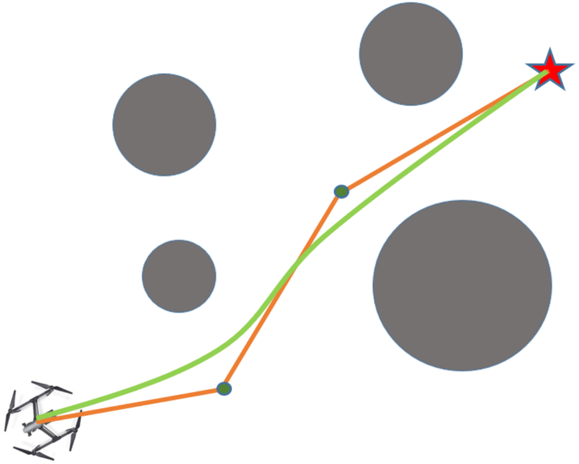
基于采样的路径规划

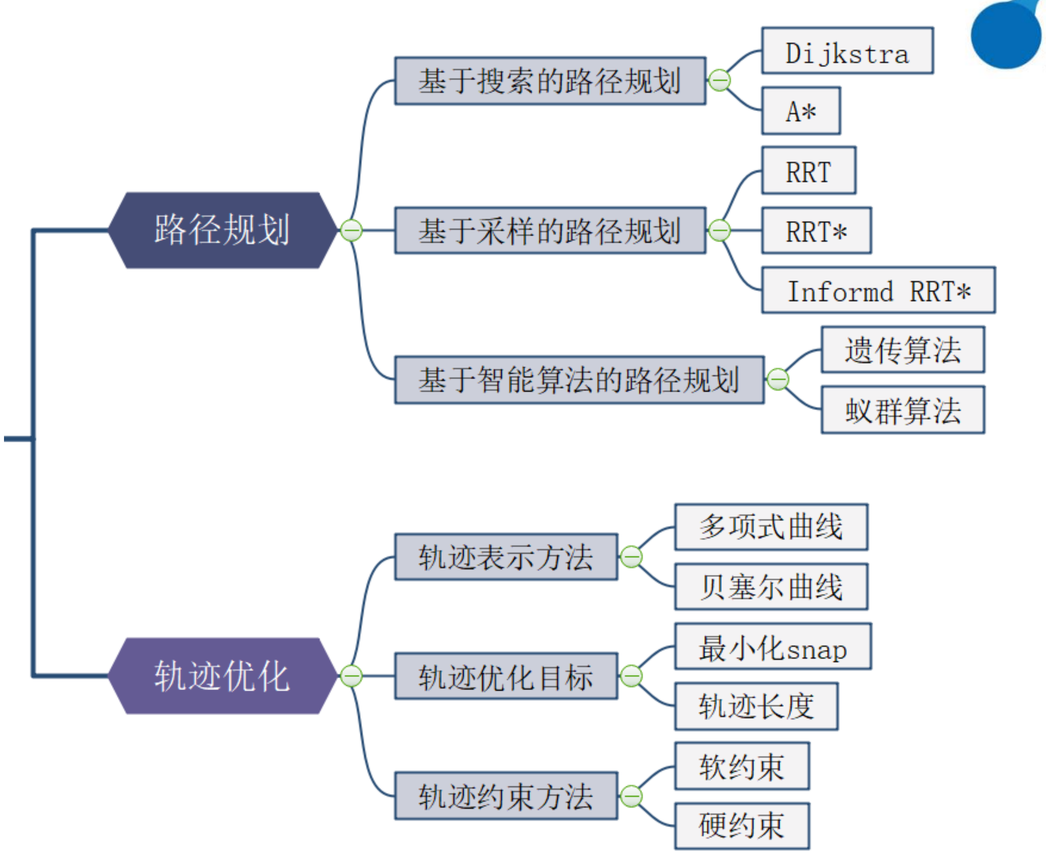
Joe　　艾若机器人　　joe_ir@163.com

公众号：Joe学习笔记

课程大纲



路径规划 ＋ 轨迹优化



- 路径规划
  - 基于搜索的路径规划
    - Dijkstra
    - A*
  - 基于采样的路径规划
    - RRT
    - RRT*
    - Informd RRT*
  - 基于智能算法的路径规划
    - 遗传算法
    - 蚁群算法
- 轨迹优化
  - 轨迹表示方法
    - 多项式曲线
    - 贝塞尔曲线
  - 轨迹优化目标
    - 最小化snap
    - 轨迹长度
  - 轨迹约束方法
    - 软约束
    - 硬约束

机器人实验：　在基于ROS的机器人平台上实现算法

联系方式

关注公众号：Joe学习笔记，获取PPT和代码

邮箱：joe_ir@163.com

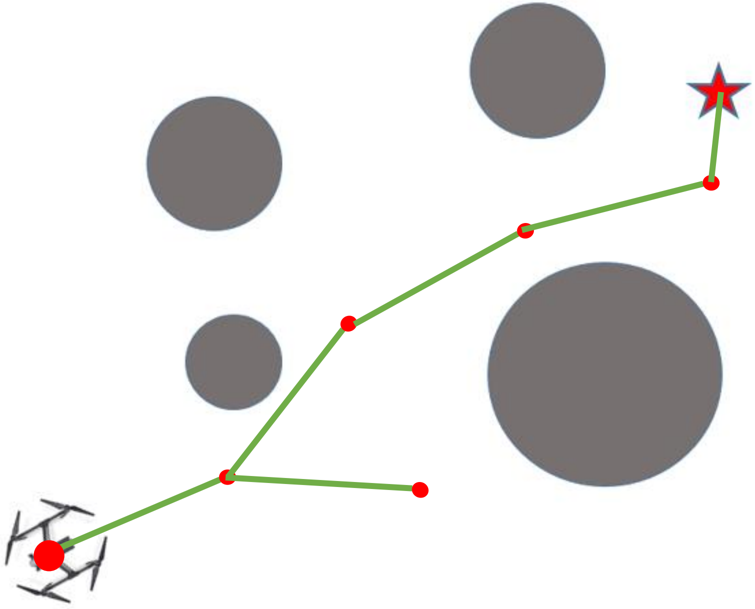
01

PART

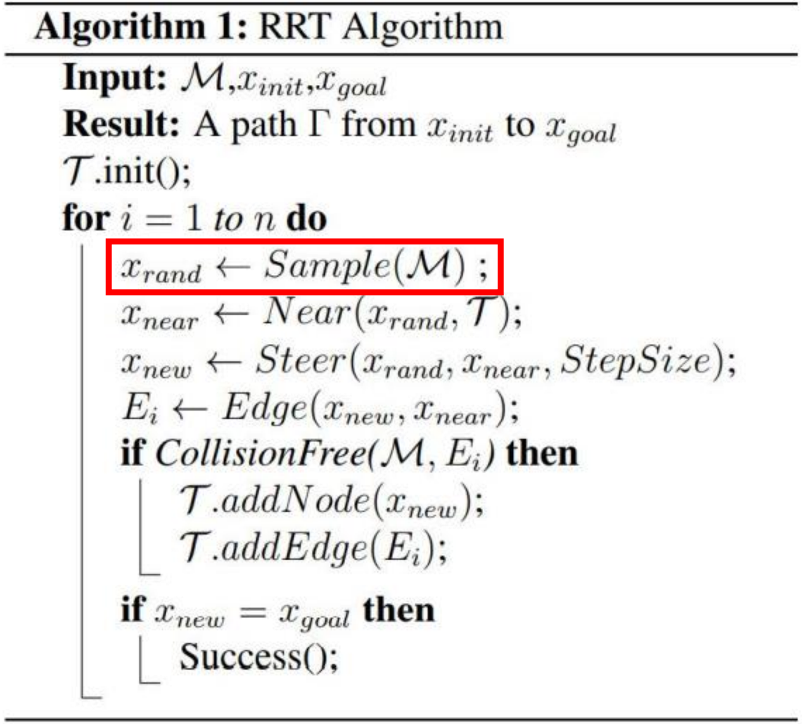
RRT算法原理讲解

RRT算法

快速扩展随机树算法（Rapidly Exploring Random Tree）

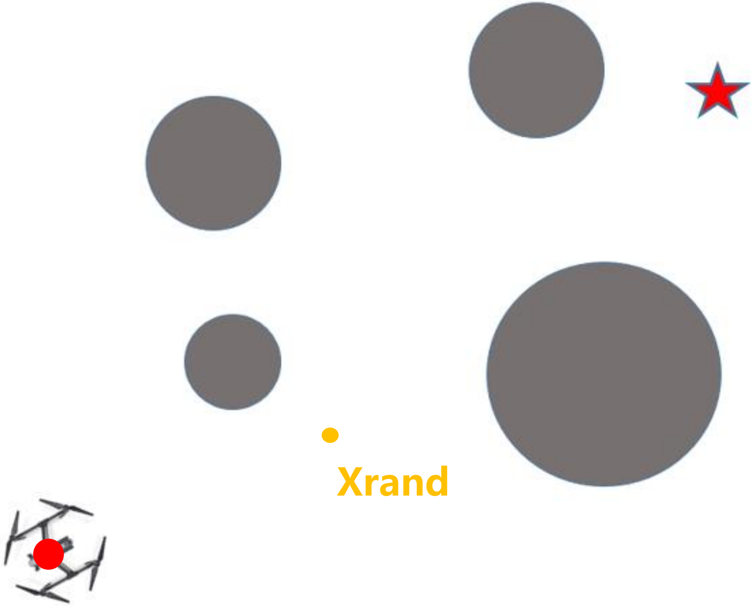


RRT算法

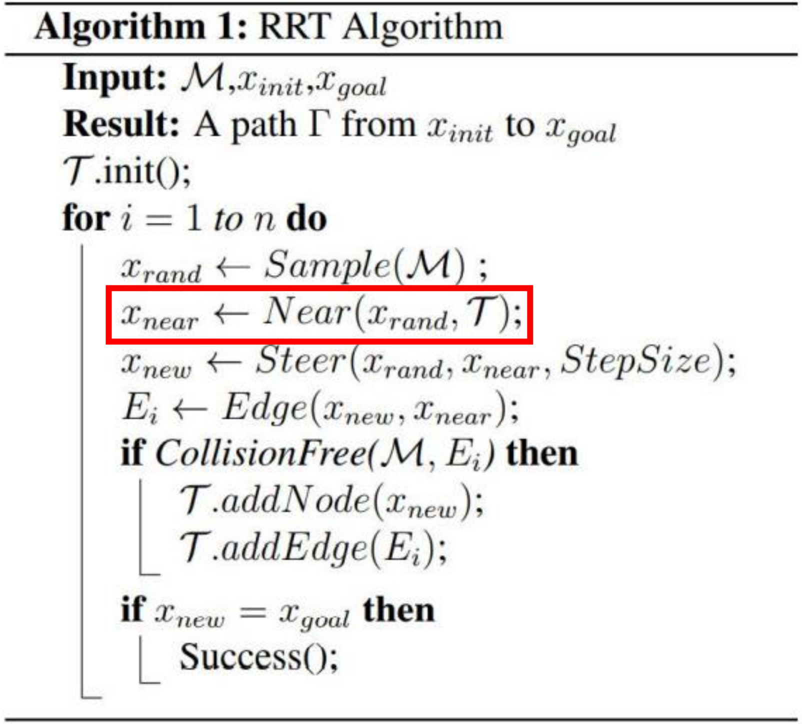


Algorithm 1: RRT Algorithm

```
Input: $\mathcal{M}$,$x_{init}$,$x_{goal}$
Result: A path $\Gamma$ from $x_{init}$ to $x_{goal}$
$\mathcal{T}$.init();
for $i = 1$ to $n$ do
    $\boxed{x_{rand} \leftarrow Sample(\mathcal{M})\ ;}$
    $x_{near} \leftarrow Near(x_{rand}, \mathcal{T})$;
    $x_{new} \leftarrow Steer(x_{rand}, x_{near}, StepSize)$;
    $E_i \leftarrow Edge(x_{new}, x_{near})$;
    if $CollisionFree(\mathcal{M}, E_i)$ then
        $\mathcal{T}.addNode(x_{new})$;
        $\mathcal{T}.addEdge(E_i)$;
    if $x_{new} = x_{goal}$ then
        Success();
```

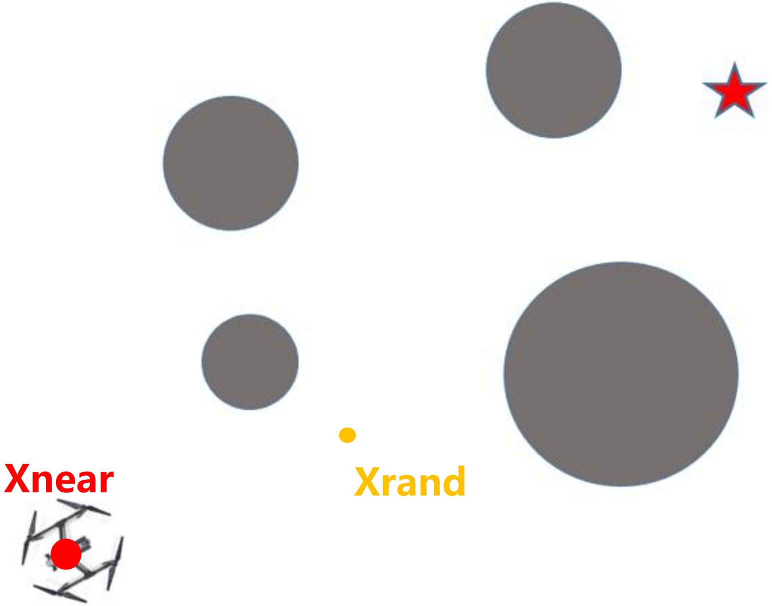


RRT算法

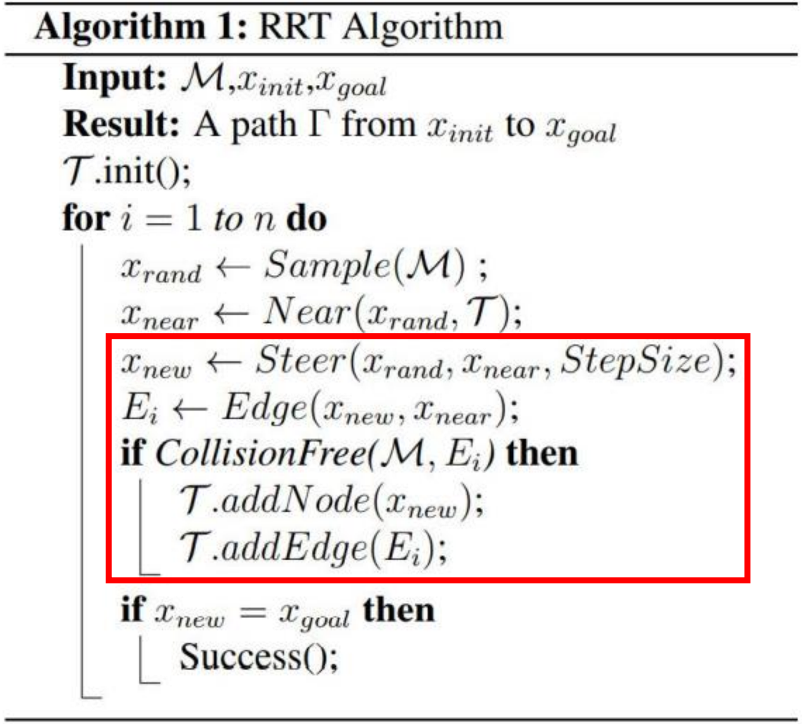


Algorithm 1: RRT Algorithm

```
Input: $\mathcal{M}$,$x_{init}$,$x_{goal}$
Result: A path $\Gamma$ from $x_{init}$ to $x_{goal}$
$\mathcal{T}$.init();
for $i = 1$ to $n$ do
    $x_{rand} \leftarrow Sample(\mathcal{M})$ ;
    $x_{near} \leftarrow Near(x_{rand}, \mathcal{T})$;
    $x_{new} \leftarrow Steer(x_{rand}, x_{near}, StepSize)$;
    $E_i \leftarrow Edge(x_{new}, x_{near})$;
    if $CollisionFree(\mathcal{M}, E_i)$ then
        $\mathcal{T}.addNode(x_{new})$;
        $\mathcal{T}.addEdge(E_i)$;
    if $x_{new} = x_{goal}$ then
        Success();
```

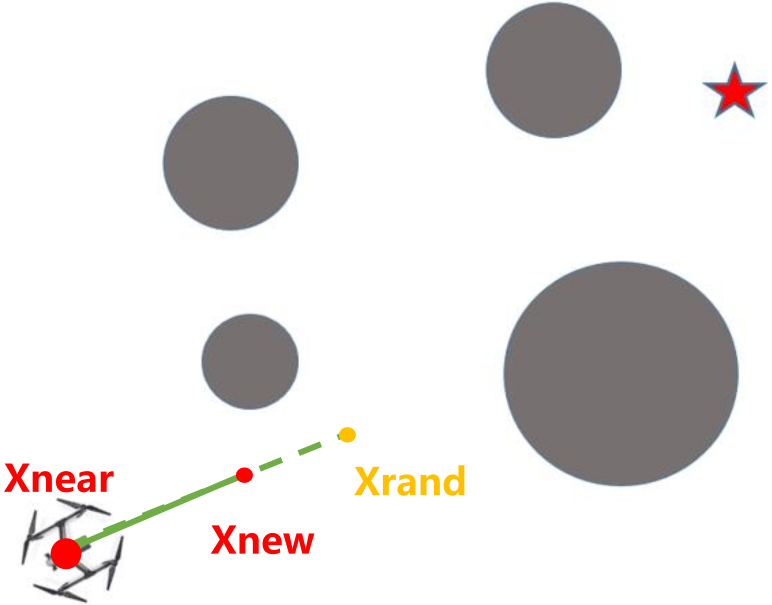


RRT算法

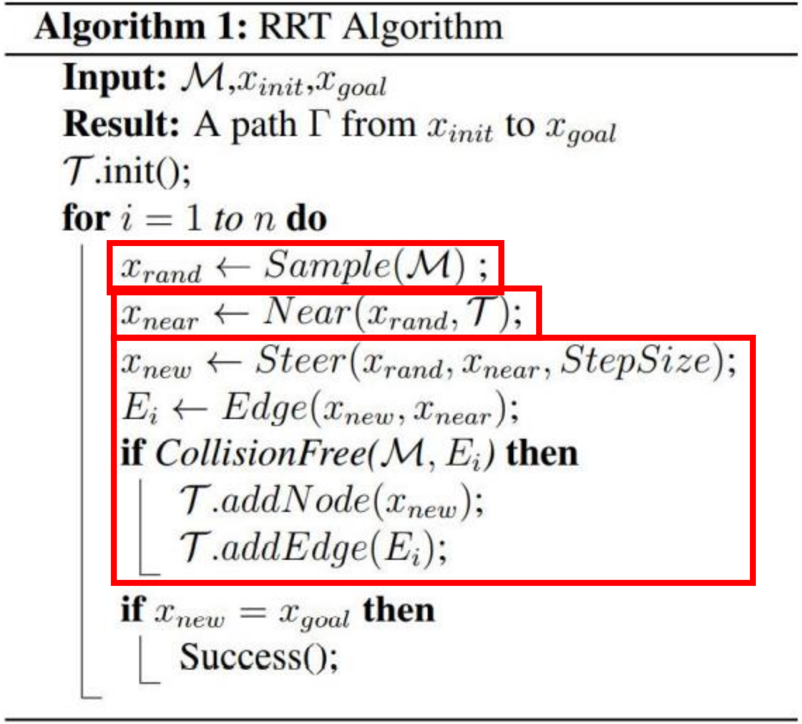


Algorithm 1: RRT Algorithm

```
Input: $\mathcal{M}$,$x_{init}$,$x_{goal}$
Result: A path $\Gamma$ from $x_{init}$ to $x_{goal}$
$\mathcal{T}$.init();
for $i = 1$ to $n$ do
    $x_{rand} \leftarrow Sample(\mathcal{M})$ ;
    $x_{near} \leftarrow Near(x_{rand}, \mathcal{T})$;
    $x_{new} \leftarrow Steer(x_{rand}, x_{near}, StepSize)$;
    $E_i \leftarrow Edge(x_{new}, x_{near})$;
    if $CollisionFree(\mathcal{M}, E_i)$ then
        $\mathcal{T}.addNode(x_{new})$;
        $\mathcal{T}.addEdge(E_i)$;
    if $x_{new} = x_{goal}$ then
        Success();
```

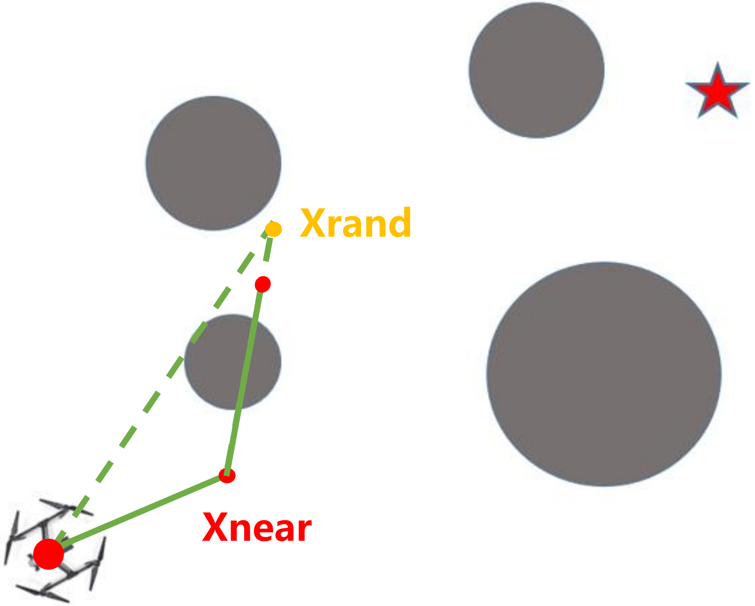


RRT算法

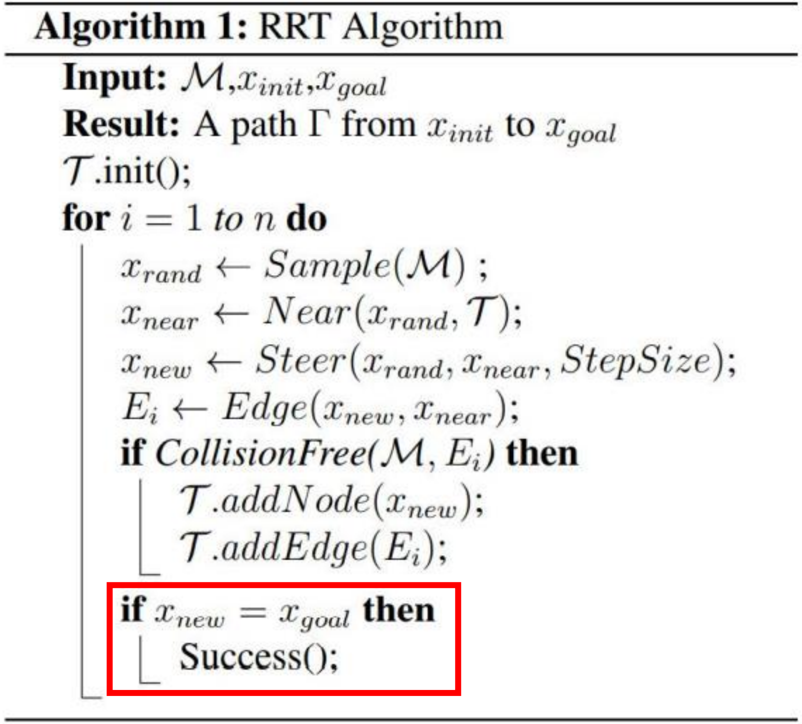


Algorithm 1: RRT Algorithm

```
Input: $\mathcal{M}$,$x_{init}$,$x_{goal}$
Result: A path $\Gamma$ from $x_{init}$ to $x_{goal}$
$\mathcal{T}$.init();
for $i = 1$ to $n$ do
    $x_{rand} \leftarrow Sample(\mathcal{M})$ ;
    $x_{near} \leftarrow Near(x_{rand}, \mathcal{T})$;
    $x_{new} \leftarrow Steer(x_{rand}, x_{near}, StepSize)$;
    $E_i \leftarrow Edge(x_{new}, x_{near})$;
    if $CollisionFree(\mathcal{M}, E_i)$ then
        $\mathcal{T}.addNode(x_{new})$;
        $\mathcal{T}.addEdge(E_i)$;
    if $x_{new} = x_{goal}$ then
        Success();
```

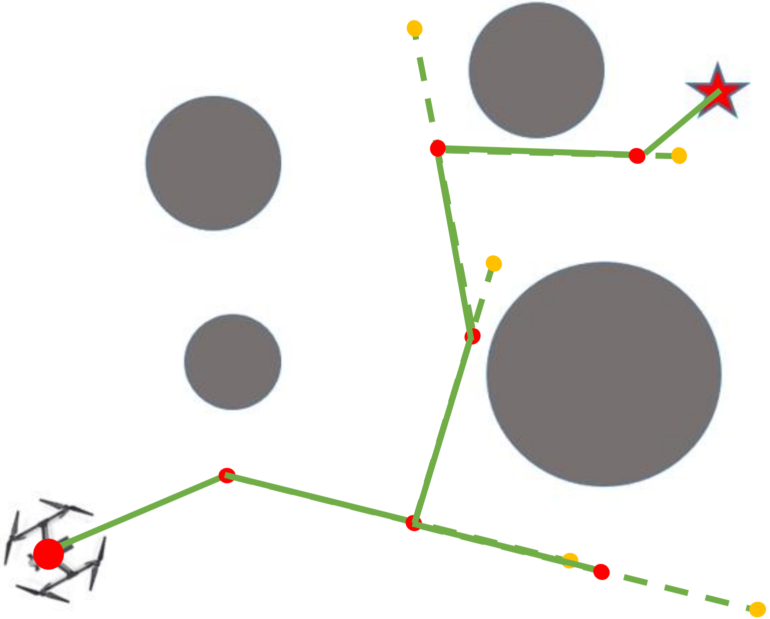


RRT算法

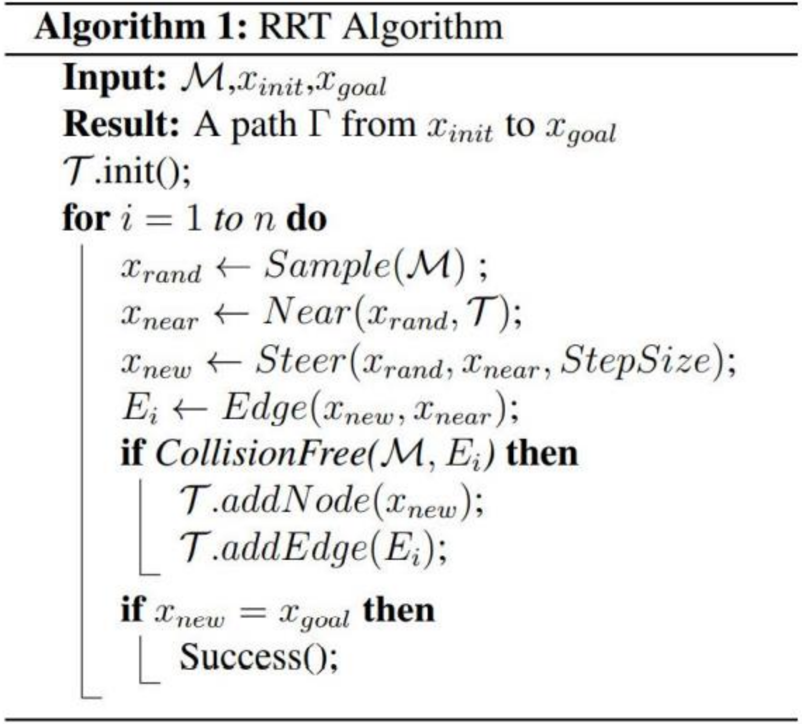


Algorithm 1: RRT Algorithm

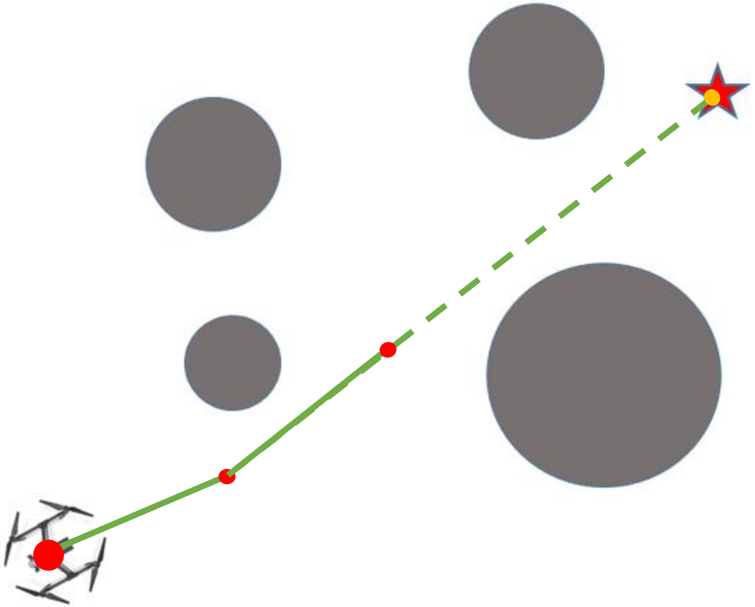
```
Input: $\mathcal{M}$,$x_{init}$,$x_{goal}$
Result: A path $\Gamma$ from $x_{init}$ to $x_{goal}$
$\mathcal{T}$.init();
for $i = 1$ to $n$ do
    $x_{rand} \leftarrow Sample(\mathcal{M})$ ;
    $x_{near} \leftarrow Near(x_{rand}, \mathcal{T})$;
    $x_{new} \leftarrow Steer(x_{rand}, x_{near}, StepSize)$;
    $E_i \leftarrow Edge(x_{new}, x_{near})$;
    if $CollisionFree(\mathcal{M}, E_i)$ then
        $\mathcal{T}.addNode(x_{new})$;
        $\mathcal{T}.addEdge(E_i)$;
    if $x_{new} = x_{goal}$ then
        Success();
```



RRT算法



```
Algorithm 1: RRT Algorithm
  Input: $\mathcal{M}$,$x_{init}$,$x_{goal}$
  Result: A path $\Gamma$ from $x_{init}$ to $x_{goal}$
  $\mathcal{T}$.init();
  for $i = 1$ to $n$ do
      $x_{rand} \leftarrow Sample(\mathcal{M})$ ;
      $x_{near} \leftarrow Near(x_{rand}, \mathcal{T})$;
      $x_{new} \leftarrow Steer(x_{rand}, x_{near}, StepSize)$;
      $E_i \leftarrow Edge(x_{new}, x_{near})$;
      if $CollisionFree(\mathcal{M}, E_i)$ then
          $\mathcal{T}.addNode(x_{new})$;
          $\mathcal{T}.addEdge(E_i)$;
      if $x_{new} = x_{goal}$ then
          Success();
```



