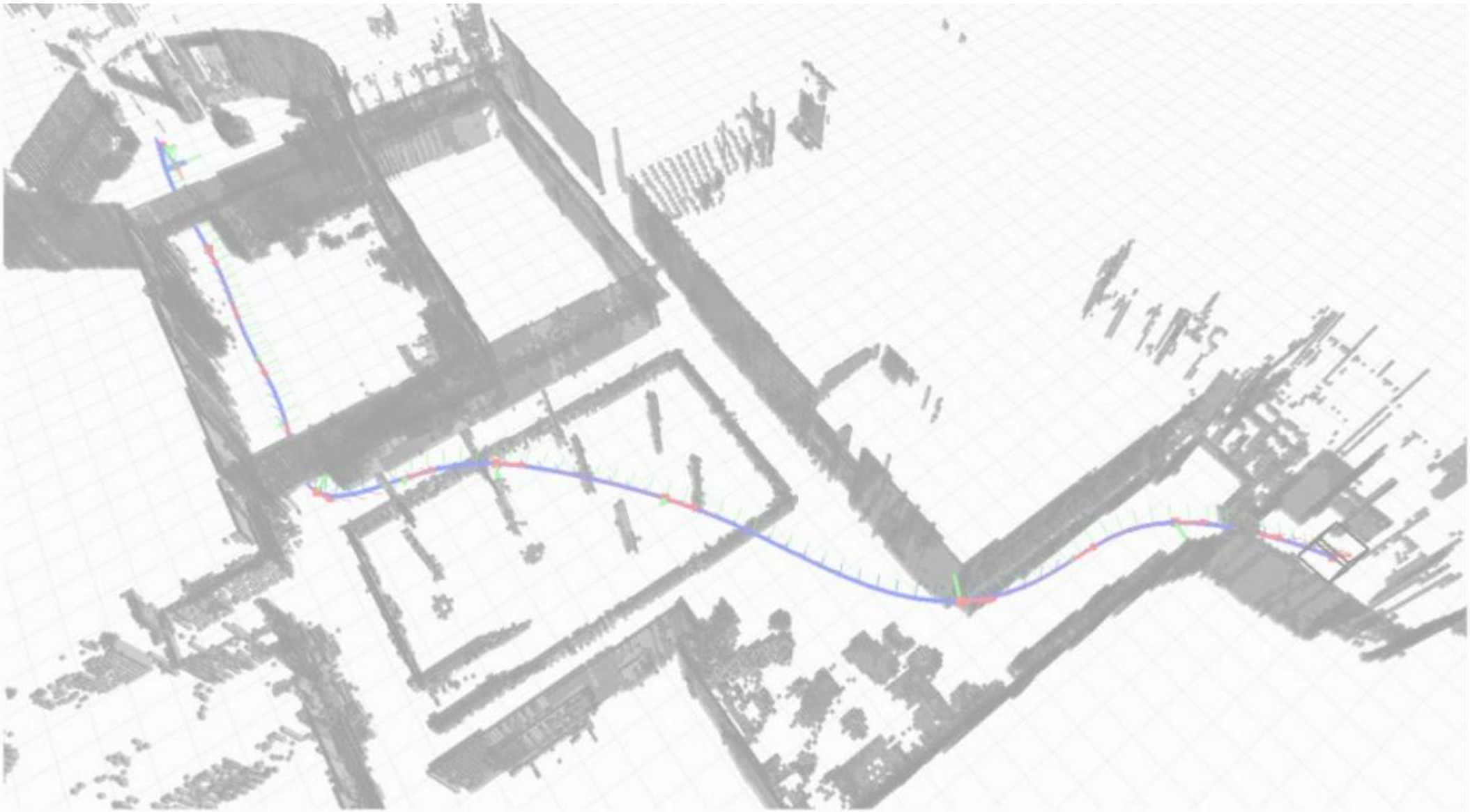
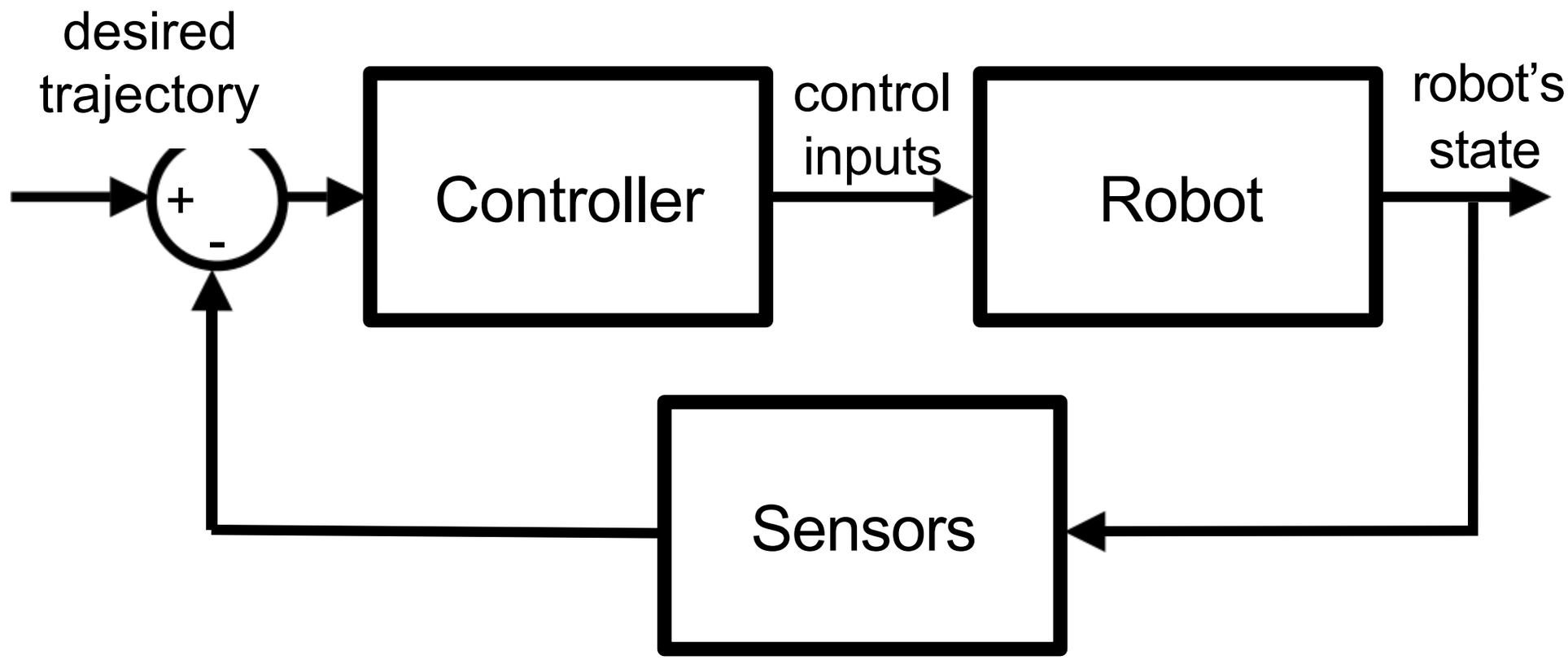


16.485: VNAV - Visual Navigation for Autonomous Vehicles

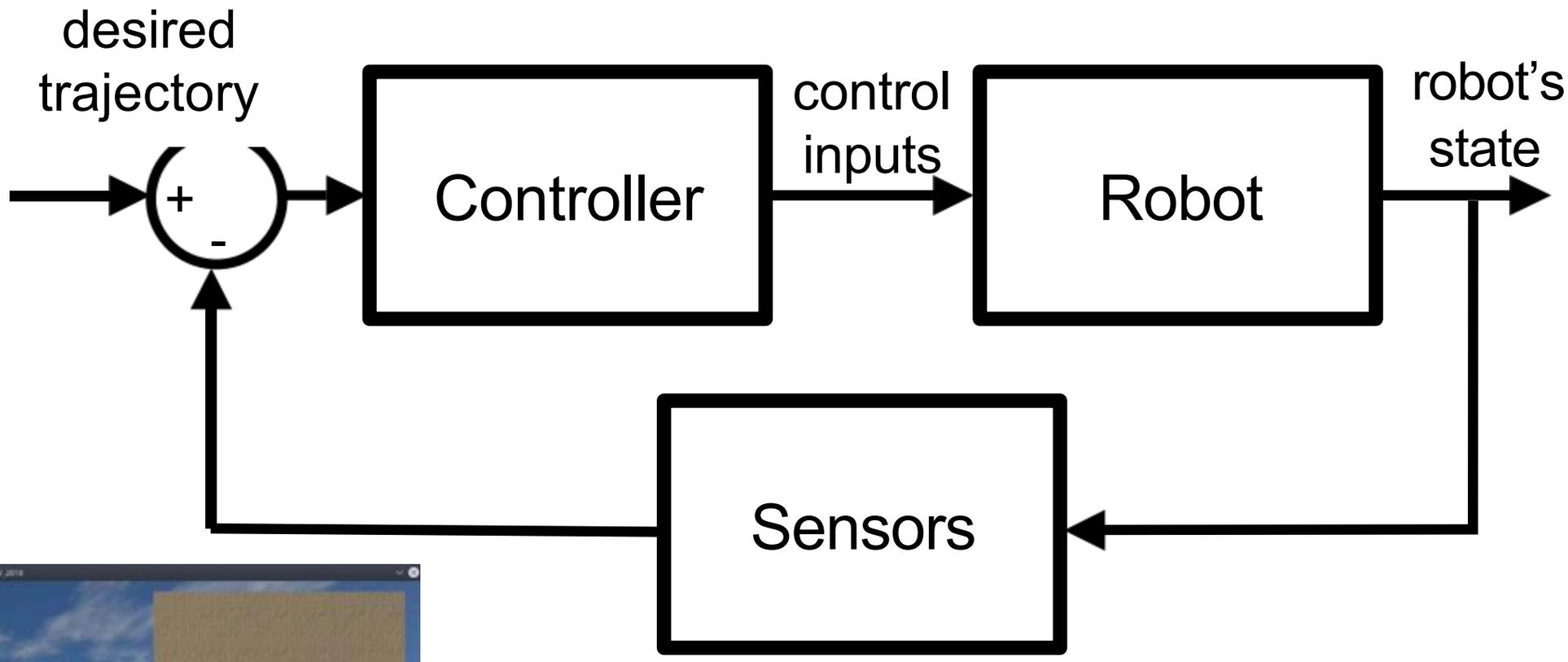
Lecture 8: Trajectory Optimization

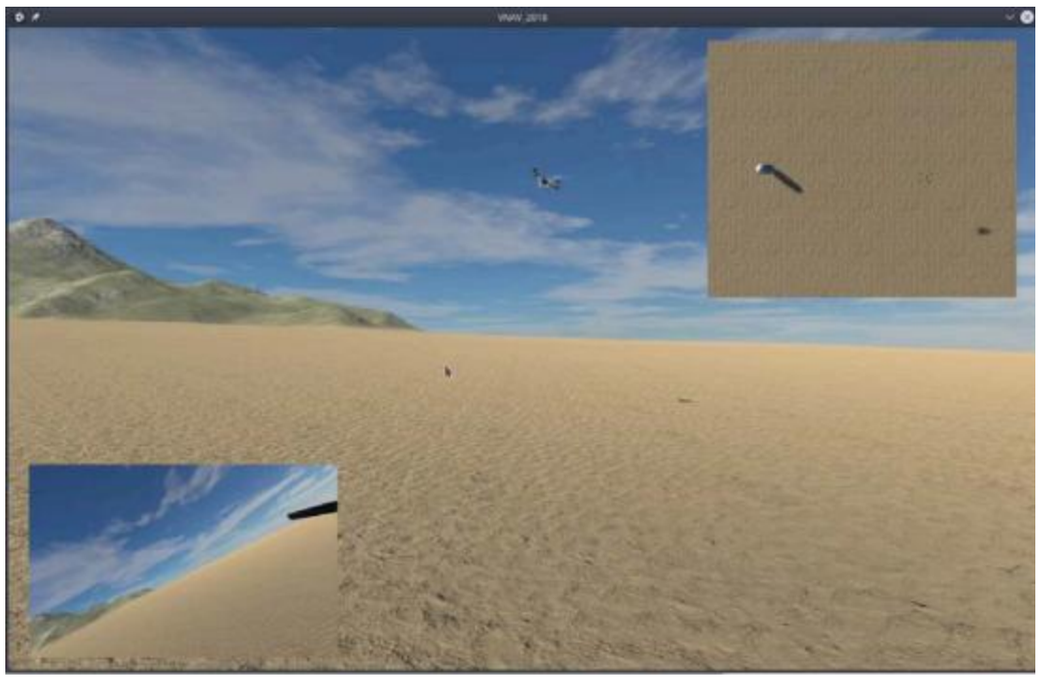
Luca Carlone

路径规划 vs. 控制 (Planning vs. Control)

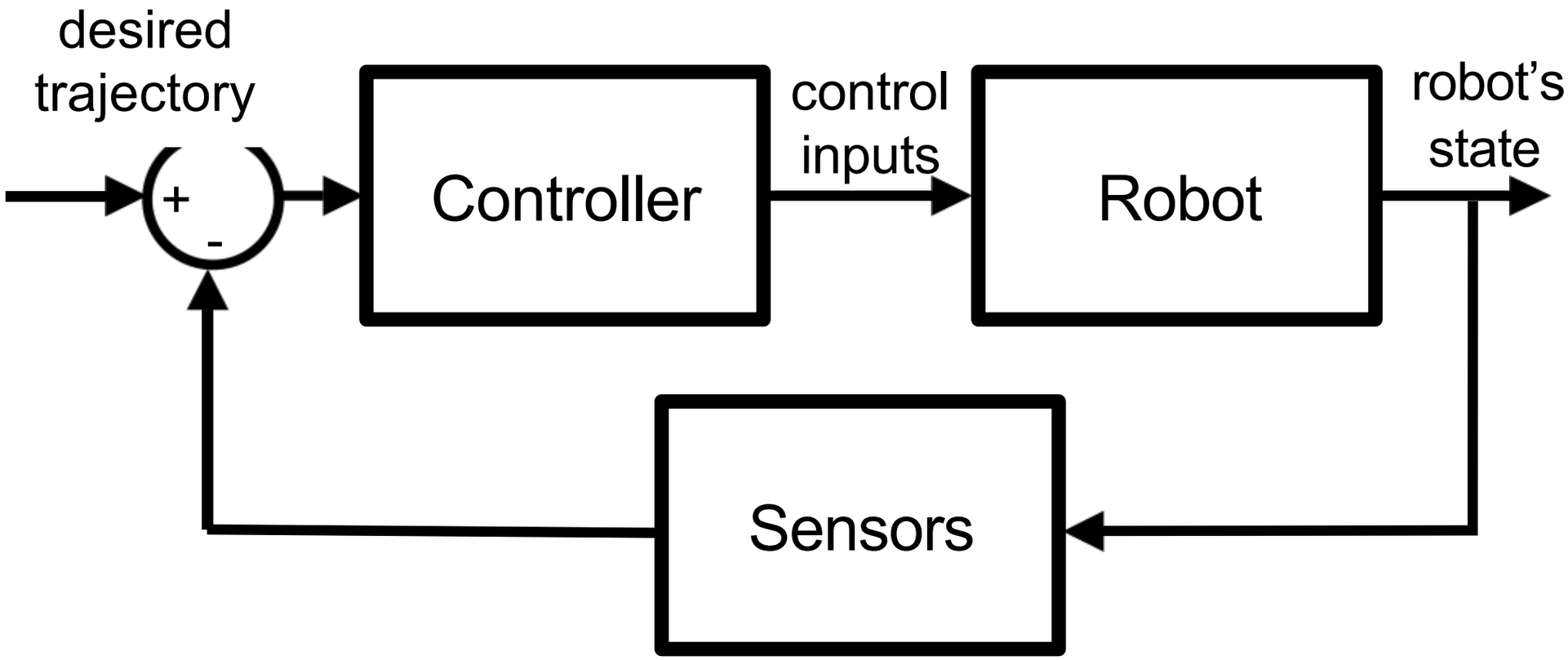


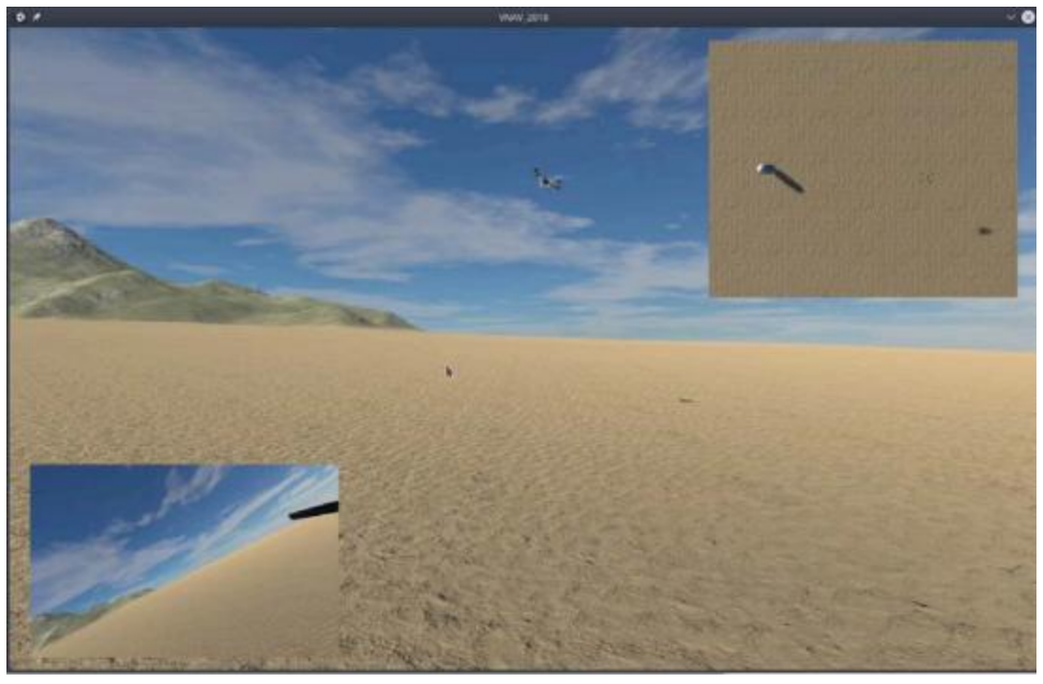
路径规划 vs. 控制

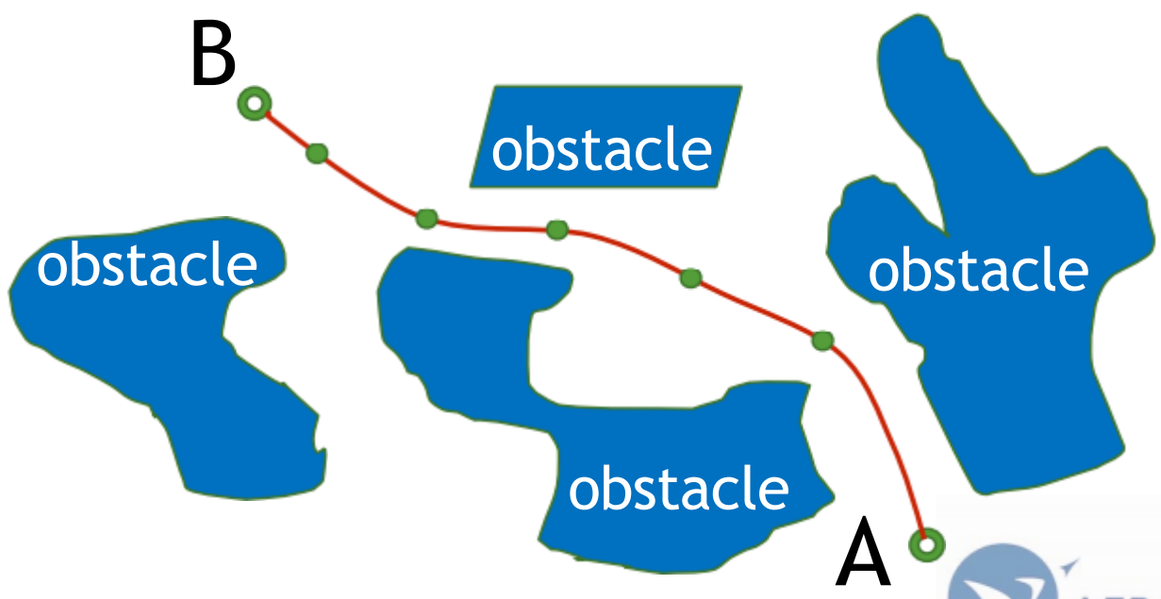

desired
trajectory
+
-
Controller
control
inputs
Robot
robot's
state
Sensors



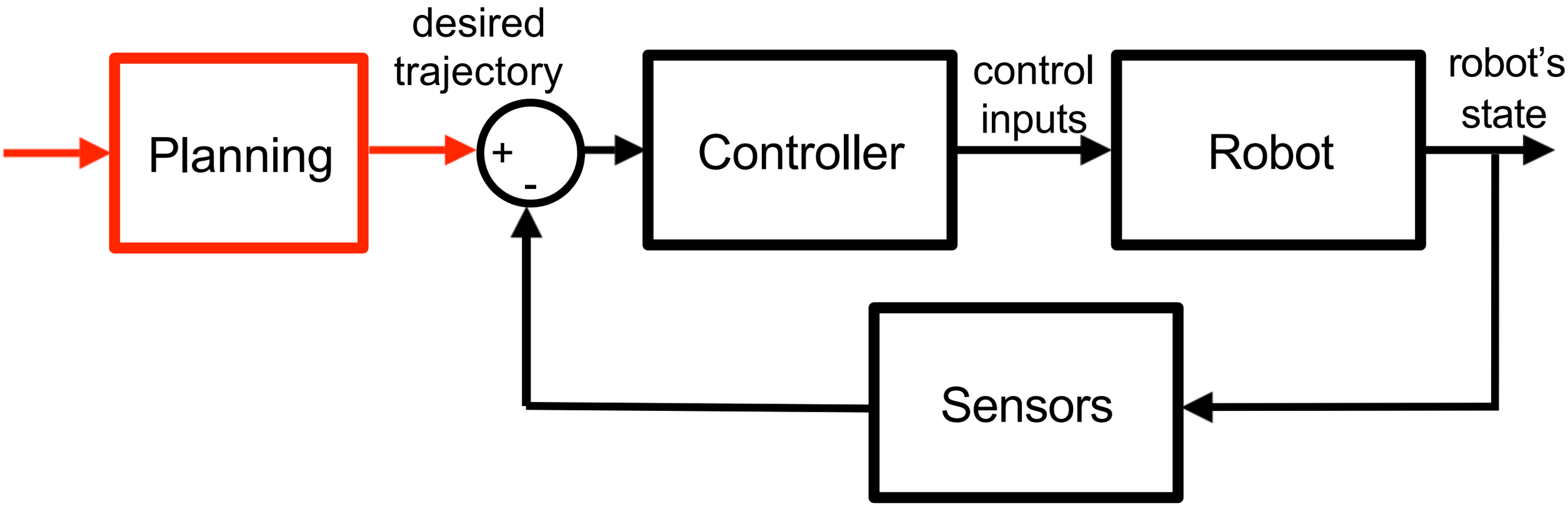
路径规划 vs. 控制

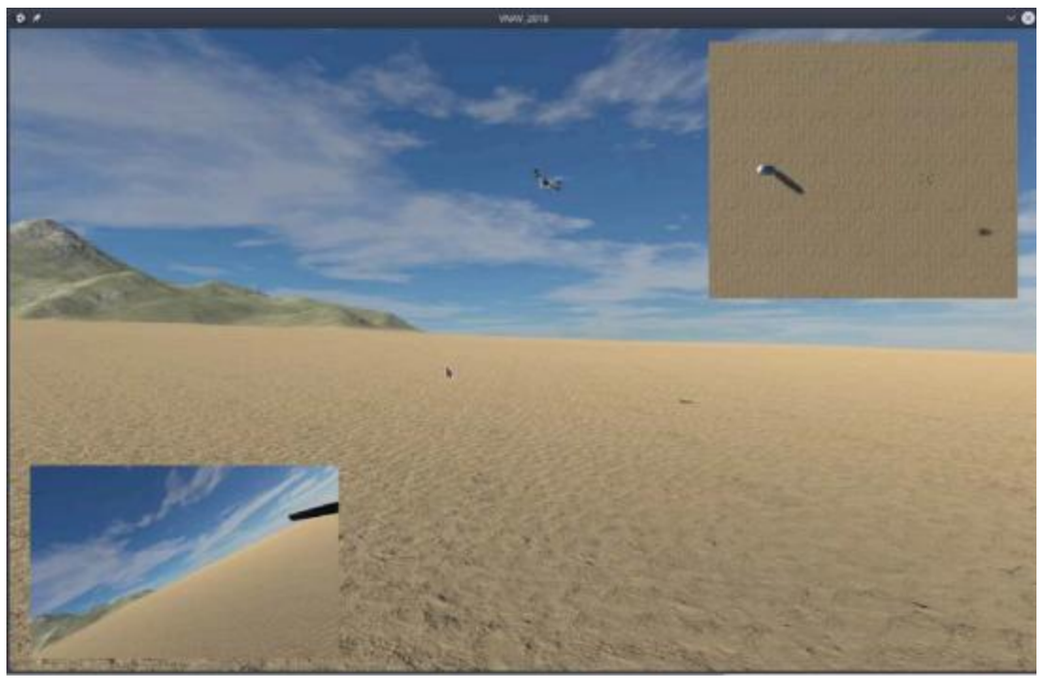

desired
trajectory
+
-
Controller
control
inputs
Robot
robot's
state
Sensors

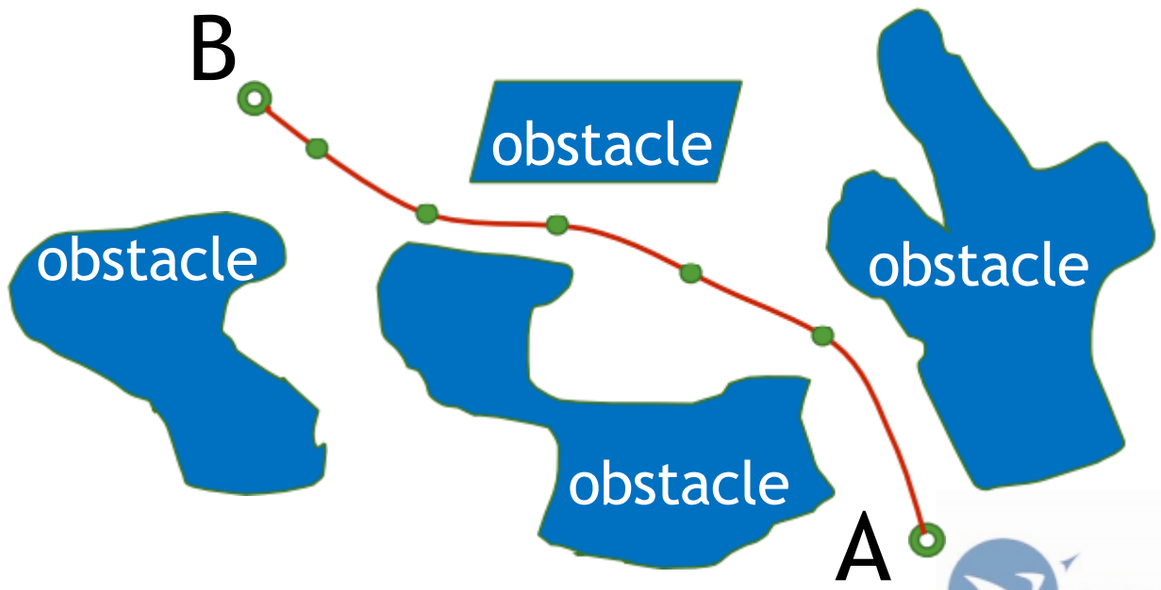



B
obstacle
obstacle
obstacle
obstacle
A

路径规划 vs. 控制





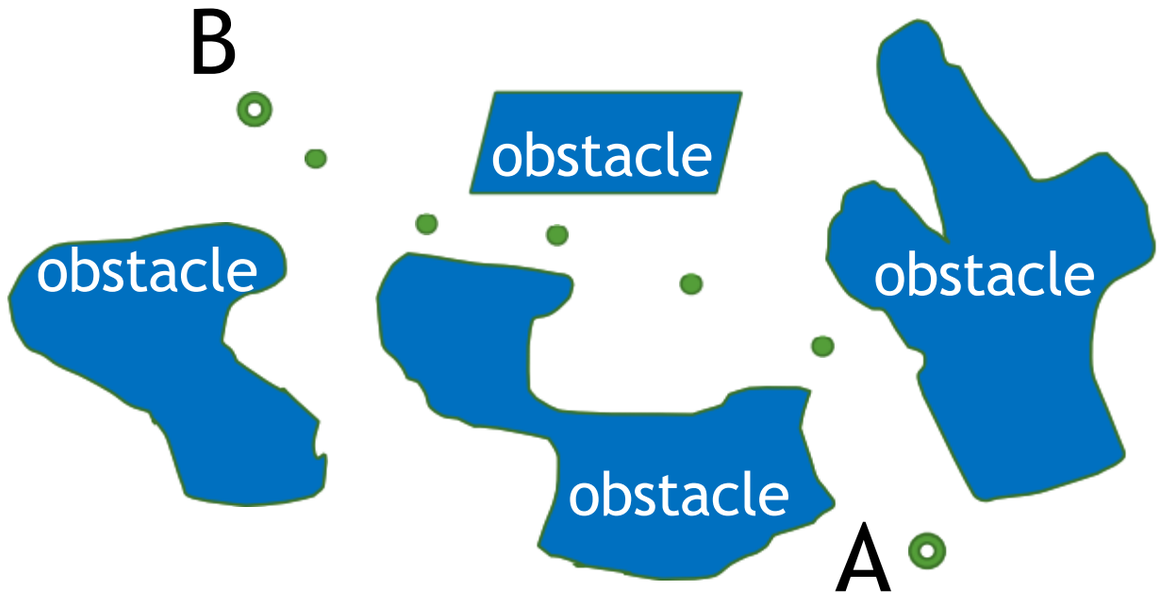


路径规划 vs. 控制

- **路径**是一系列路径点（位于无障碍空间内），没有时间标签，也没有关于速度或更高阶导数的信息。
- **Path** is a sequence of waypoints (in the obstacle-free space), without *time labels* or *information about velocity or higher order of derivatives*.

DOES NOT ACCOUT FOR DYNAMICS

不考虑动态



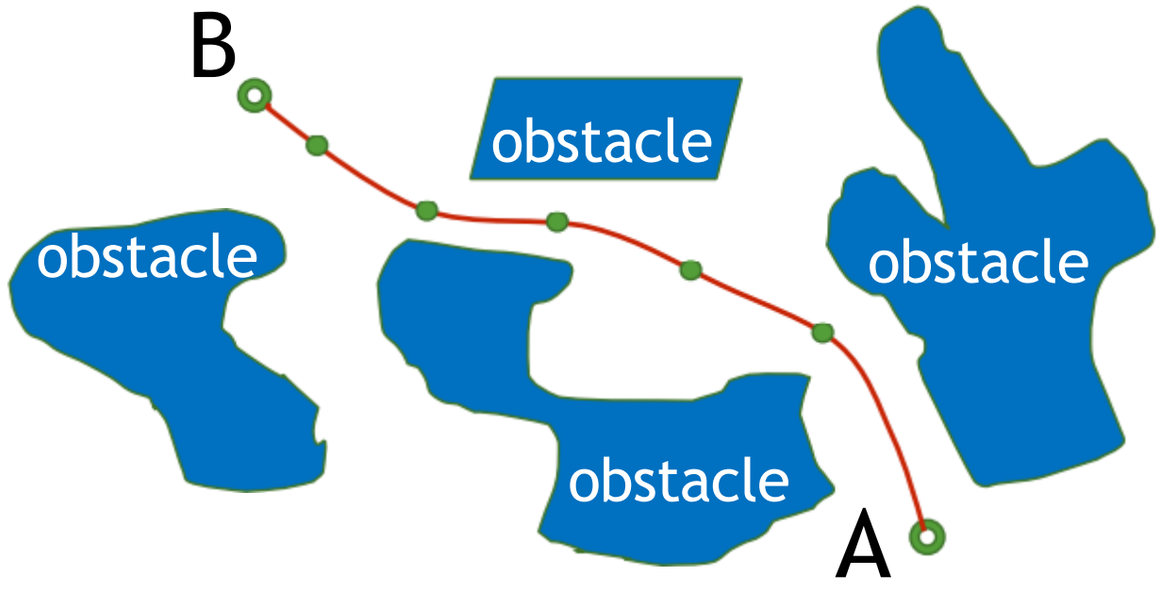
路径规划 vs. 轨迹规划

- 路径是一系列路径点（位于无障碍空间内），没有时间标签，也没有关于速度或更高阶导数的信息。

DOES NOT ACCOUT FOR DYNAMICS

- 轨迹是机器人应完成的一系列运动顺序。**Trajectory** is the sequence of movements the robot should make.

动力学的考虑 ACCOUNTS FOR DYNAMICS



Path Planning vs. Trajectory Planning

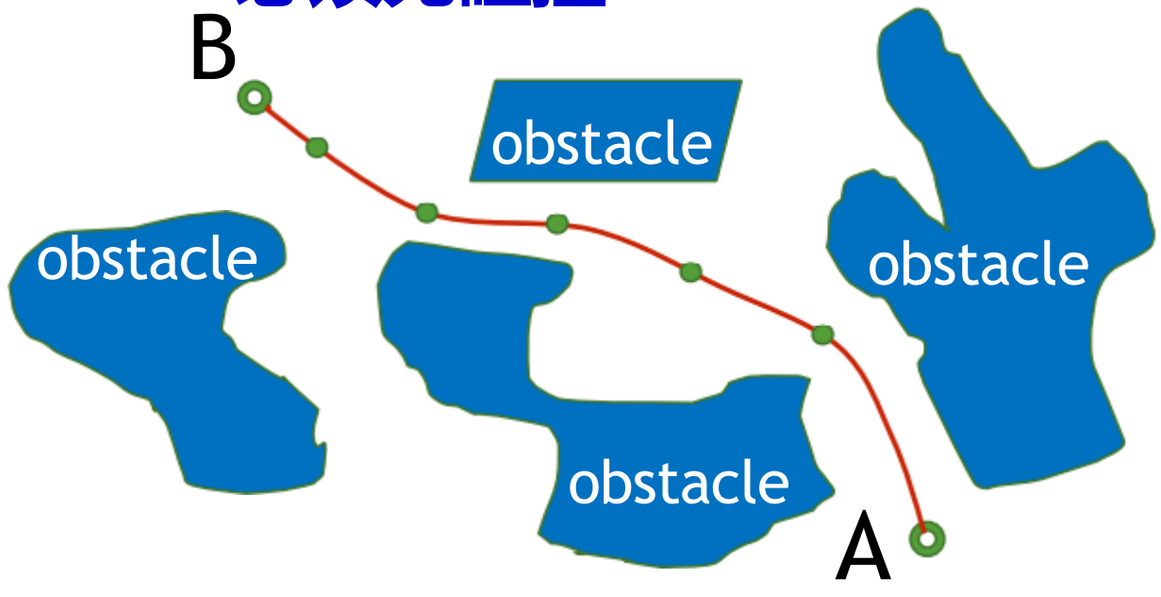
- 路径是一系列路径点（位于无障碍空间内），没有时间标签，也没有关于速度或更高阶导数的信息。

DOES NOT ACCOUT FOR DYNAMICS

- 轨迹是机器人应完成的一系列运动顺序。**Trajectory** is the sequence of movements the robot should make.

动力学的考虑 ACCOUNTS FOR DYNAMICS

MUST BE COLLISION FREE 必须无碰撞



Path Planning vs. Trajectory Planning

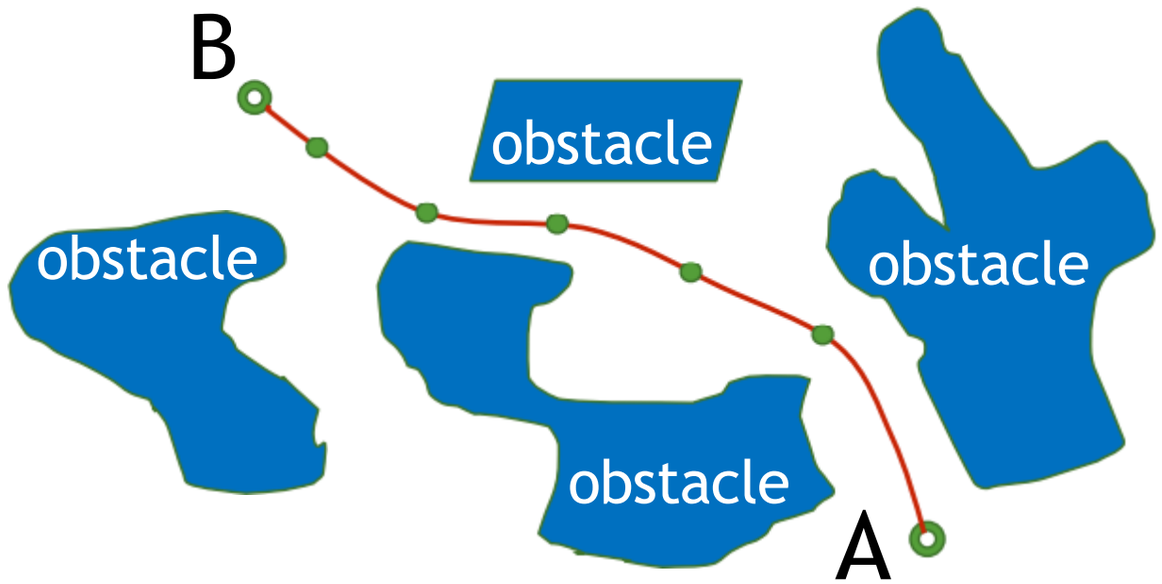
- **路径**是一系列路径点（位于无障碍空间内），没有时间标签，也没有关于速度或更高阶导数的信息。

DOES NOT ACCOUT FOR DYNAMICS

***Can account for dynamics but can be slow** (Bry et al., IJRR '15)

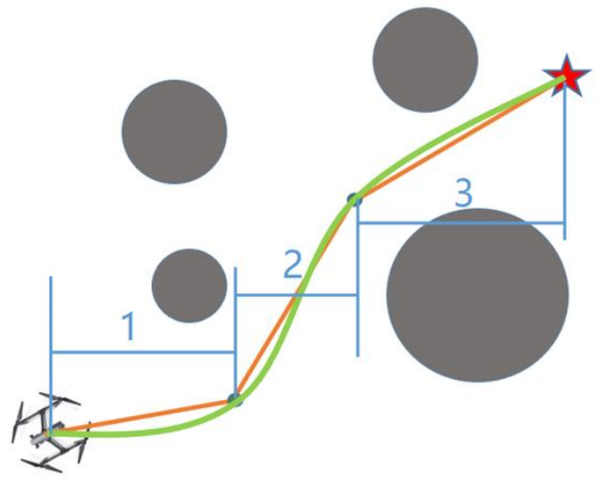
- **轨迹**是机器人应完成的一系列运动顺序。**Trajectory** is the sequence of movements the robot should make.

动力学的考虑 ACCOUNTS FOR DYNAMICS



Trajectory Planning vs. Path Planning

- **路径 vs. 轨迹**
  - 一条路径通常包含一系列路标点，没有任何时间标签或关于速度或更高阶导数的信息.
  - **轨迹**定义了机器人**应随时间航行的路径**。
- **Path planning 路径规划:**
  - 慢速“系统 “slow” systems
  - 多种方法：基于图的算法（PRM、A*及其变体）、基于抽样的方法（RRT、RRT*及其变体）
- **Trajectory planning 轨迹规划:**
  - 无人机高速机动的关键



路径规划算法

Open-source libraries

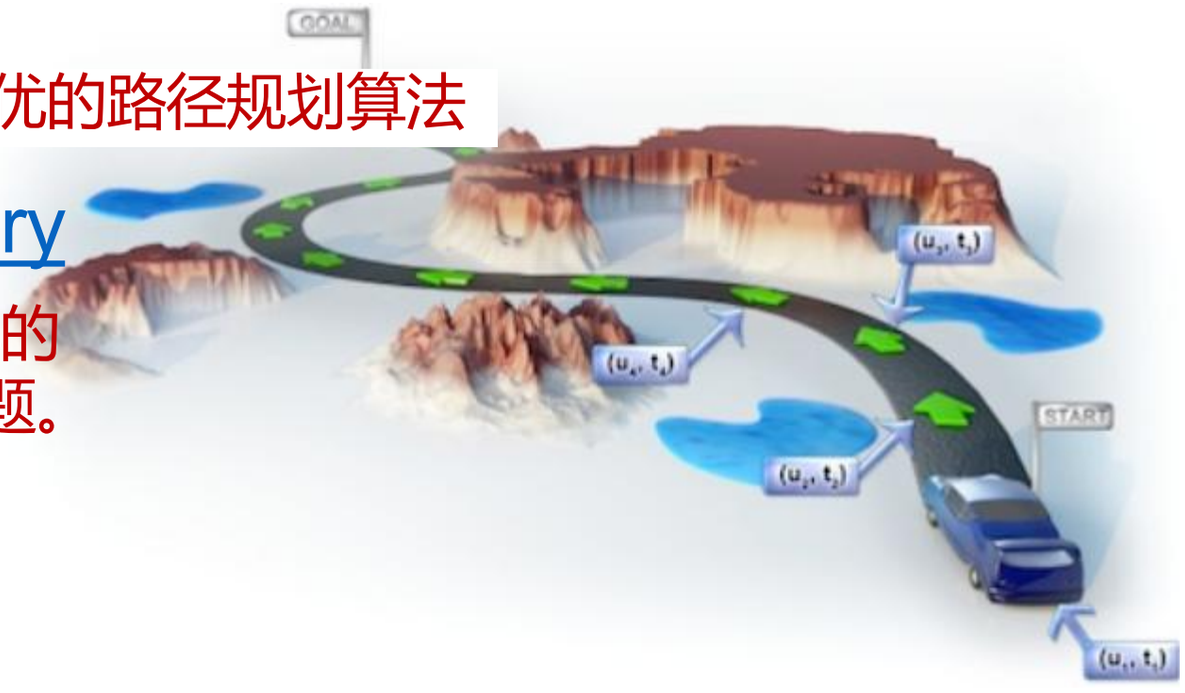
- Open Motion Planning Library (OMPL)
  - http://ompl.kavrakilab.org/
- Motion Strategy Library (MSL)
  - http://msl.cs.uiuc.edu/msl/
- RRT* Library
- Sampling Based Planning Library

OMPL是莱斯大学Kavraki实验室提供的开源运动规划库

MSL是UIUC的Steve Lavalle等人用C++写的机器人运动规划仿真平台

RRT*算法是一种渐近最优的路径规划算法

sampling-based-planners 是一个基于采样的规划器集合，旨在解决机器人路径规划问题。



References

- Howie Choset et al., “Principles of Robot Motion,” MIT press, 2005.
- Steven Lavalle, “Planning Algorithms,” Cambridge University Press, 2006.

路径规划的两种类型

组合规划

- （resolution）完备。

如果存在解决方案，他们会找到，否则报告失败

- 无需碰撞检查，因为 是明确定义的
- 计算成本高，对高维系统不切实际. 细胞数量爆炸. Exploding number of cells
- 输出路径通常是锯齿状的，需要平滑处理

基于采样的规划

- 概率完备.

如果解存在，找到解的概率趋向于一个，否则解可能永远运行下去。

- 更实用。
- 狭窄通道挑战。
- 输出路径通常呈锯齿状，需要
平滑。

Path / Motion Planning

IRON ROBOT

完备性：

是指如果在起始点和目标点间有路径解存在，那么一定可以得到解，如果得不到解那么一定说明没有解存在；

概率完备性：

是指如果在起始点和目标点间有路径解存在，只要规划或搜索的时间足够长，就一定能确保找到一条路径解；

最优性：

是指规划得到的路径在某个评价指标上是最优的（评价指标一般为路径的长度）；

渐进最优性：

是指经过有限次规划迭代后得到的路径是接近最优的次优路径，且每次迭代后都与最优路径更加接近，是一个逐渐收敛的过程；

算法类型	具体算法	是否完备	是否最优
基于搜索	Dijkstra算法、A*算法	完备	是
基于采样	PRM、RRT	概率完备	否
	RRT* 、Informed RRT*	概率完备	渐进最优
基于智能算法	遗传算法、蚁群算法	完备	否

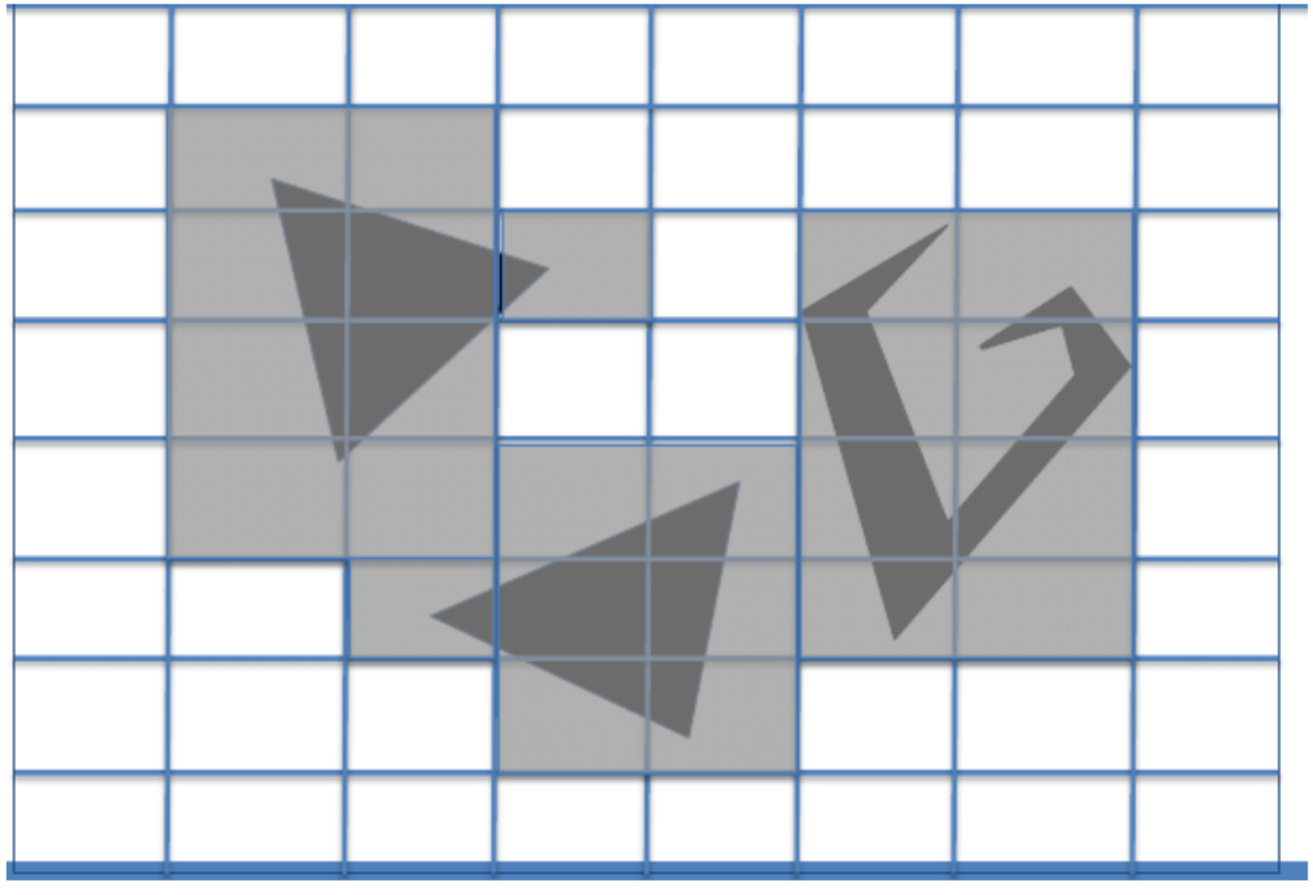
路径规划

组合规划 COMBINATORIAL PLANNING

需要将图进行单元分割（cell decomposition），将其转化为离散的空间。分割后就可以使用Dijsktra算法或A*算法等来求解路径。

Uniform cell decomposition

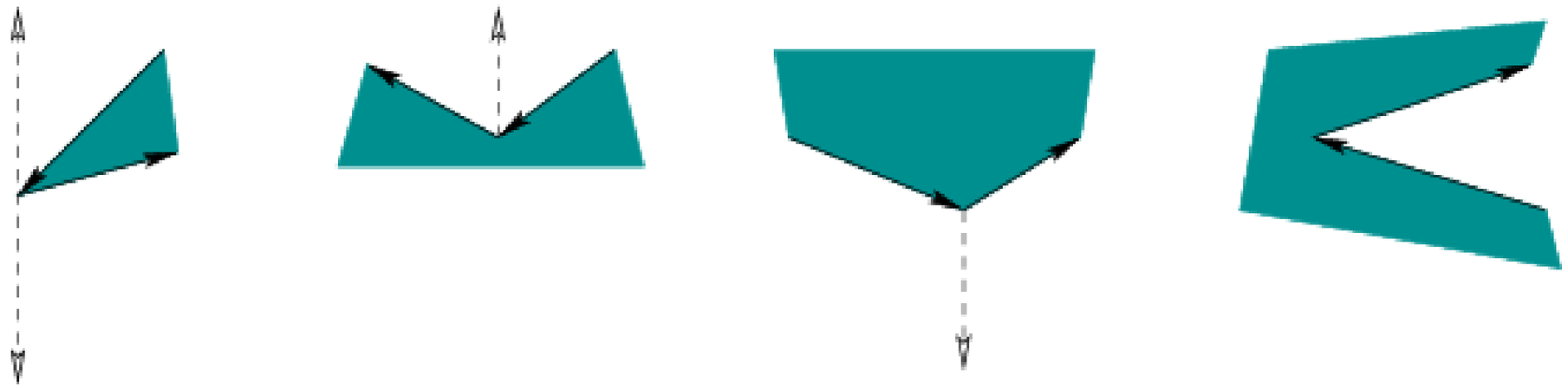
用栅格 grid 做空间分割

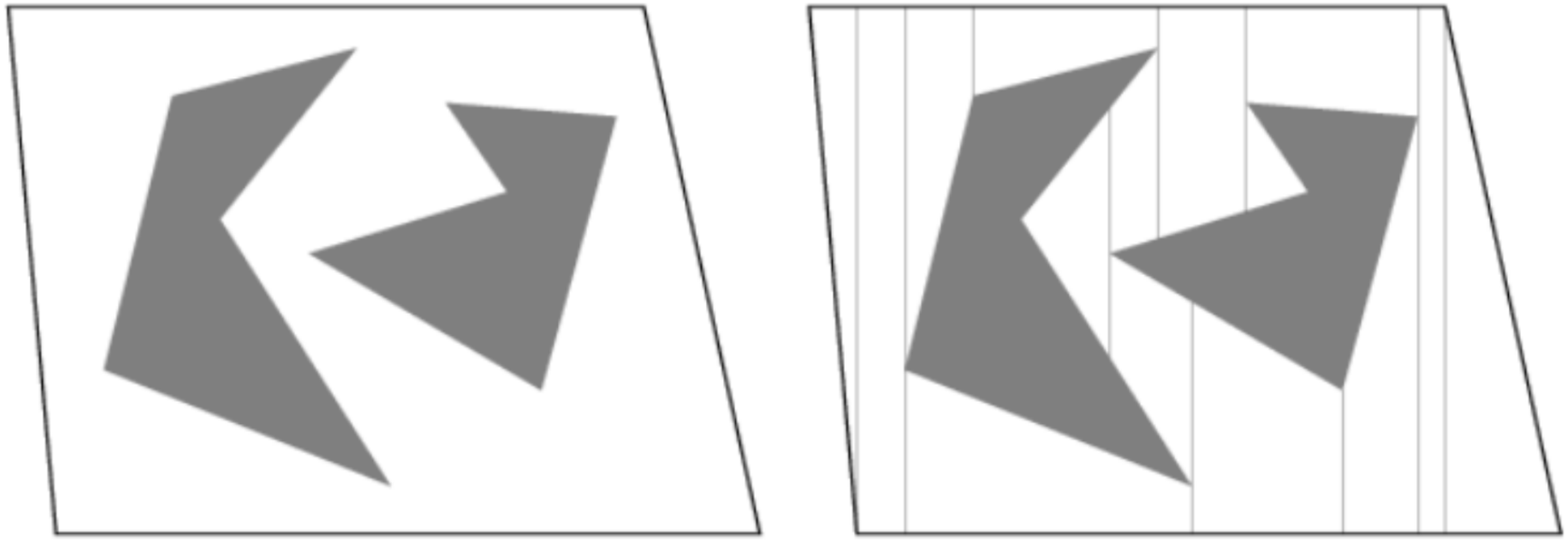


路径规划

组合规划 COMBINATORIAL PLANNING

垂直单元分解

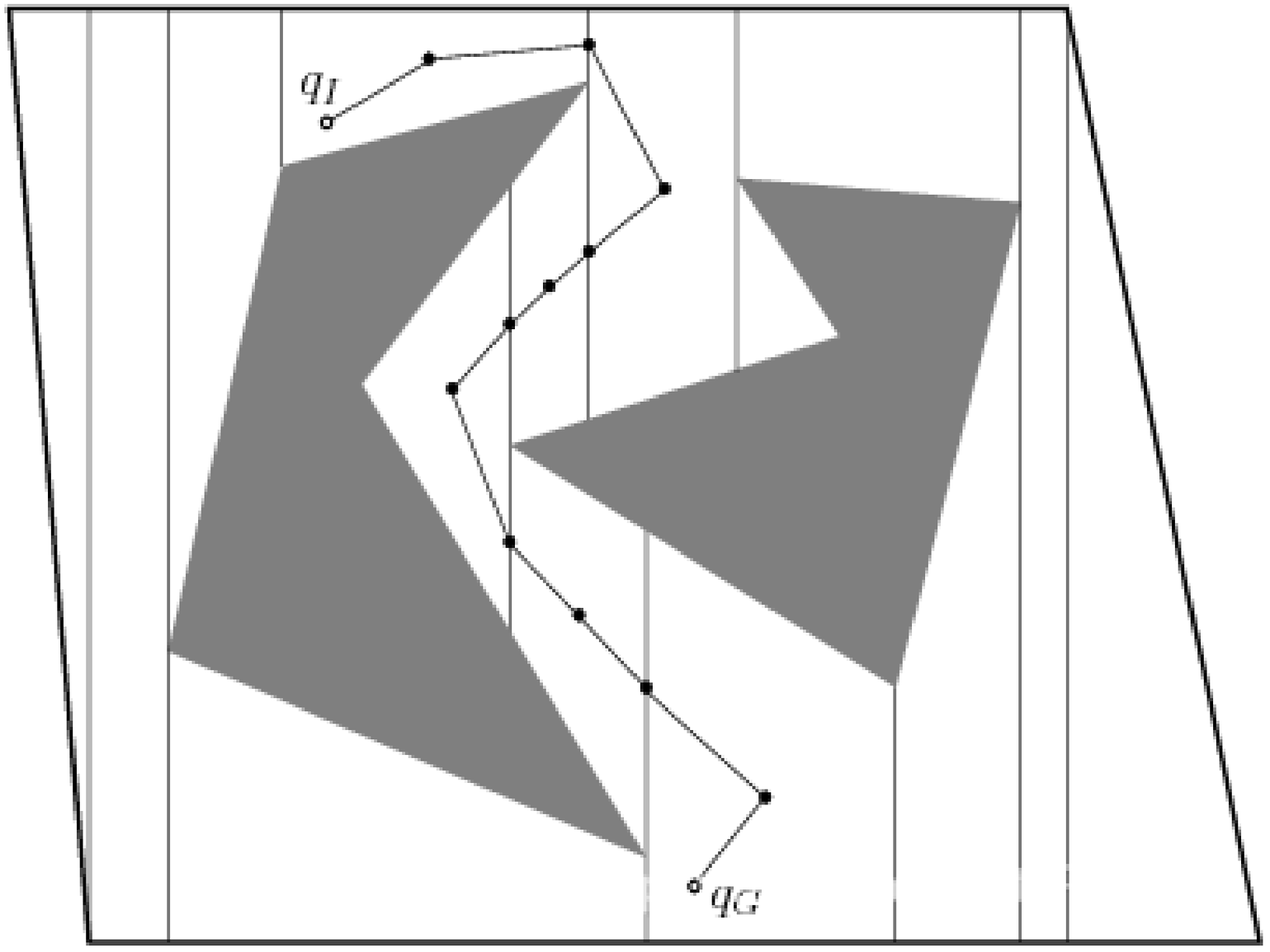




Path / Motion Planning

组合规划 COMBINATORIAL PLANNING

产生的路径与两侧障碍物之间的距离之差较小



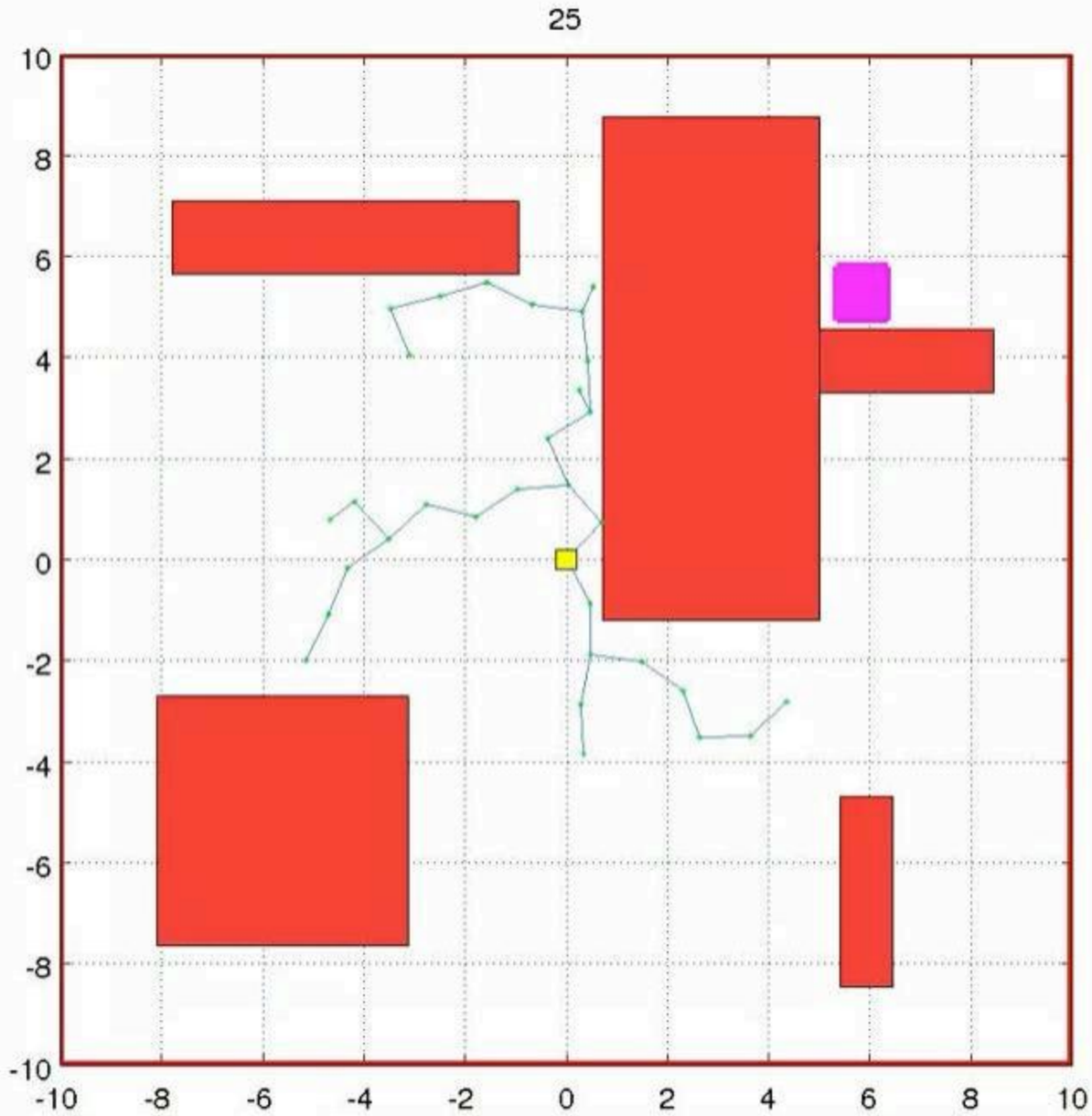
路径规划算法--示例

基于采样的路径规划

RRT*: Rapidly exploring Random Trees

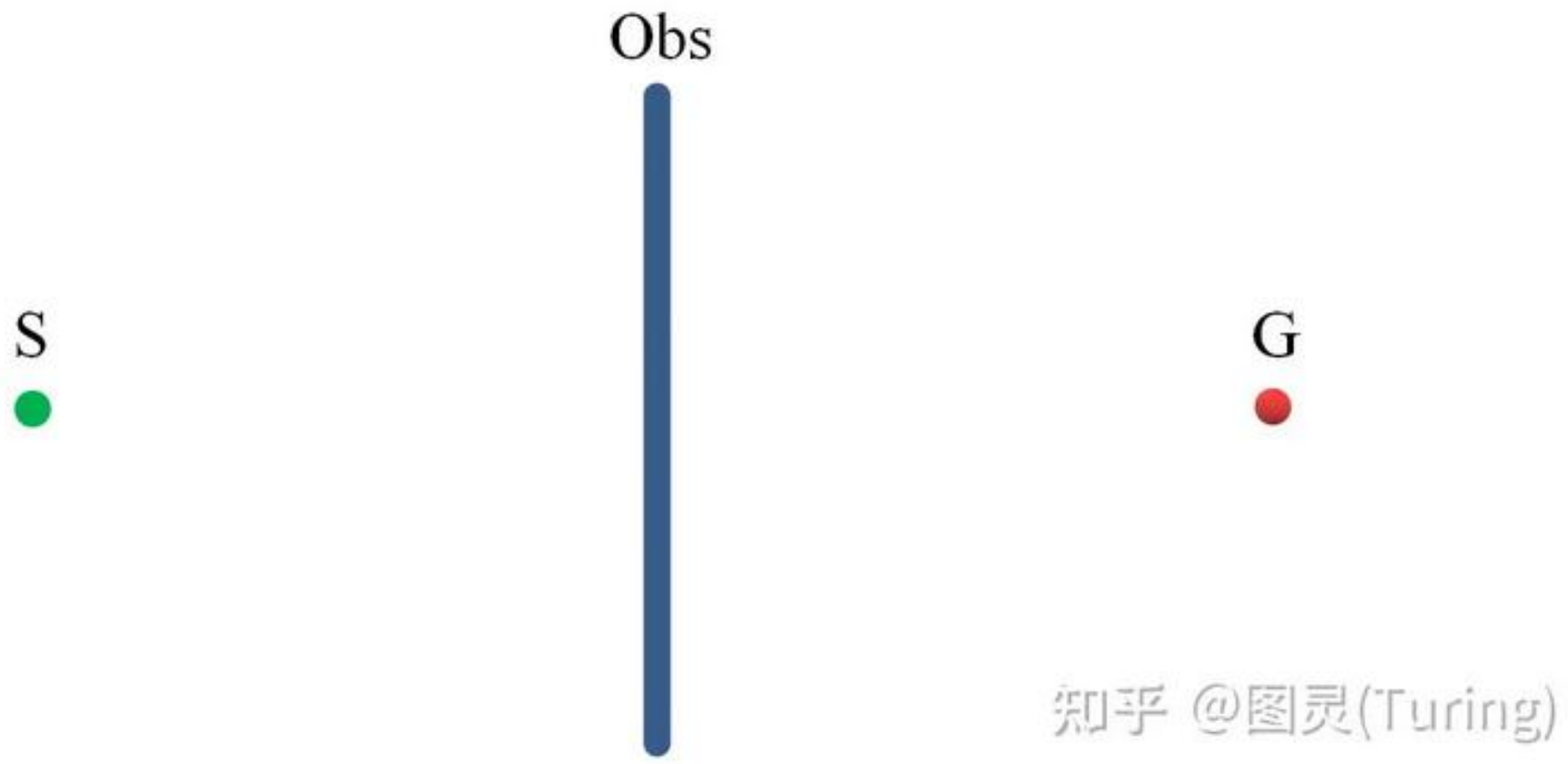
算法讲解:

