    %Step 2: 遍历树, 从树中找到最近邻近点x_near
    %提示: x_near已经在树T里
    [x_near,index]= Near(x_rand,T);
    plot(x_near(1), x_near(2), 'go', 'MarkerSize',2);

    %x_new=[];
    %Step 3: 扩展得到x_new节点
    %提示: 注意使用扩展步长Delta
    x_new=Steer(x_rand,x_near,Delta);
    %检查节点是否是collision-free
    if ~collisionChecking(x_near,x_new,Imp) %是障碍物函数返回false;
        continue;
    end
    count=count+1;
   %Step 4: 将x_new插入树T
   %提示: 新节点x_new的父节点是x_near
   T.v(count).x = x_new(1);
   T.v(count).y = x_new(2);
   T.v(count).xPrev = x_near(1);% 父节点的坐标:起始节点的父节点仍然是其本身
   T.v(count).yPrev = x_near(2);
   T.v(count).dist=Distance(x_new,x_near);%从父节点到该节点的距离，这里可取欧氏距离
   T.v(count).indPrev = index;     %父节点的索引
   %Step 5:检查是否到达目标点附近
   %提示: 注意使用目标点阈值Thr，若当前节点和终点的欧式距离小于Thr，则跳出当前for循环
   if Distance(x_new,goal) < Thr
       break;
   end
   %Step 6:将x_near和x_new之间的路径画出来
   %提示 1: 使用plot绘制，因为要多次在同一张图上绘制线段，所以每次使用plot后需要接上hold on命令
   %提示 2: 在判断终点条件弹出for循环前，记得把x_near和x_new之间的路径画出来
   line([x_near(1),x_new(1)],[x_near(2),x_new(2)]);

  pause(0.1); %暂停0.1s，使得RRT扩展过程容易观察
end
```

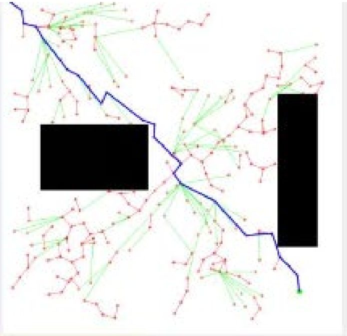
结果



MATLAB作业-RRT*算法

```matlab
%%=========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).dis
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;
```

```matlab
%%=========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);
```



- STEP1:查找
- STEP2:剪枝

ROS作业-RRT*算法

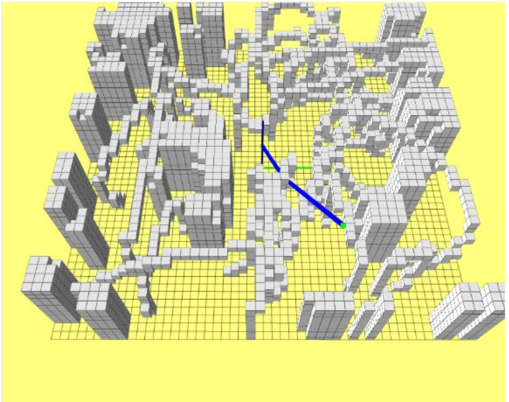
```
// Set our robot's starting state to be the bottom-left corner of
// the environment, or (0,0).
ob::ScopedState<> start(space);
start->as<ob::RealVectorStateSpace::StateType>()->values[0] = start_pt(0);
start->as<ob::RealVectorStateSpace::StateType>()->values[1] = start_pt(1);
start->as<ob::RealVectorStateSpace::StateType>()->values[2]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

官方文档详见

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

结果



深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening