RRF 主函数

map=im2bw(imread('map1.bmp')); % 从 bmp 文件中读取输入地图。若要使用新地图，请在此处写入文件名

这些行设置了 RRT 算法的参数，包括地图、源和目标点、步长、距离阈值、最大失败尝试次数和是否显示生长过程。然后，它开始一个循环，用于在地图上生长 RRT。

source=[10 10]; % 源位置在 Y, X 格式

goal=[490 490]; % 目标位置在 Y, X 格式

stepsize=20; % 每次 RRT 扩展的大小

disTh=20; % 距离阈值，用于检查新节点是否接近现有节点

maxFailedAttempts = 10000; % 最大失败尝试次数

display=true; % 是否显示 RRT 生长过程

%%%%% 参数结束 %%%%%

```
tic;
if ~feasiblePoint(source,map), error('source lies on an obstacle or outside map'); end
if ~feasiblePoint(goal,map), error('goal lies on an obstacle or outside map'); end
if display, imshow(map);rectangle('position',[1 1 size(map)-1],'edgecolor','k'); end
RRTree=double([source -1]); % RRT 以源节点为根，包含表示节点和父索引
failedAttempts=0;
counter=0;
pathFound=false;
```

这些行在循环中生成 RRT。首先，它随机选择一个样本点，然后找到与该样本点最近的树节点。它计算从最近节点到样本点的方向，并使用这个方向和步长来生成一个新的节点。如果新节点不可行，它增加失败尝试次数并继续下一次循环。如果新节点到目标点的距离小于阈值，则认为找到了路径并退出循环。如果新节点不在树中，它将其添加到树中。

```
while failedAttempts<=maxFailedAttempts  % 循环以生长 RRT
    if rand < 0.5,
        sample=rand(1,2) .* size(map); % 随机样本
    else
        sample=goal; % 样本取为目标点，以偏置树向目标生长
    end
    [A, I]=min( distanceCost(RRTree(:,1:2),sample) ,[],1); % 找到与样本最近的节点
    closestNode = RRTree(I(1),1:2);
    theta=atan2(sample(1)-closestNode(1),sample(2)-closestNode(2)); % 方向，用于扩展最近节点到样本
    newPoint = double(int32(closestNode(1:2) + stepsize * [sin(theta) cos(theta)]));
    if ~checkPath(closestNode(1:2), newPoint, map) % 如果最近节点的扩展到新点不可行
        failedAttempts=failedAttempts+1;
        continue;
    end
    if distanceCost(newPoint,goal)<disTh, pathFound=true;break; end % 如果新点到目标点的距离小于阈值，则找到路径
    [A, I2]=min( distanceCost(RRTree(:,1:2),newPoint) ,[],1); % 检查新节点是否已存在于树中
    if                distanceCost(newPoint,RRTree(I2(1),1:2))<disTh, failedAttempts=failedAttempts+1;continue; end
    RRTree=[RRTree;newPoint I(1)]; % 添加新节点
    failedAttempts=0;
    if display,
    line([closestNode(2);newPoint(2)],[closestNode(1);newPoint(1)]);
        counter=counter+1;M(counter)=getframe;
```

```
    end
End
```

这些行在找到路径后，绘制从最近节点到目标点的路径，并将当前帧添加到动画序列中。

```
    if display && pathFound
        line([closestNode(2);goal(2)],[closestNode(1);goal(1)]);
        counter=counter+1;M(counter)=getframe;
    end
End
```

这行代码在显示动画时等待用户点击或按键来结束显示。

```
if display
    disp('click/press any key');
    waitforbuttonpress;
end
if ~pathFound, error('no path found. maximum attempts reached'); end
```

这行代码检查是否找到了路径。如果没有找到路径，并且已经达到最大失败尝试次数，则抛出一个错误。

这行代码从目标点开始，逆向遍历树，以构建从目标点到源点的路径。

```
path=[goal];
prev=I(1);
while prev>0
    path=[RRTree(prev,1:2);path];
    prev=RRTree(prev,3);
end
```

这行代码计算路径的长度。它遍历路径中的每个相邻节点对，并使用 distanceCost 函数计算它们之间的距离，然后累加这些距离来得到路径的总长度。

```
pathLength=0;
for i=1:length(path)-1,
```

```
    pathLength=pathLength+distanceCost(path(i,1:2),path(i+1,1:2));
end
```

这行代码使用 fprintf 函数在控制台输出处理时间和路径长度。它使用 toc 函数计算从 tic 开始的处理时间，并将这两个值格式化输出。

```
    fprintf('processing time=%d \nPath Length=%d \n\n',
toc,pathLength);
```

这行代码显示原始地图，并在地图上绘制路径。

```
    imshow(map);rectangle('position',[1 1
size(map)-1],'edgecolor','k');
line(path(:,2),path(:,1));
```