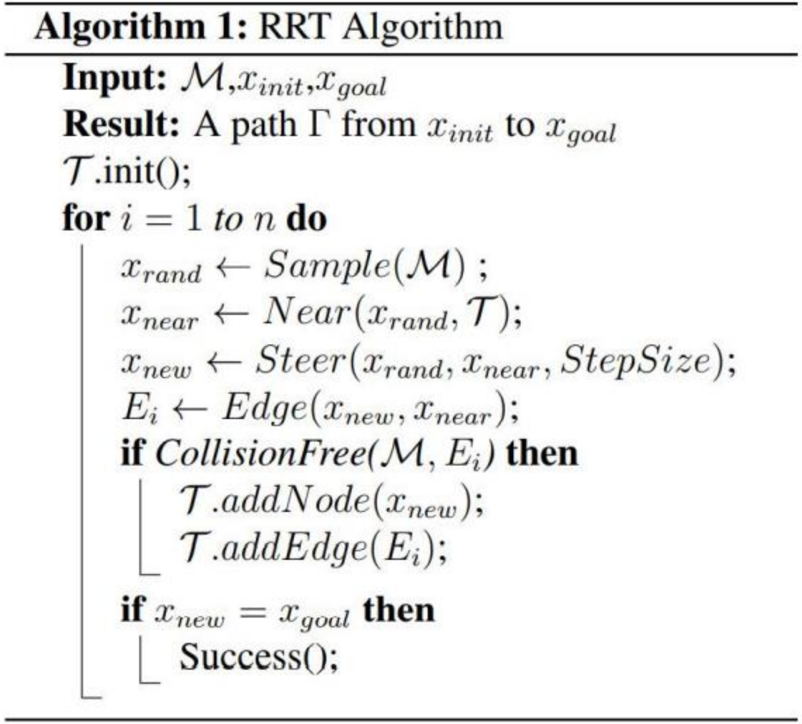
小技巧：以一定概率选择终点作为采样点

02

PART

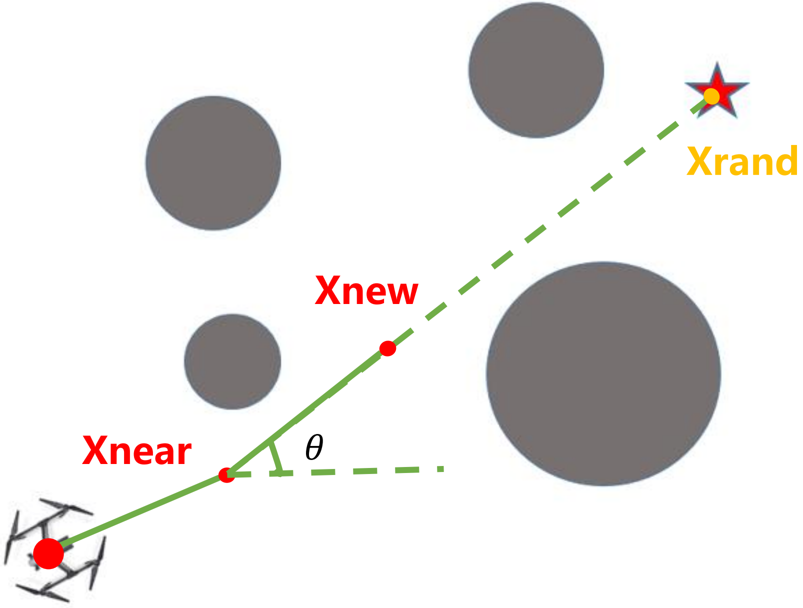
RRT算法代码讲解

RRT算法



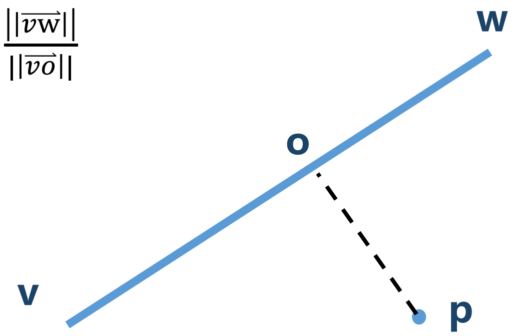
Algorithm 1: RRT Algorithm

```
Input: $\mathcal{M}$,$x_{init}$,$x_{goal}$
Result: A path $\Gamma$ from $x_{init}$ to $x_{goal}$
$\mathcal{T}$.init();
for $i = 1$ to $n$ do
    $x_{rand} \leftarrow Sample(\mathcal{M})$ ;
    $x_{near} \leftarrow Near(x_{rand}, \mathcal{T})$;
    $x_{new} \leftarrow Steer(x_{rand}, x_{near}, StepSize)$;
    $E_i \leftarrow Edge(x_{new}, x_{near})$;
    if $CollisionFree(\mathcal{M}, E_i)$ then
        $\mathcal{T}.addNode(x_{new})$;
        $\mathcal{T}.addEdge(E_i)$;
    if $x_{new} = x_{goal}$ then
        Success();
```



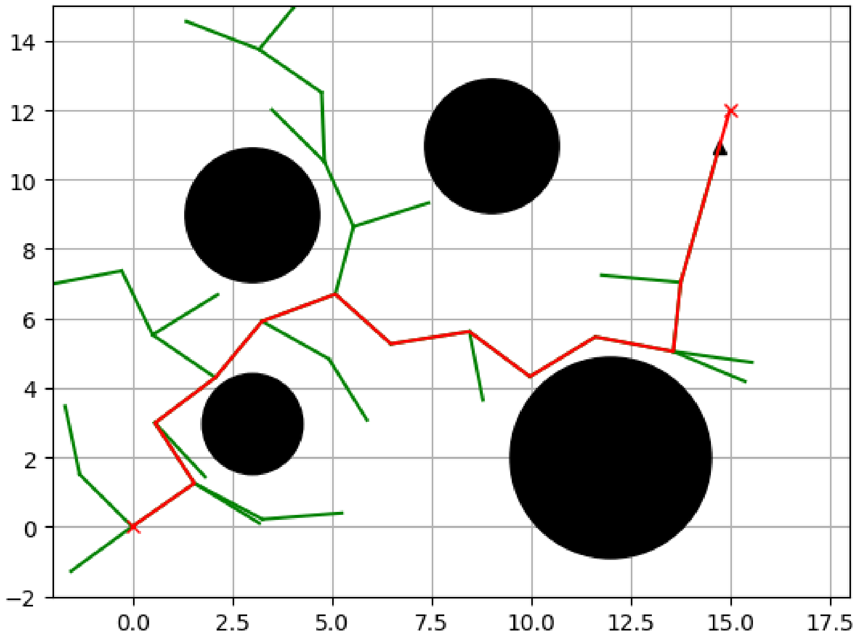
RRT算法

$$\frac{||\overrightarrow{vw}||^2}{\overrightarrow{vp}.\overrightarrow{vw}} = \frac{||\overrightarrow{vw}||^2}{||\overrightarrow{vw}||||\overrightarrow{vp}||cos\alpha} = \frac{||\overrightarrow{vw}||^2}{||\overrightarrow{vw}||||\overrightarrow{vo}||} = \frac{||\overrightarrow{\mathrm{vw}}||}{||\overrightarrow{vo}||}$$