https://zhuanlan.zhihu.com/p/709123692



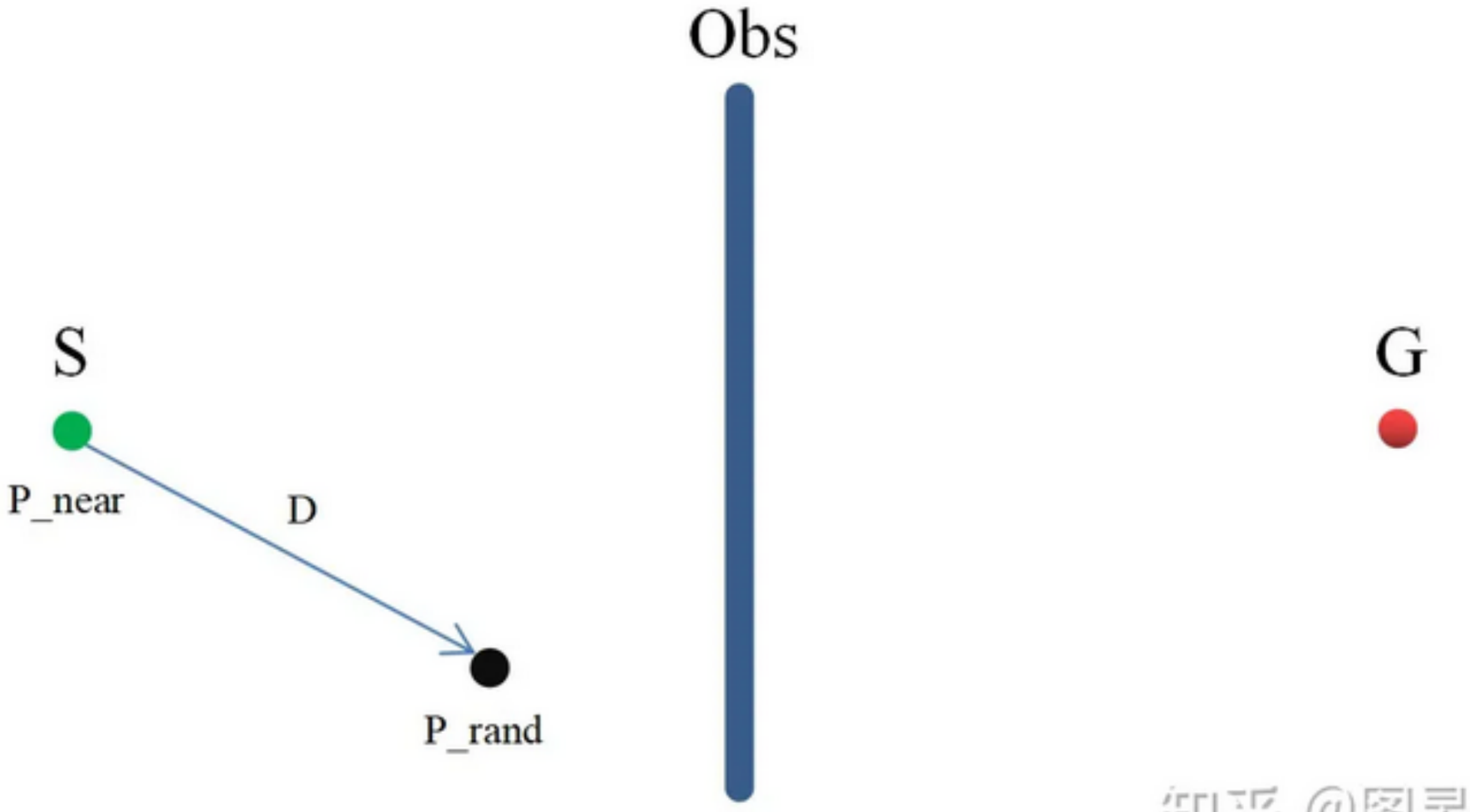
Path / Motion Planning Example

RRT*作为RRT算法的改进，首先需要了解RRT算法的基本原理



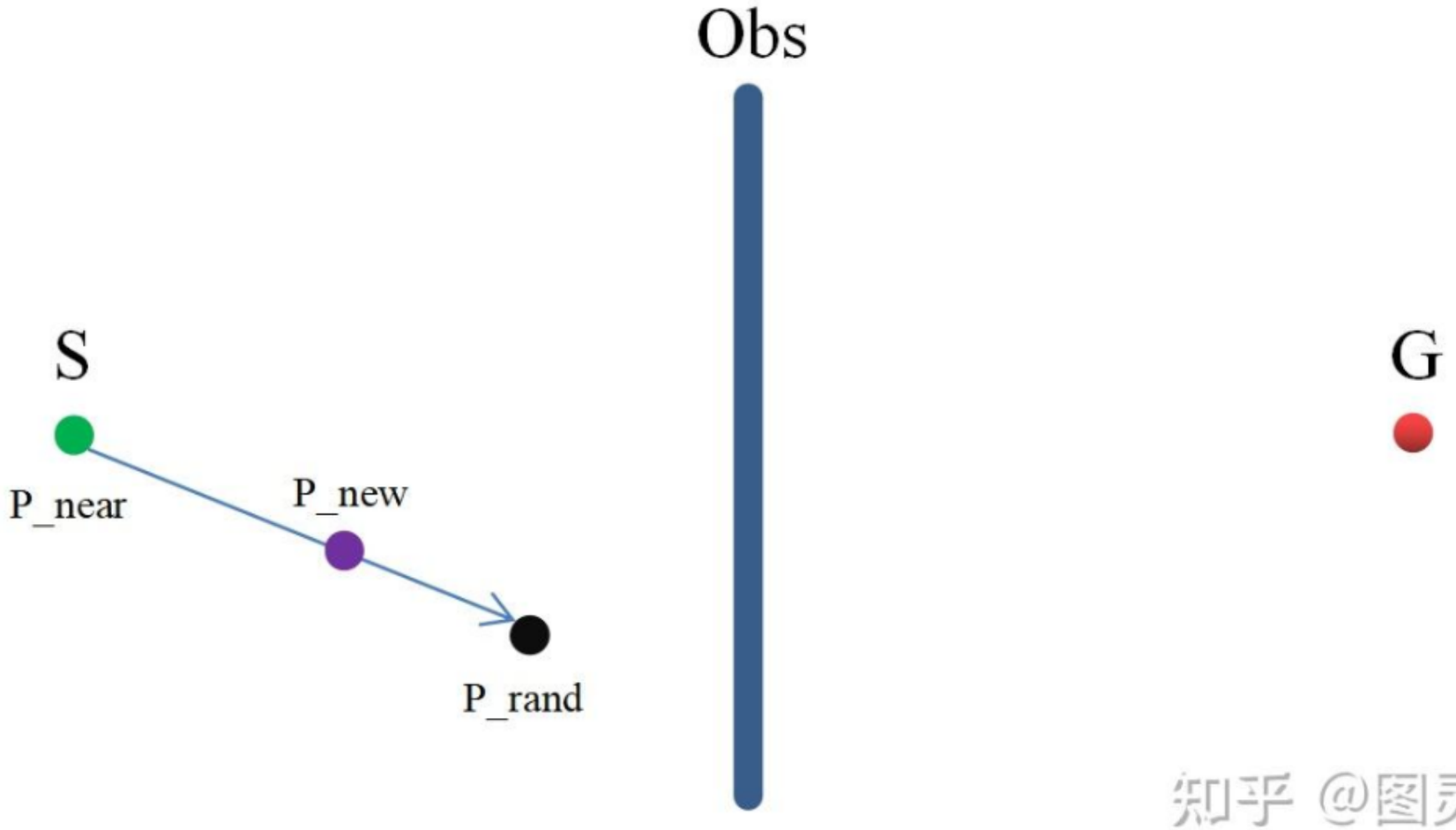
Path / Motion Planning Example

RRT*作为RRT算法的改进，首先需要了解RRT算法的基本原理



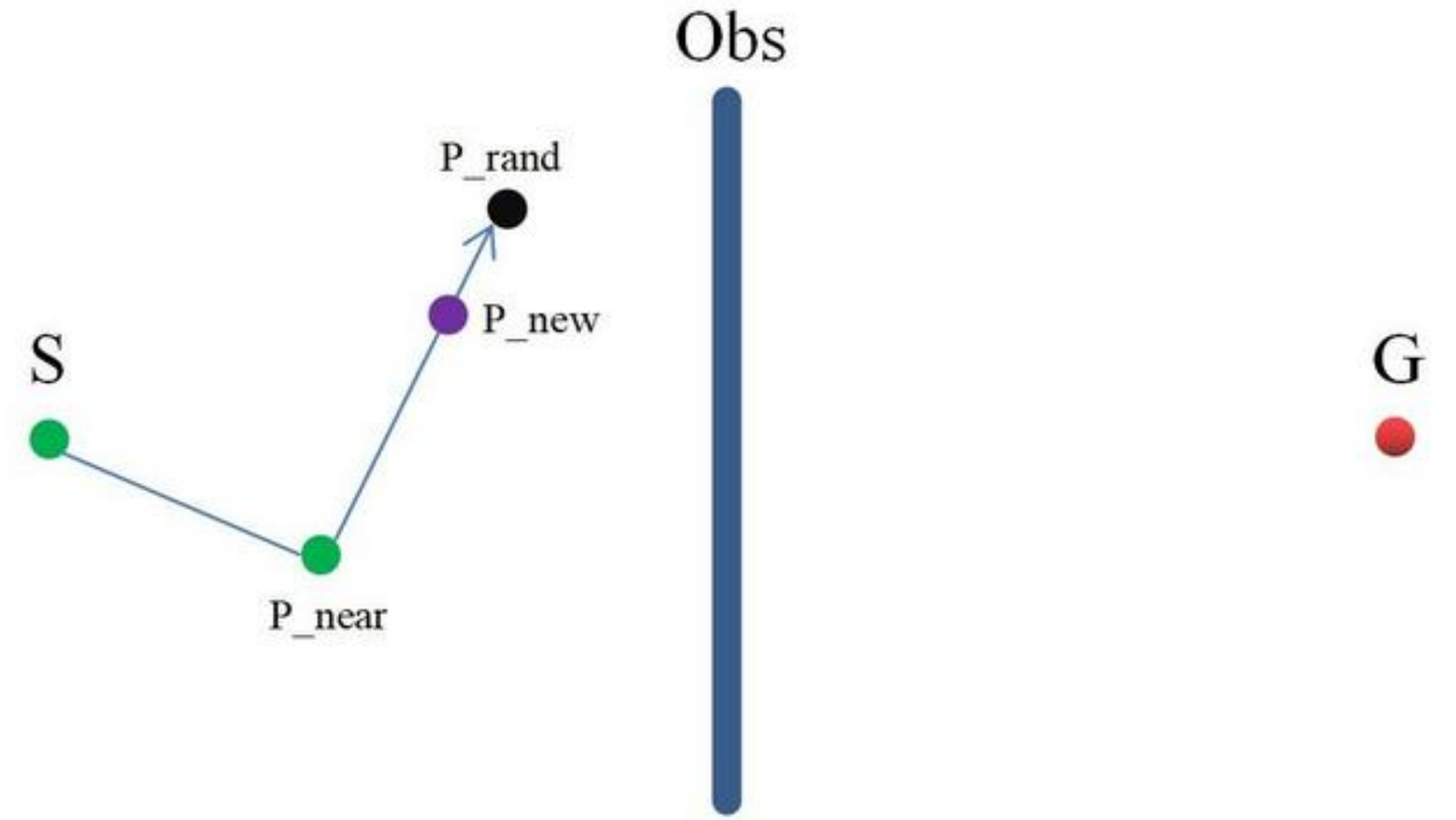
Path / Motion Planning Example

RRT*作为RRT算法的改进，首先需要了解RRT算法的基本原理



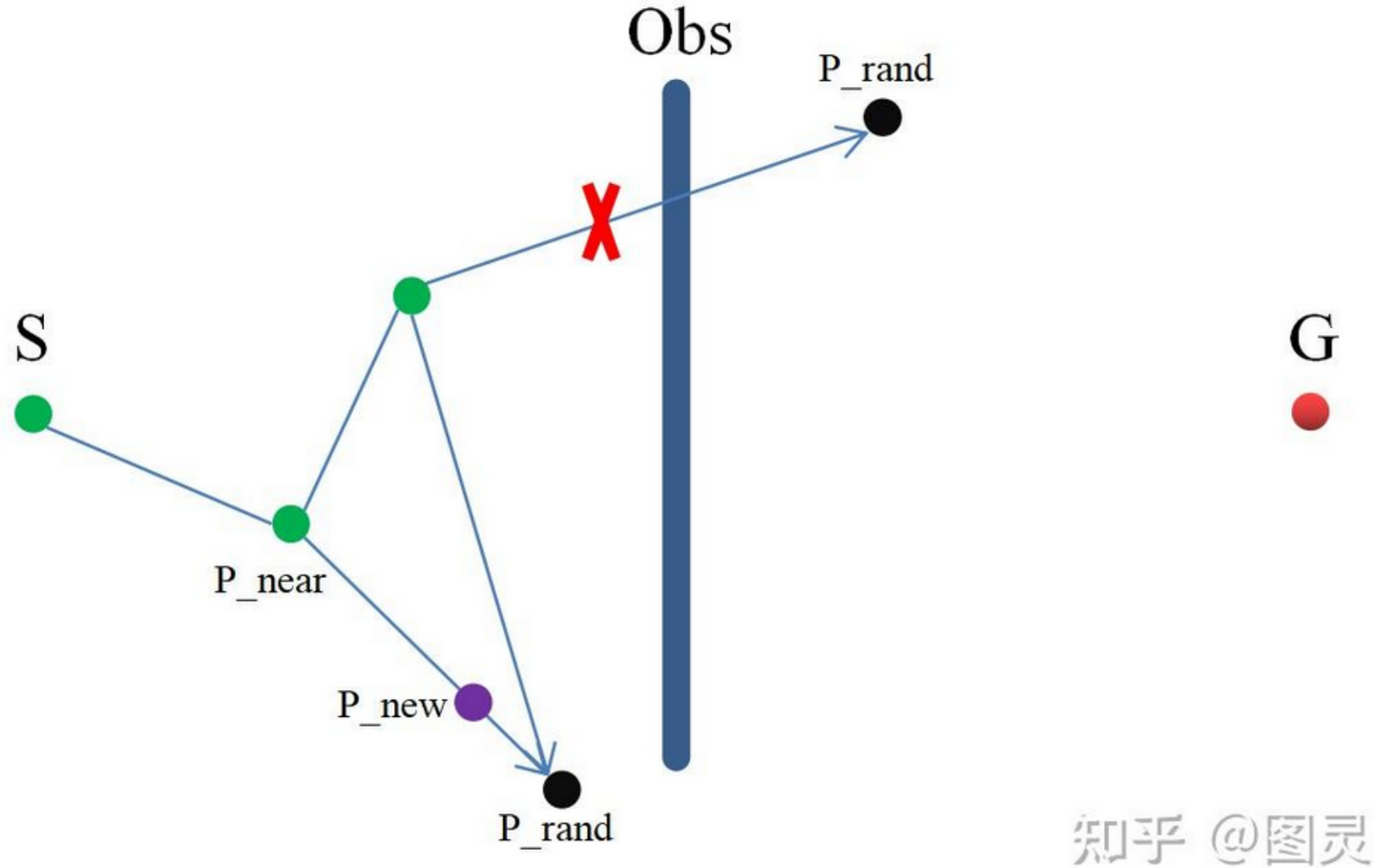
Path / Motion Planning Example

RRT*作为RRT算法的改进，首先需要了解RRT算法的基本原理



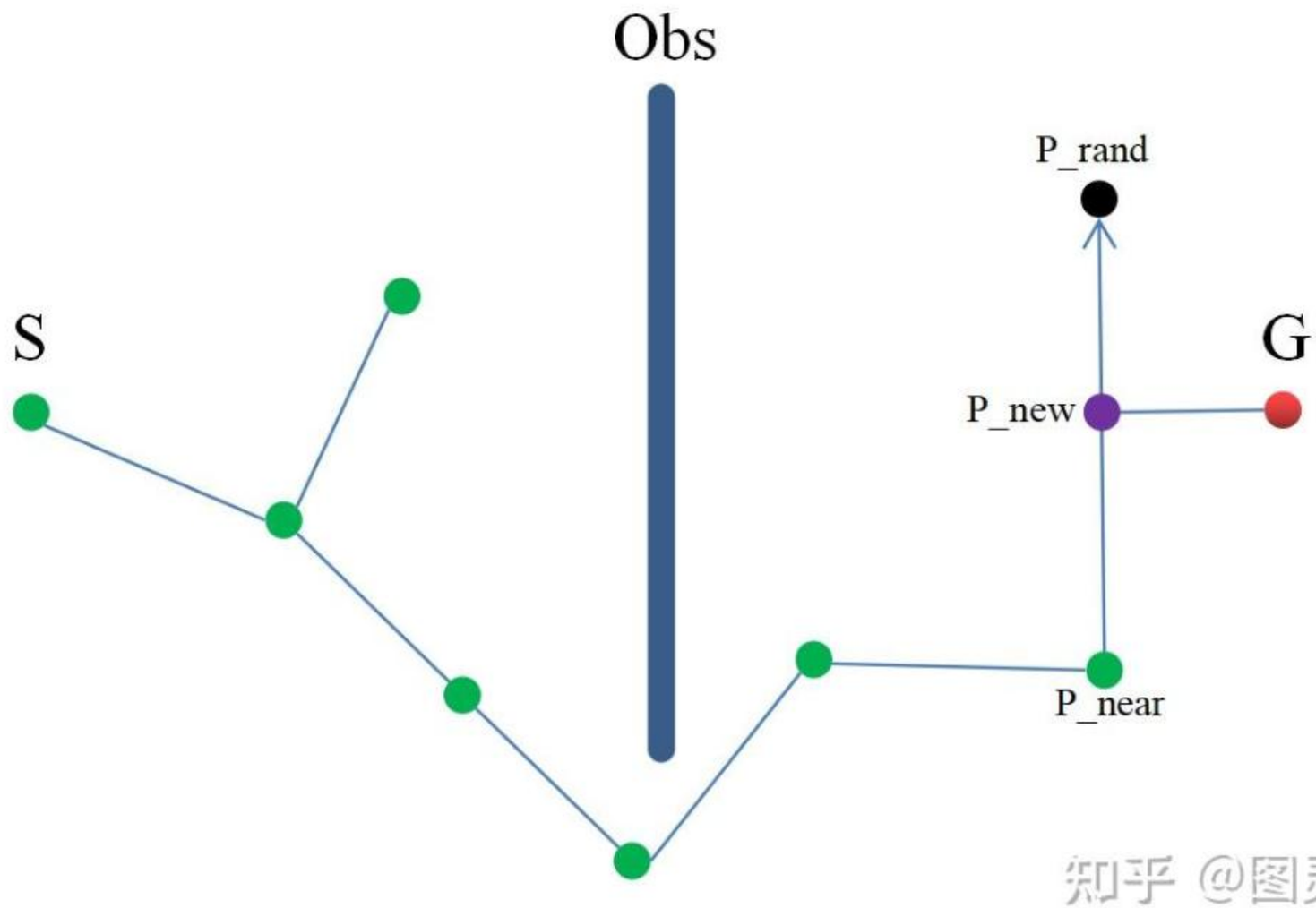