点到直线的距离

RRT算法



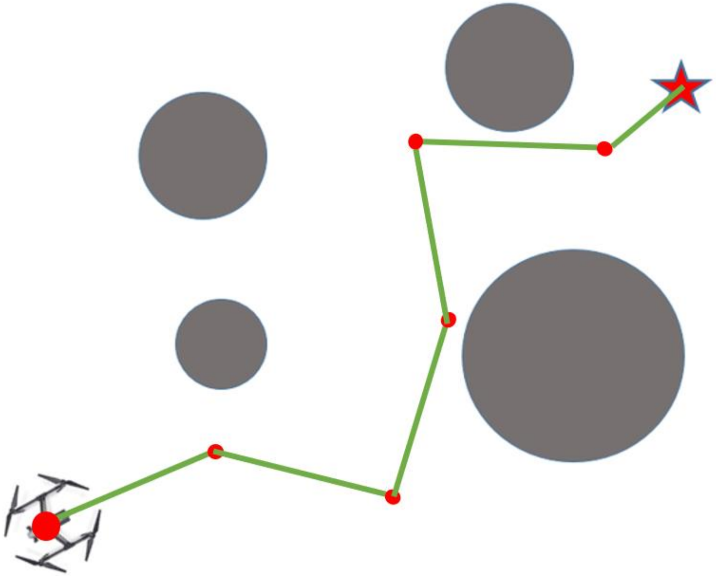
RRT算法结果

03

PART

RRT*算法原理讲解

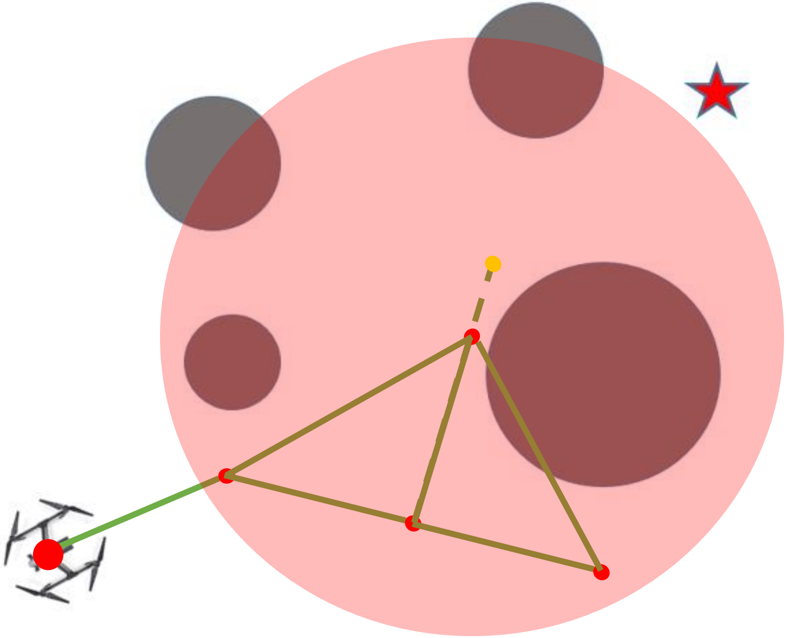
RRT*算法



RRT*算法提出的动机(motivation)：

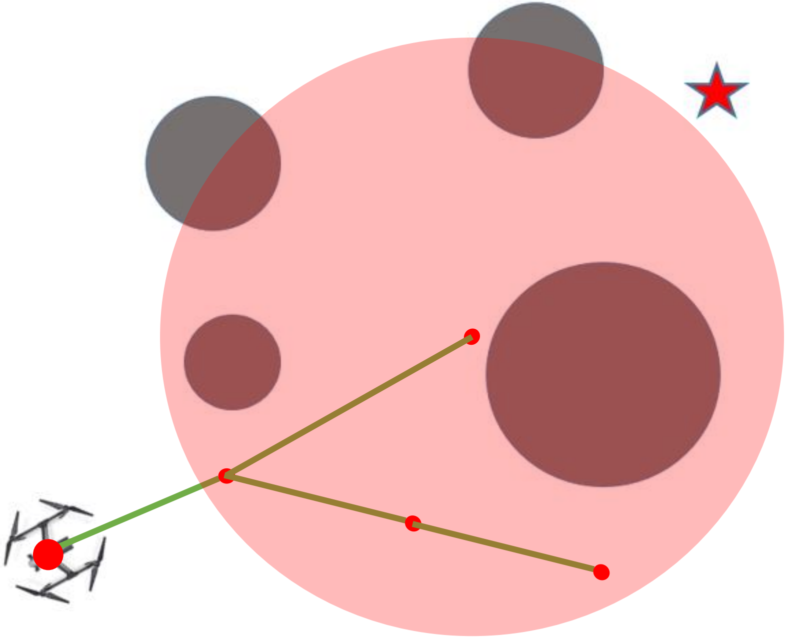
能否能找到一条最优的路径?

RRT*算法



当前节点重新选择父节点

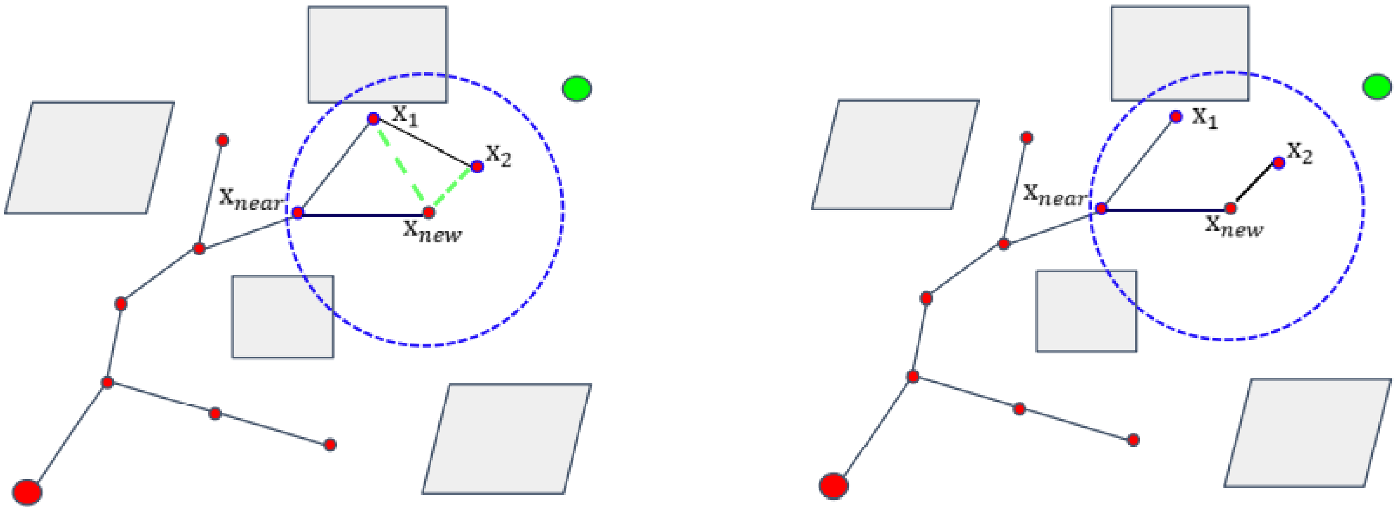
RRT*算法



范围内的节点重新连接（rewire）

RRT*算法

范围内的节点重新连接（rewire）

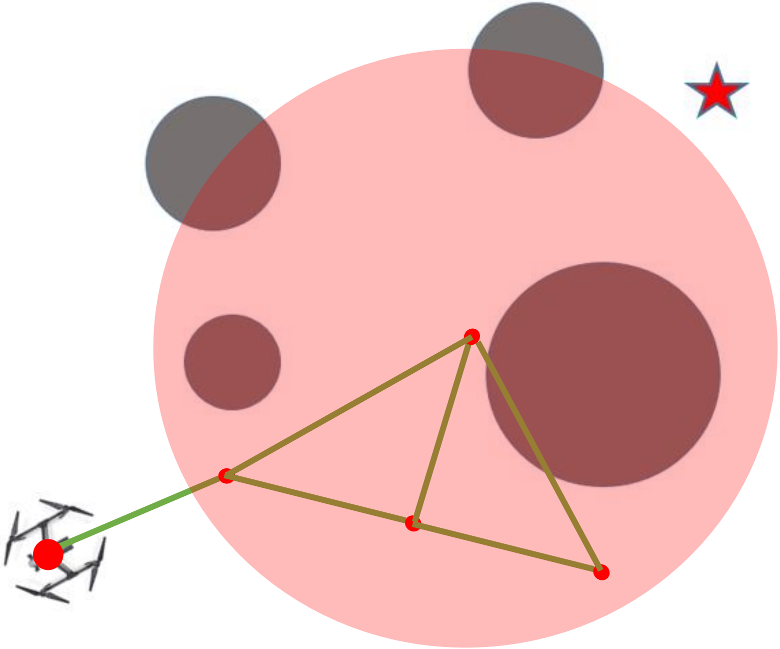


03

PART

RRT*算法代码讲解

RRT*算法



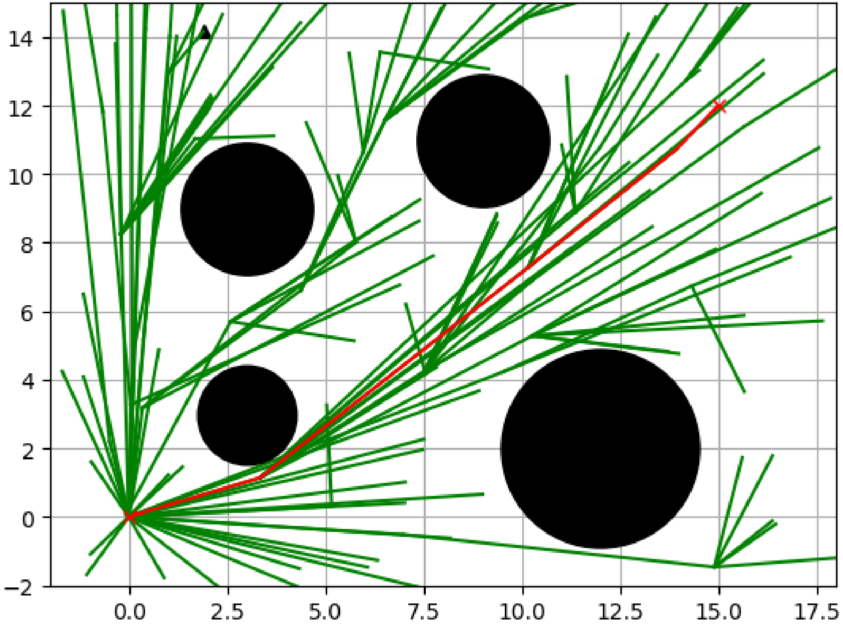
当前节点重新选择父节点

04

PART

Informed RRT*算法原理讲解

Informed RRT*算法

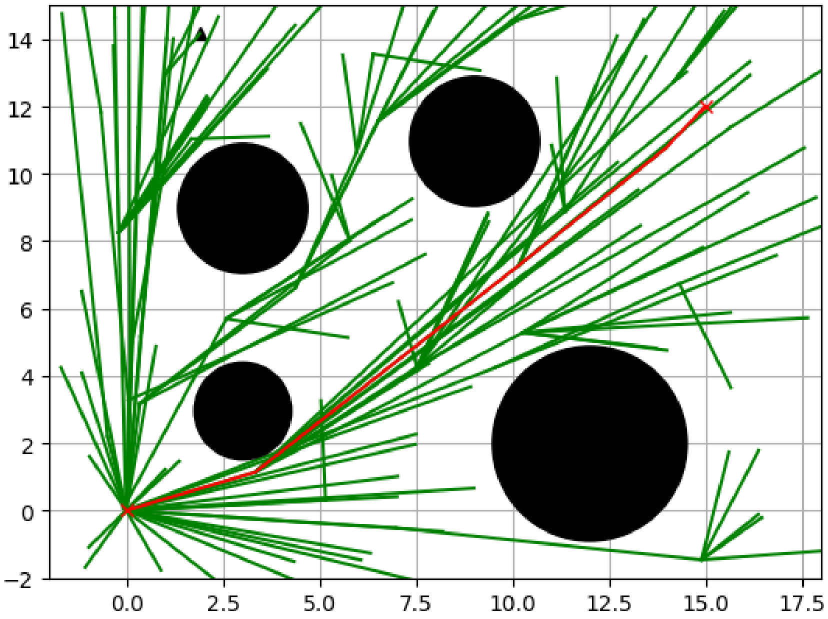


RRT*算法结果

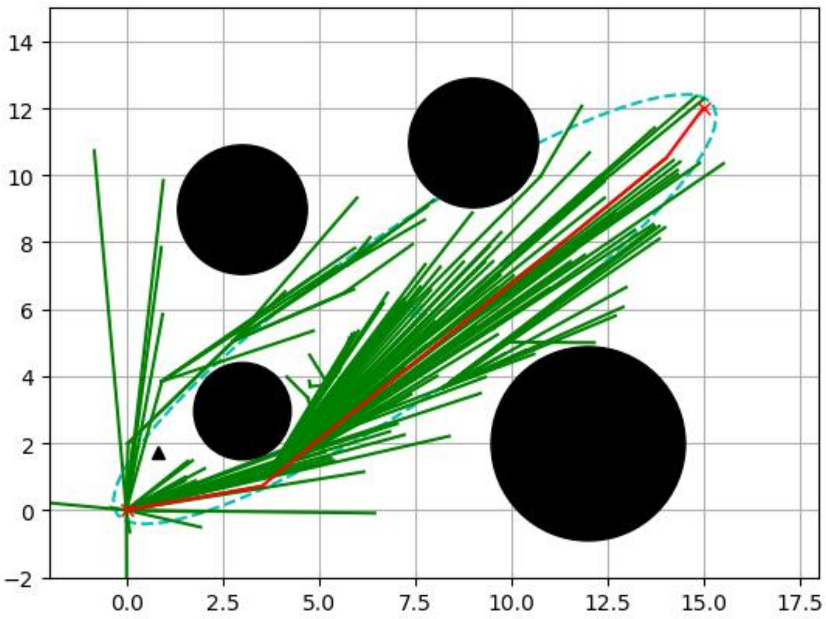
Informed RRT*算法提出的动机(motivation)：

能否增加渐近最优的速度?

Informed RRT*算法



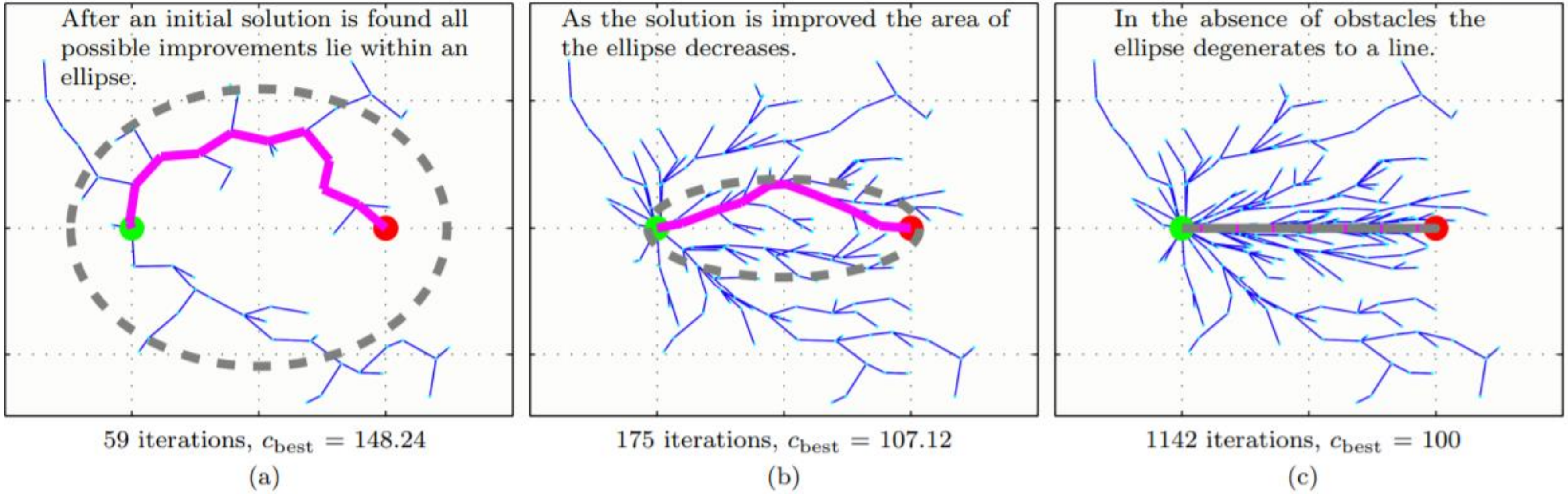
RRT*算法结果

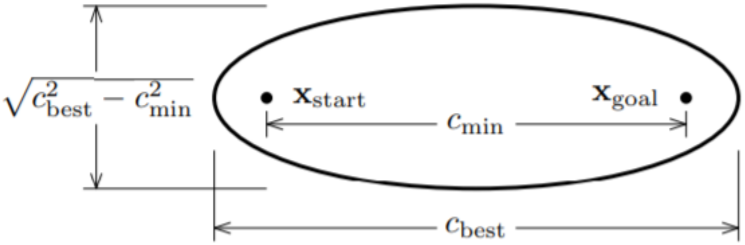


Informed RRT*算法结果

Informed RRT*: Optimal Sampling-based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic

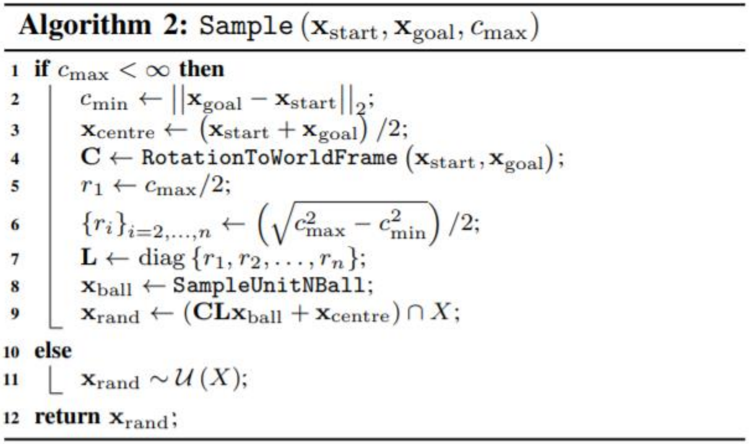
Informed RRT*算法



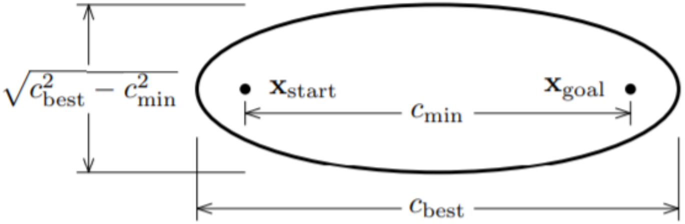


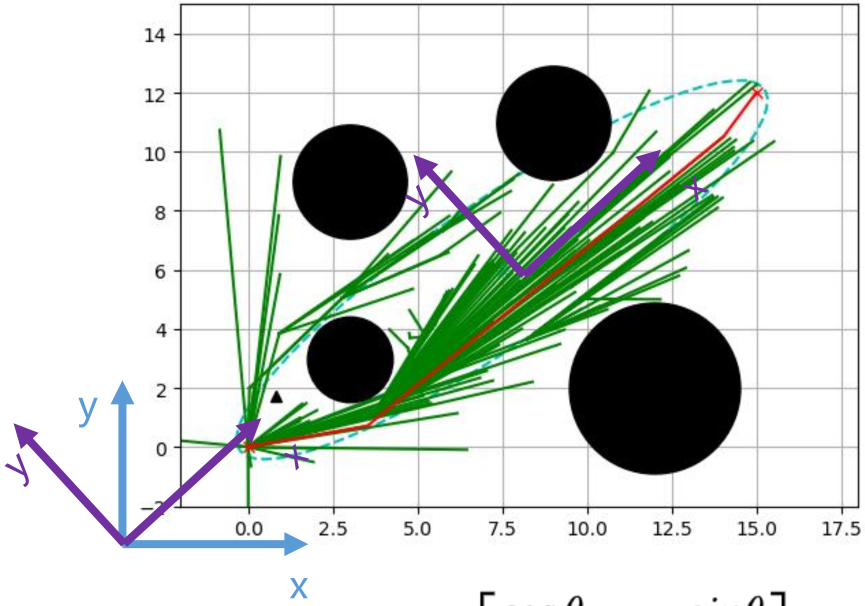
$\sqrt{c_{\text{best}}^2 - c_{\text{min}}^2}$
$\mathbf{x}_{\text{start}}$
$\mathbf{x}_{\text{goal}}$
c_{min}
c_{best}

Informed RRT*算法



```
Algorithm 2: Sample $(\mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}}, c_{\max})$
if $c_{\max} < \infty$ then
    $c_{\min} \leftarrow \|\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{start}}\|_2$;
    $\mathbf{x}_{\text{centre}} \leftarrow (\mathbf{x}_{\text{start}} + \mathbf{x}_{\text{goal}})/2$;
    $\mathbf{C} \leftarrow \texttt{RotationToWorldFrame}(\mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}})$;
    $r_1 \leftarrow c_{\max}/2$;
    $\{r_i\}_{i=2,\ldots,n} \leftarrow \left(\sqrt{c_{\max}^2 - c_{\min}^2}\right)/2$;
    $\mathbf{L} \leftarrow \text{diag}\{r_1, r_2, \ldots, r_n\}$;
    $\mathbf{x}_{\text{ball}} \leftarrow \texttt{SampleUnitNBall}$;
    $\mathbf{x}_{\text{rand}} \leftarrow (\mathbf{C}\mathbf{L}\mathbf{x}_{\text{ball}} + \mathbf{x}_{\text{centre}}) \cap X$;
else
    $\mathbf{x}_{\text{rand}} \sim \mathcal{U}(X)$;
return $\mathbf{x}_{\text{rand}}$;
```


$\sqrt{c_{\text{best}}^2 - c_{\min}^2}$
$\mathbf{x}_{\text{start}}$
$\mathbf{x}_{\text{goal}}$
$c_{\min}$
c_{best}



$$C(\theta) = \begin{bmatrix} \cos\theta & -sin\theta \\ sin\theta & \cos\theta \end{bmatrix}$$

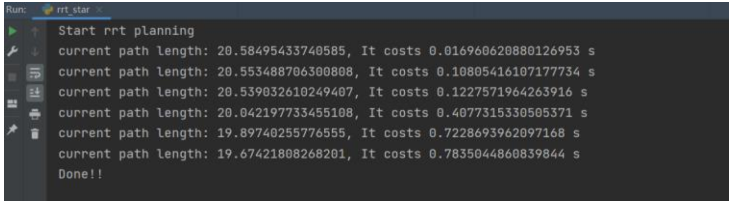
Kabsch算法求解旋转矩阵

05

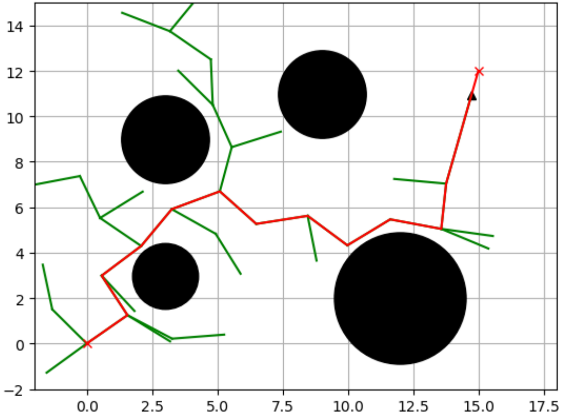
PART

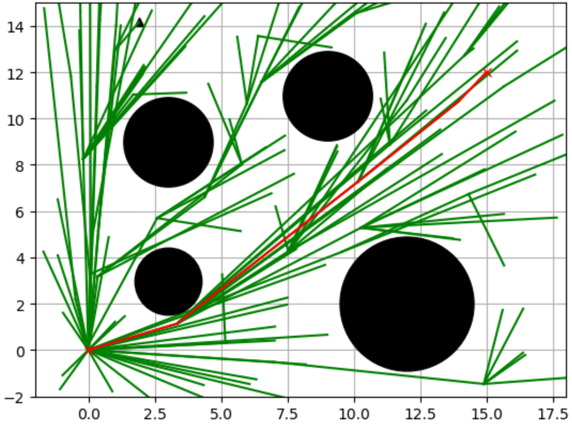
Informed RRT*算法代码讲解

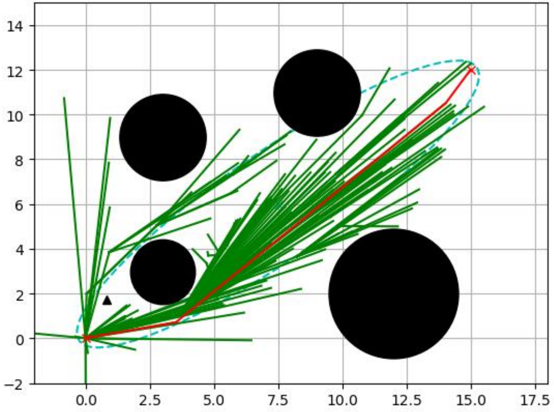
Informed RRT*算法

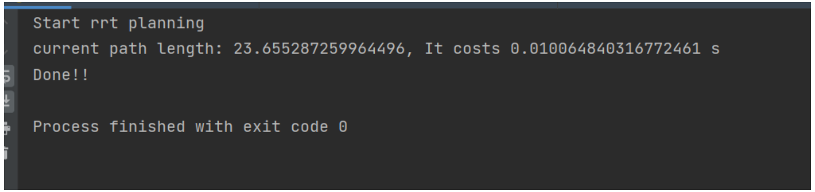


```
Start rrt planning
current path length: 20.58495433740585, It costs 0.016960620880126953 s
current path length: 20.553488706300808, It costs 0.10805416107177734 s
current path length: 20.539032610249407, It costs 0.1227571964263916 s
current path length: 20.042197733455108, It costs 0.4077315330505371 s
current path length: 19.89740255776555, It costs 0.7228693962097168 s
current path length: 19.67421808268201, It costs 0.7835044860839844 s
Done!!
```



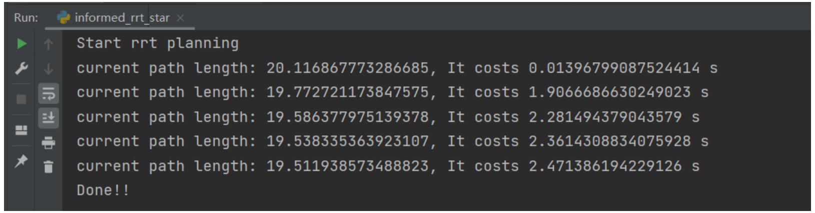






```
Start rrt planning
current path length: 23.655287259964496, It costs 0.010064840316772461 s
Done!!

Process finished with exit code 0
```



```
Start rrt planning
current path length: 20.116867773286685, It costs 0.01396799087524414 s
current path length: 19.772721173847575, It costs 1.9066686630249023 s
current path length: 19.586377975139378, It costs 2.281494379043579 s
current path length: 19.538335363923107, It costs 2.3614308834075928 s
current path length: 19.511938573488823, It costs 2.471386194229126 s
Done!!
```

THANKS