Path / Motion Planning Example

RRT*作为RRT算法的改进，首先需要了解RRT算法的基本原理

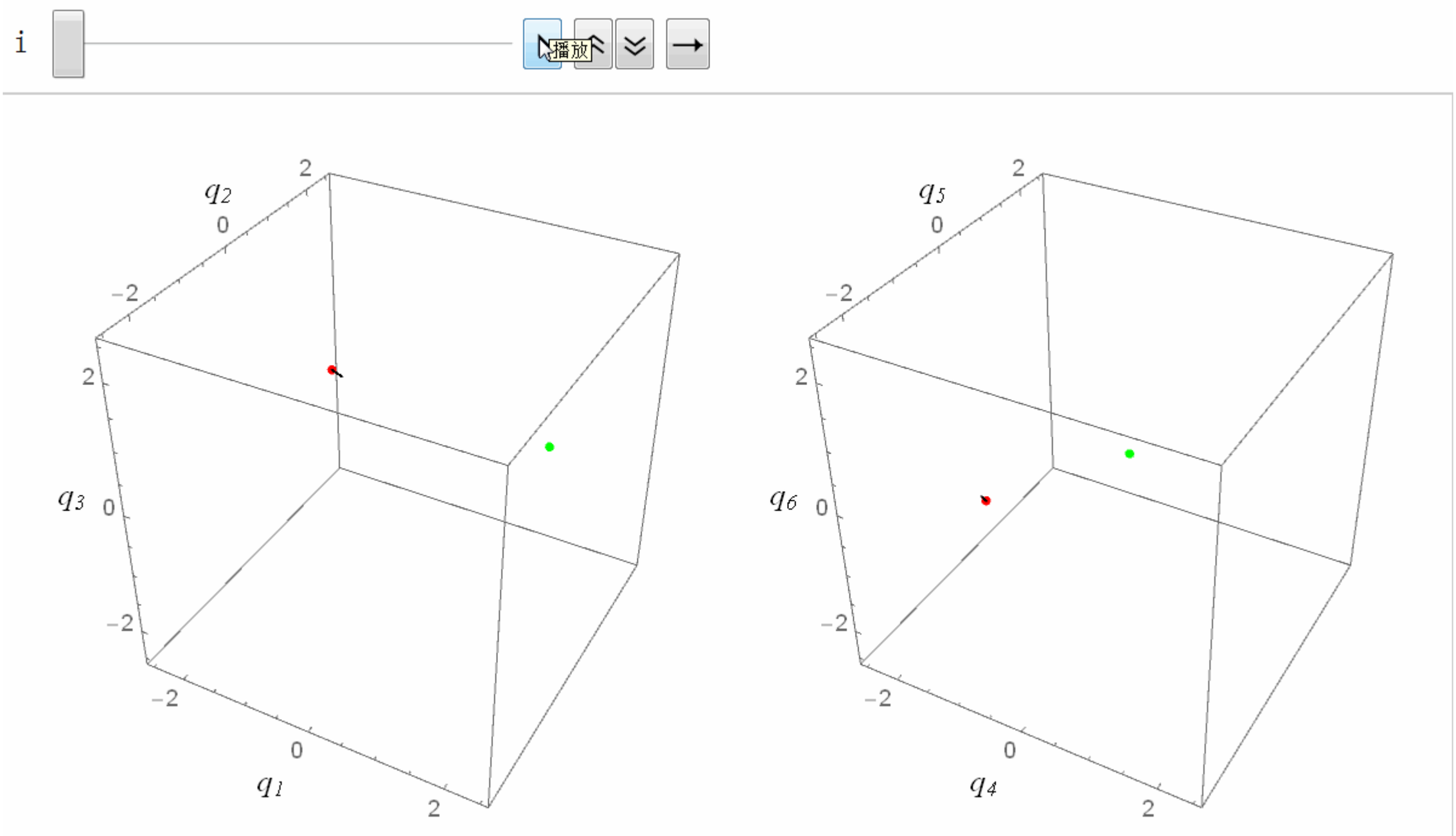


Path / Motion Planning Example

- RRT*作为RRT算法的改进，首先需要了解RRT算法的基本原理



路径规划算法--示例

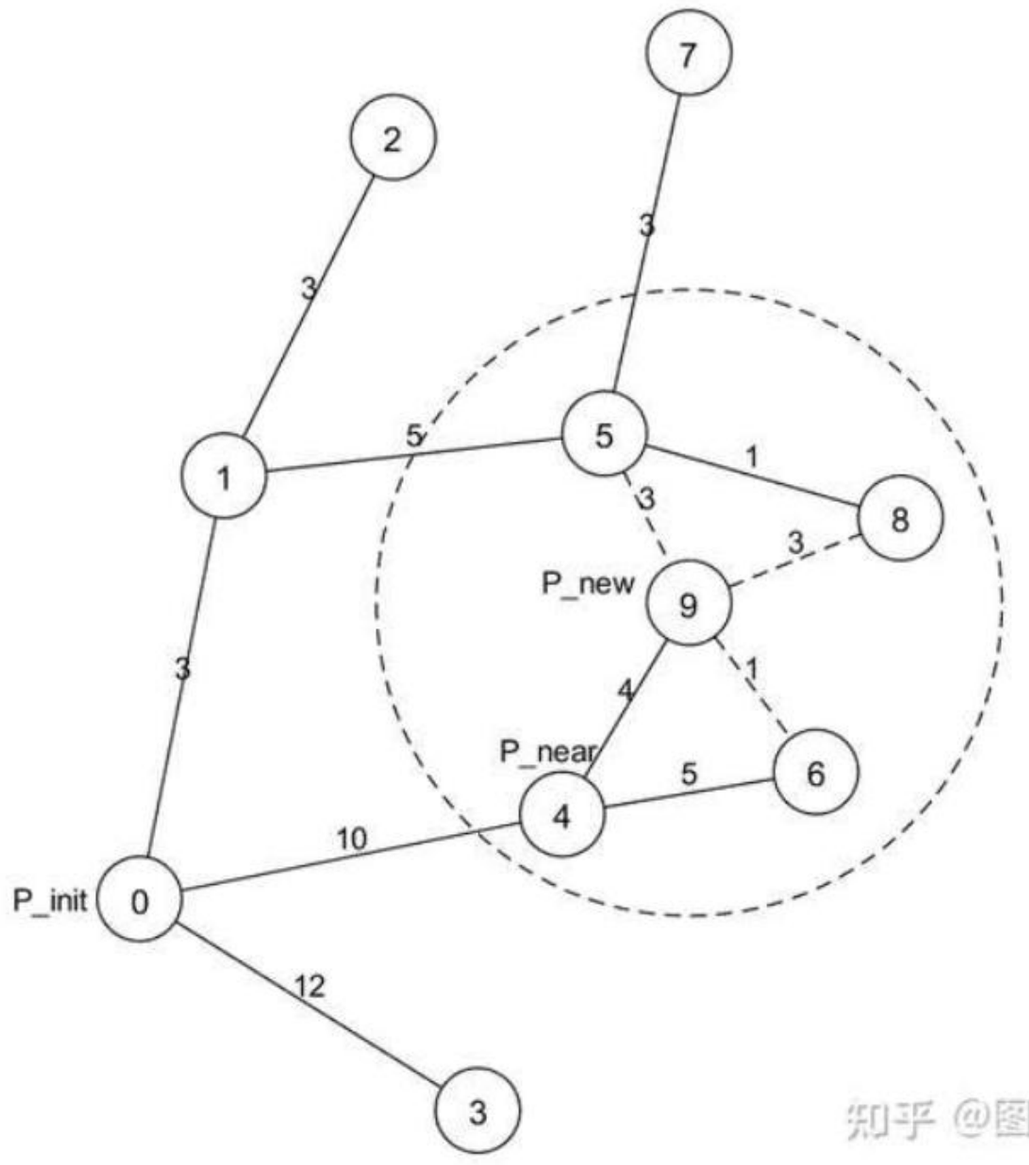


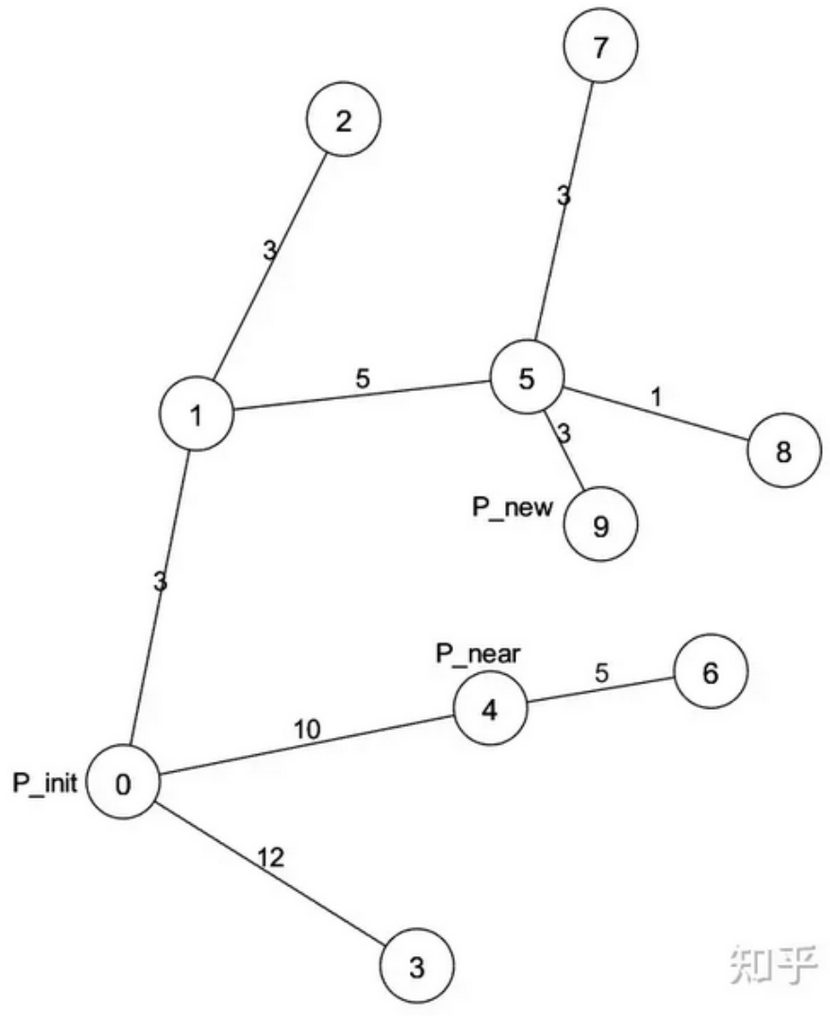
路径规划算法--示例

RRT*一共改进了两处

1、重新选择父节点过程

在新产生的节点 p_new附近以定义的半径范围内寻找“近邻”，作为替换 父节点的备选。依次计算“近邻”节点到起点的路径代价加上到每个“近邻”的路径代价，具体过程见下图：





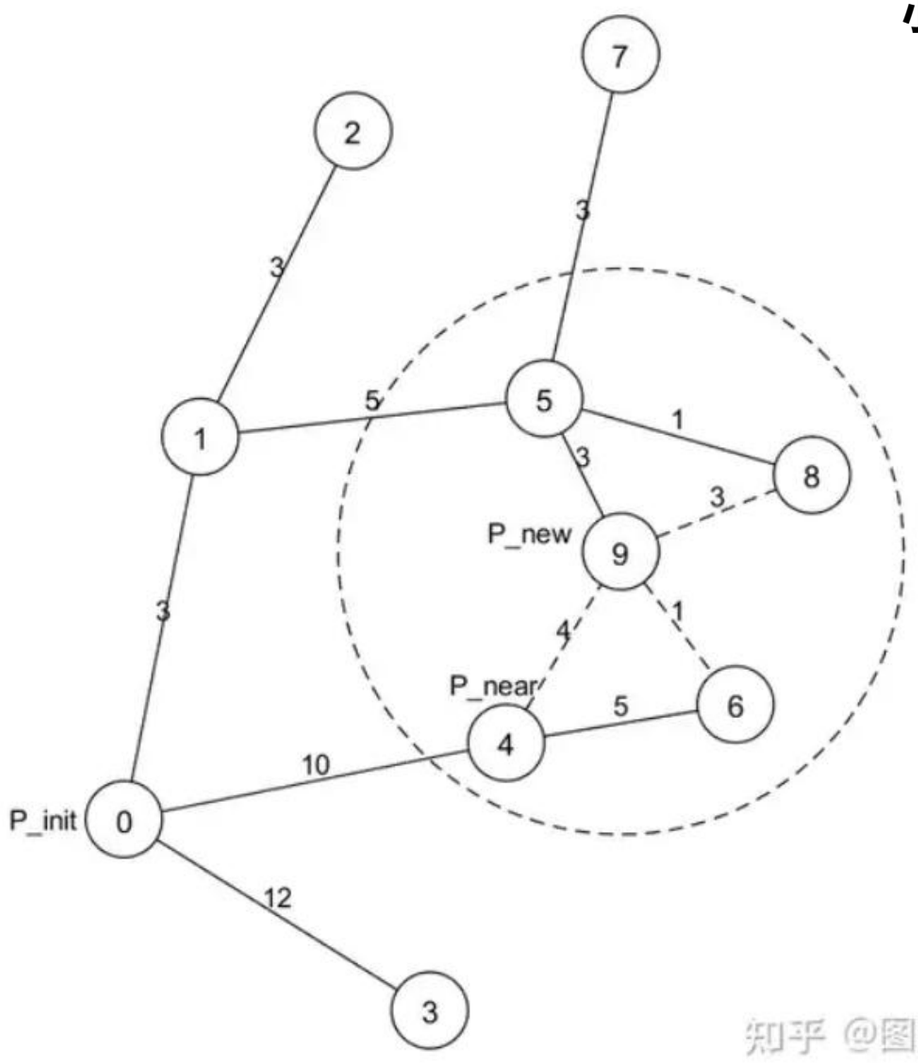
路径规划算法--示例

RRT*一共改进了两处

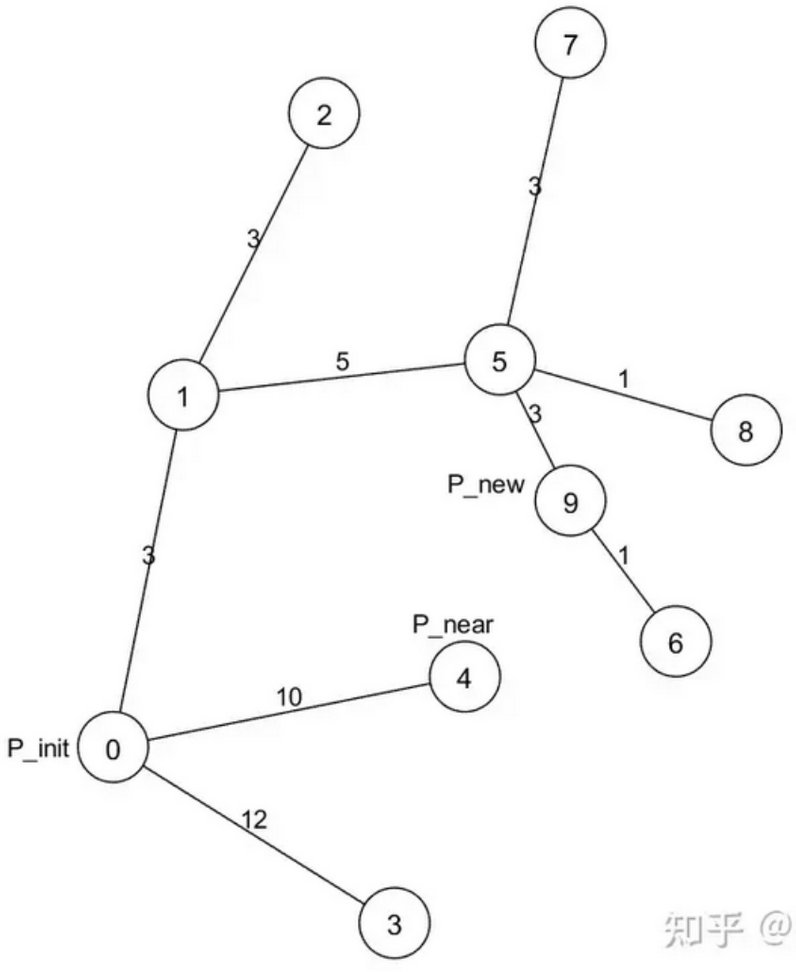
[IR艾若机器人的个人空间-IR艾若机器人个人主页-哔哩哔哩视频]

2、重新布线过程

在为P_new重新选择父节点之后，为进一步使得随机树节点间连接的代价尽量小，为随机树进行重新布线。过程也可以被表述成：如果近邻节点的父节点改为P_new节点可以减小路径代价，则进行更改。如下图：



这两个过程相辅相成，重新选择父节点使新生成的节点路径代价尽可能小，重布线使得生成新节点后的随机树减少冗余通路，减小路径代价。



路径规划算法--示例

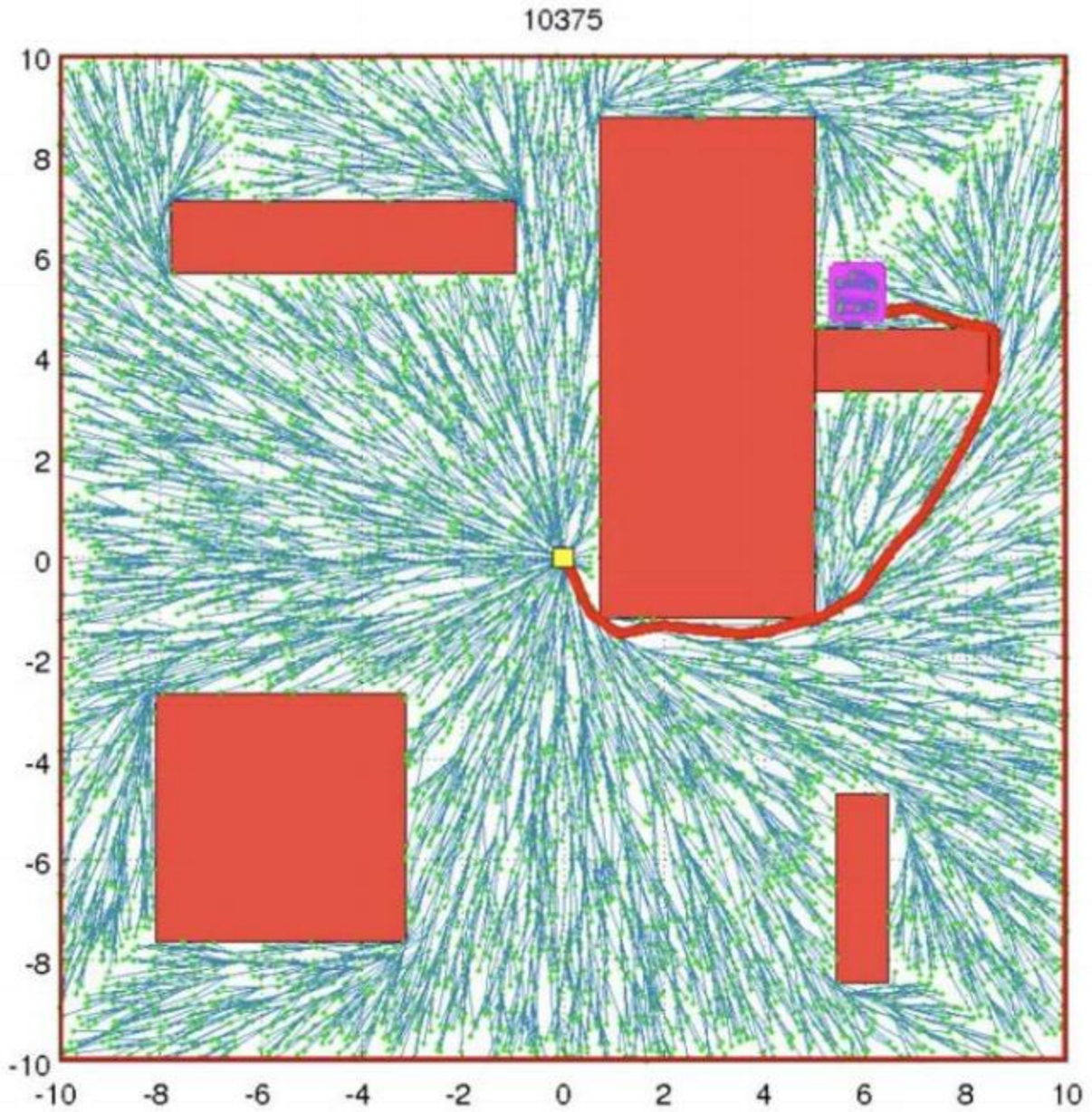
RRT*: Rapidly exploring Random Trees

Pros

- 寻找最优路径 (if one exists)

Cons

- Impractical running time *if asked for path with smooth trajectory* (Bry et al., IJRR '15)
- Jagged path *otherwise*



路径规划算法

RRT*: Rapidly exploring Random Trees

Pros

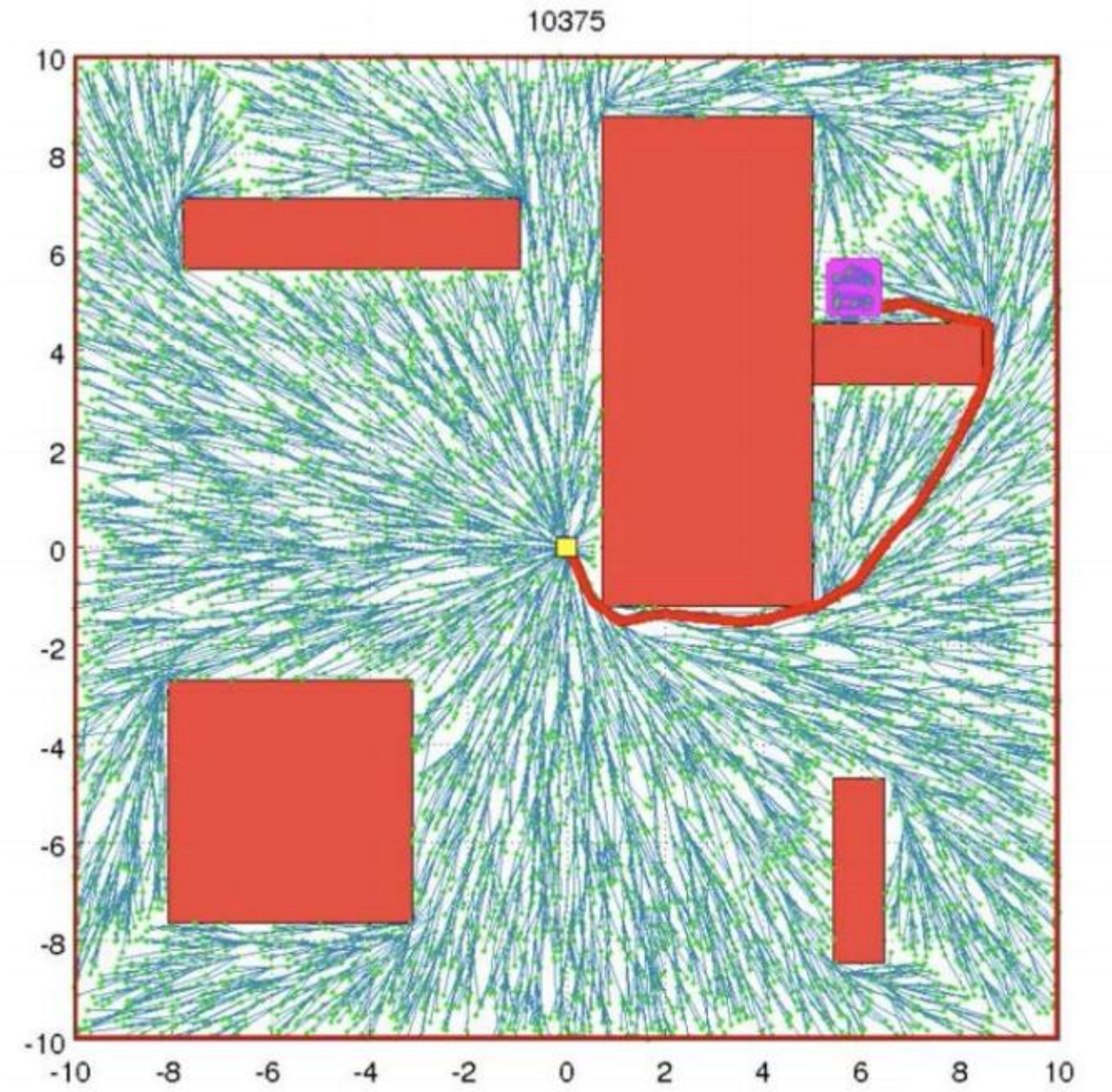
- Finds optimal path (if one exists)

Cons

- Impractical running time *if asked for path with smooth trajectory* (Bry et al., IJRR '15)

Difficult to apply for online planning in unknown/dynamic environments

- Jagged path *otherwise*



路径规划算法

RRT*: Rapidly exploring Random Trees

Pros

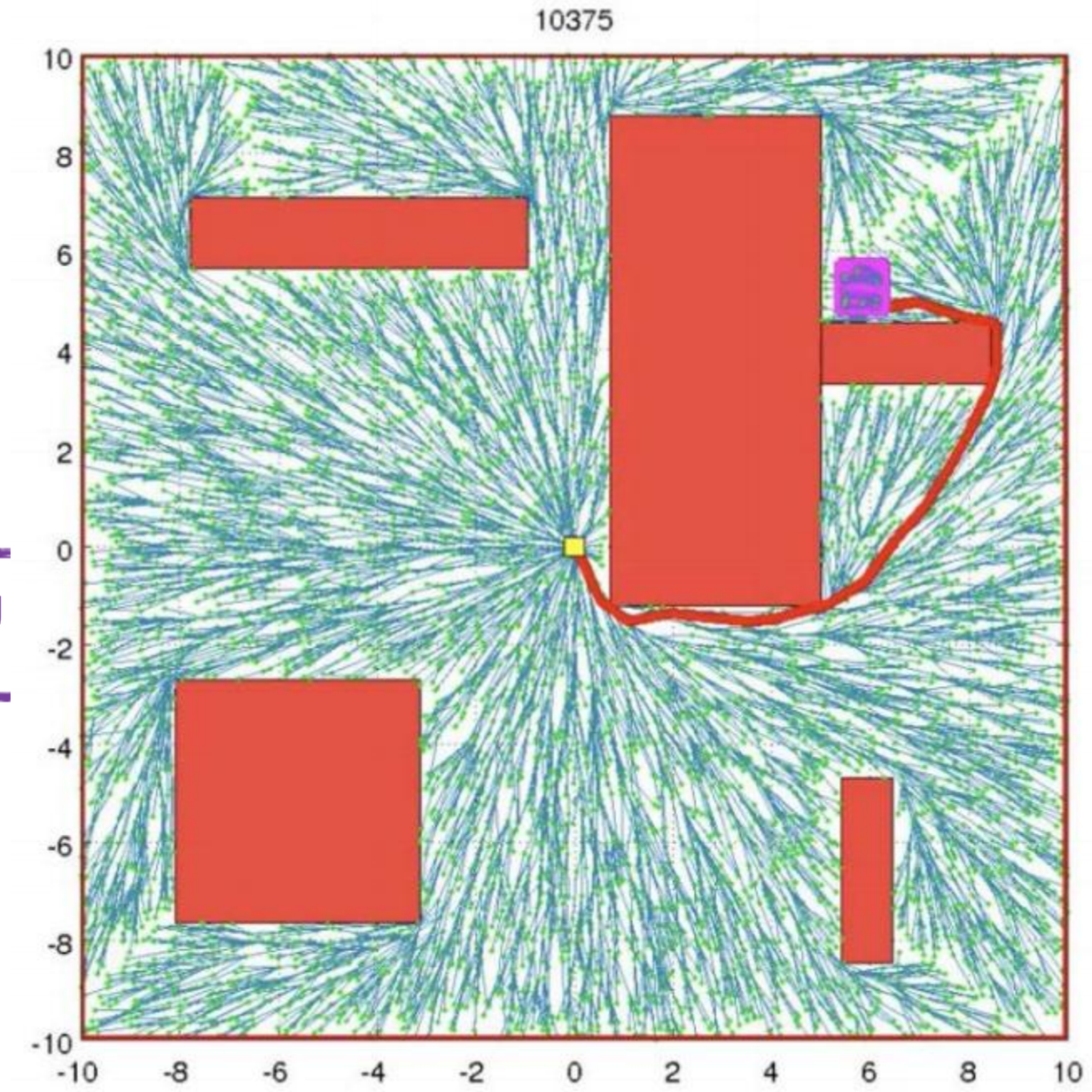
- Finds optimal path (if one exists)

Cons

- Impractical running time *if asked for path with smooth trajectory** (Bry et al., IJRR '15)

*the simultaneous path+trajectory planning is called **direct trajectory planning**

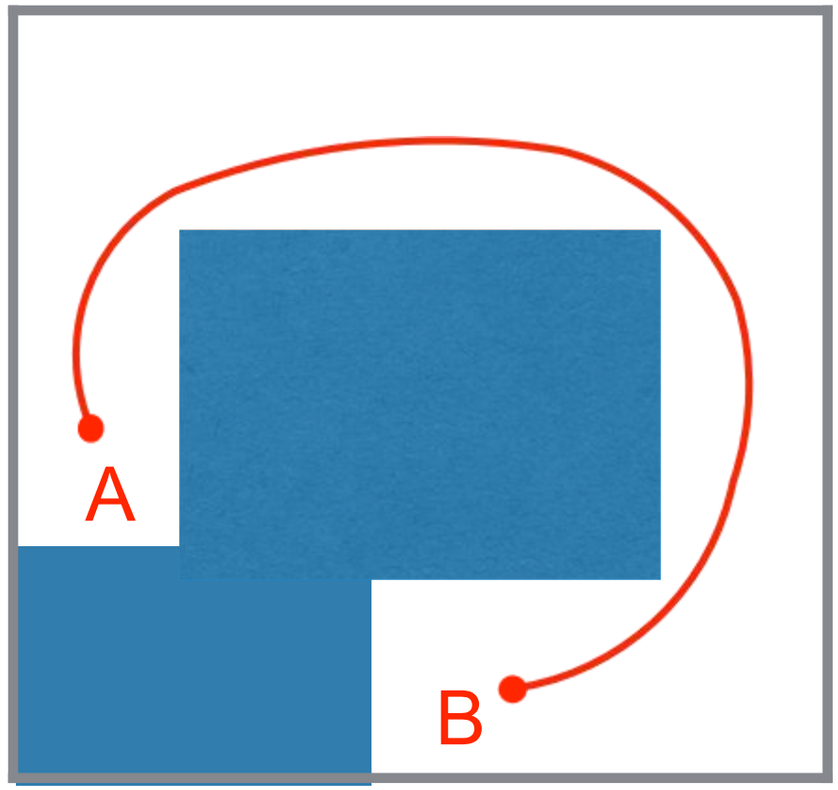
- Jagged path *otherwise*



[courtesy of Prof. Sertac Karaman]

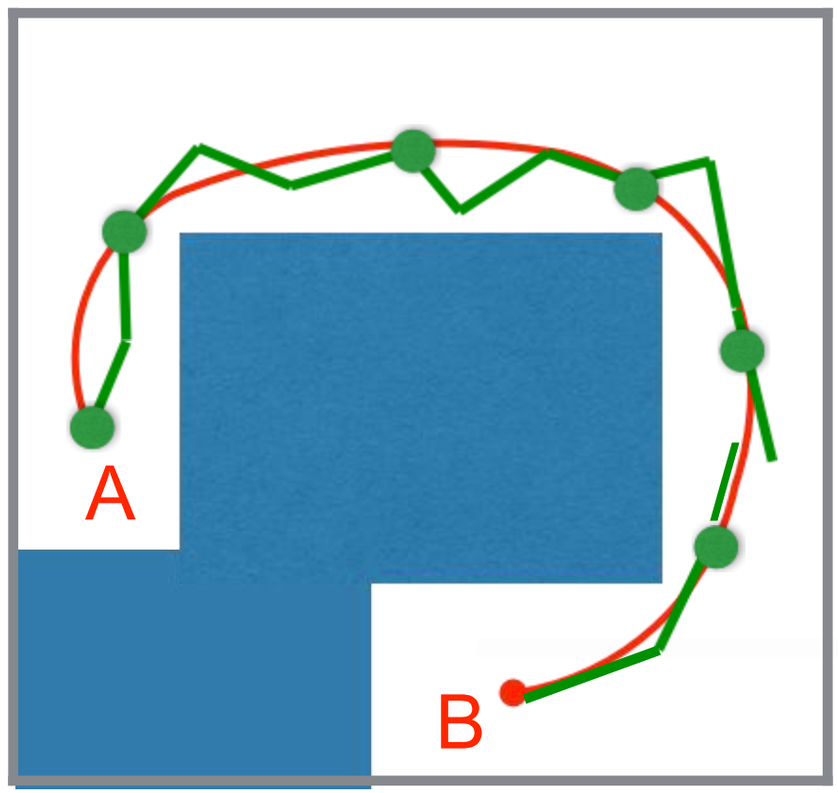
轨迹规划: Two Approaches

- **Direct trajectory planning**：同时规划路径并优化轨迹
  - 通常在线规划、快速动态规划时速度慢（difficult to account for obstacles)
  - 快速启发式：motion primitives(运动原语)



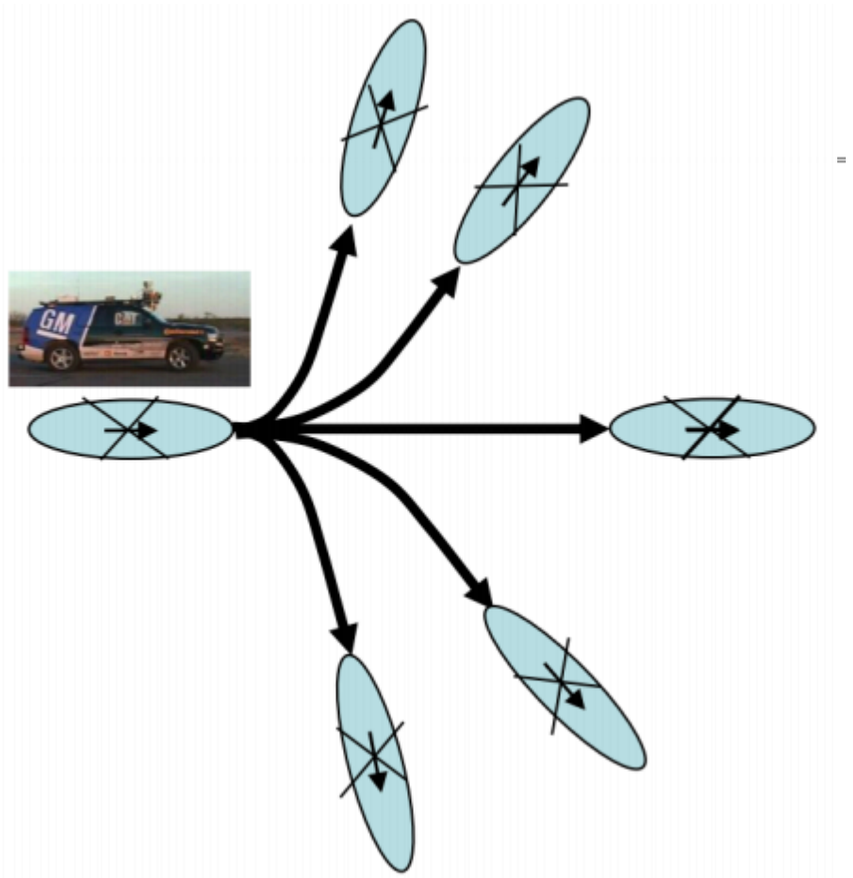
快速启发式通常指的是一种利用简化的方法来快速生成可行路径或轨迹的技术。通过预定义的简单动作组合来迅速找到解决方案，尽量减少计算复杂性，以应对动态环境中的实时规划需求。

- **Decoupled trajectory planning-解耦轨迹规划**
  首先是路径规划，然后是轨迹生成
  - 快速计算（仅部分考虑障碍物）
  - 多项式轨迹优化

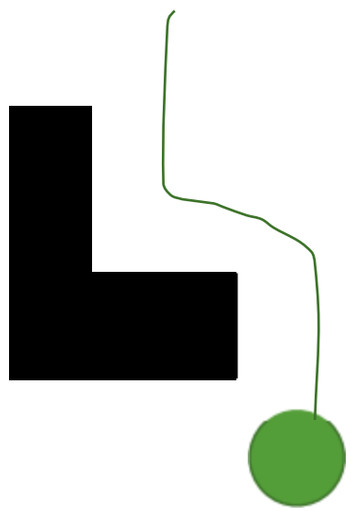


Motion Primitives 运动原语

- 输入：
  - 一个运动原语库：每个运动原语由对给定控制序列的系统进行前向仿真
  - 当前机器人状态
- 输出：一种运动原语选择，在避免障碍的同时最小化成本函数。



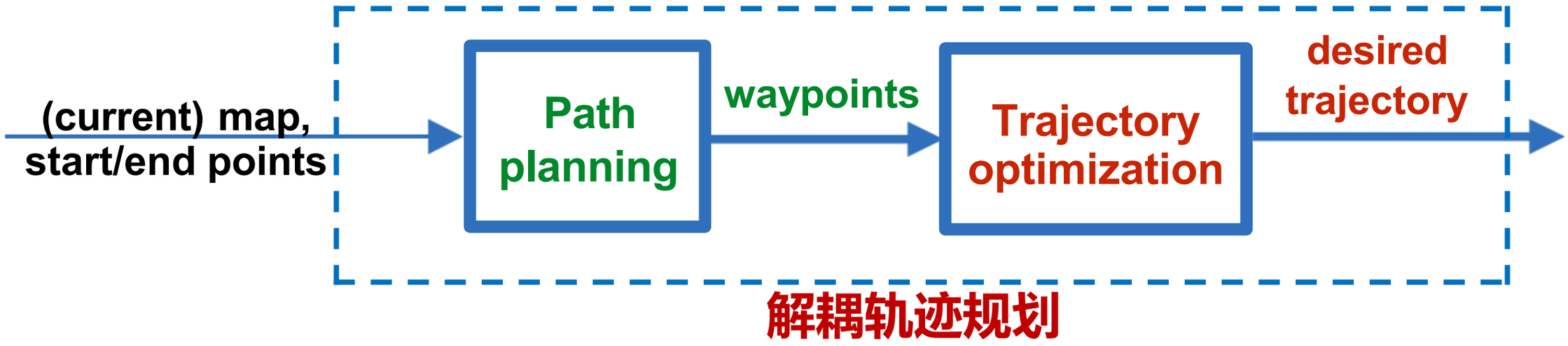
- The final trajectory is usually not smooth
- More complex instantiations: https://www.cs.cmu.edu/~maxim/files/tutorials/robschooltutorial_oct10.pdf



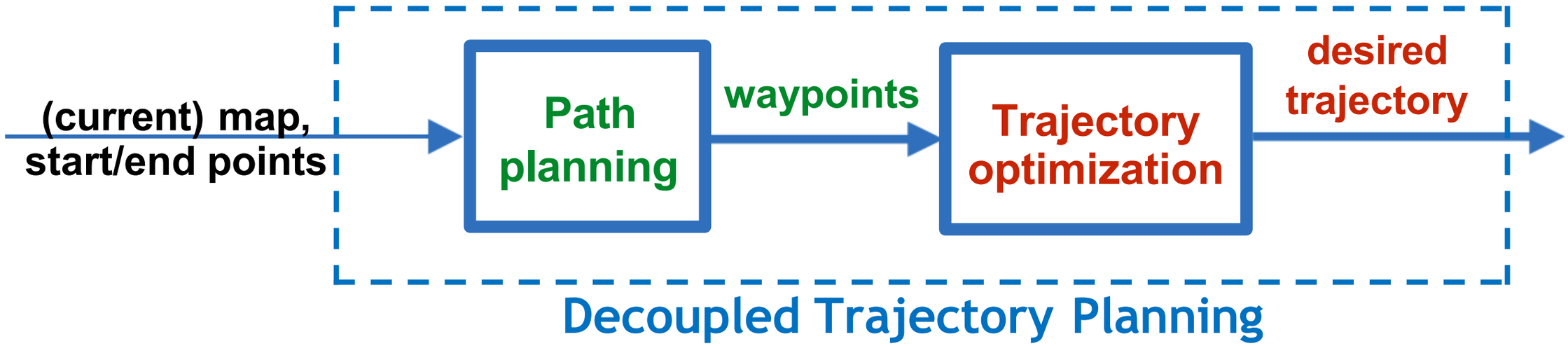
Number	Description
1	Straight
2	Climb
3	Takeoff (no throttle)
4	Gentle left
5	Gentle right
6	Left jog
7	Right jog

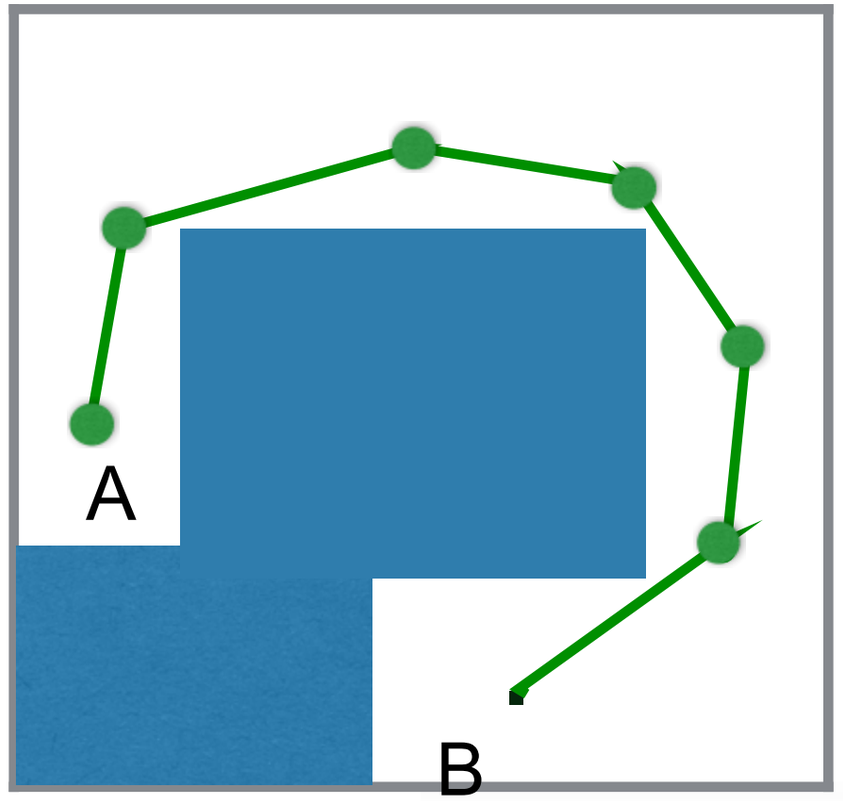
Output: 1->1>4->1->1->6

解耦轨迹规划

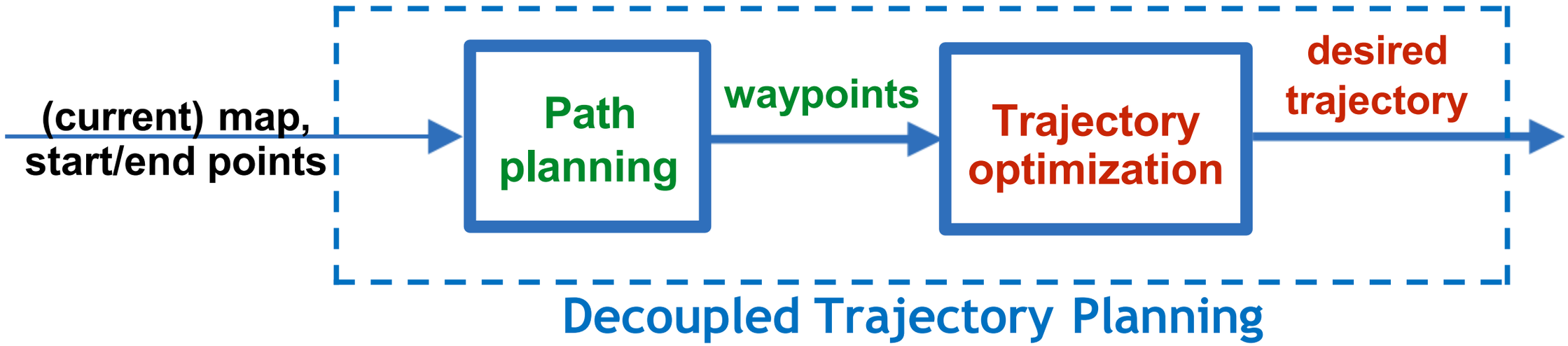

(current) map,
start/end points
Path planning
waypoints
Trajectory optimization
desired trajectory
解耦轨迹规划

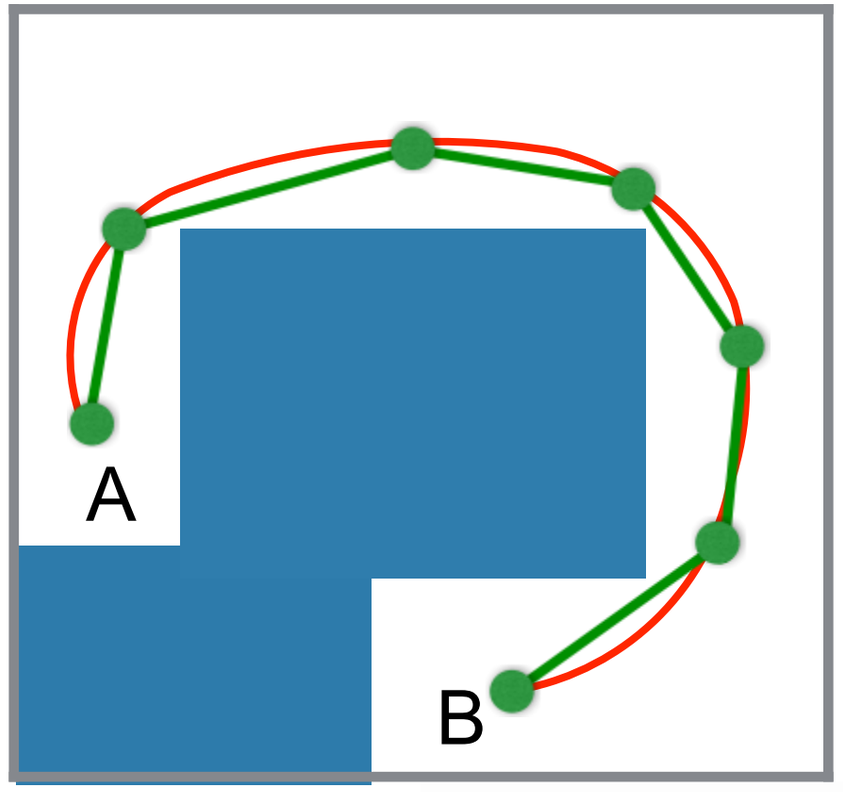
解耦轨迹规划


(current) map,
start/end points
Path planning
waypoints
Trajectory optimization
desired trajectory
Decoupled Trajectory Planning

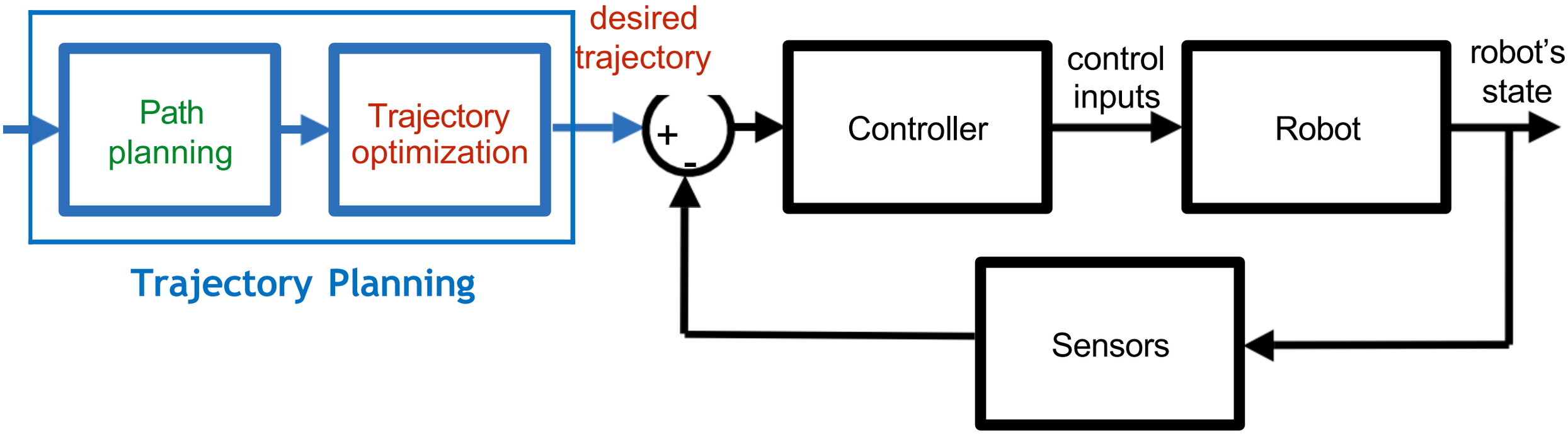

A
B

解耦轨迹规划


(current) map,
start/end points
Path planning
waypoints
Trajectory optimization
desired trajectory
Decoupled Trajectory Planning

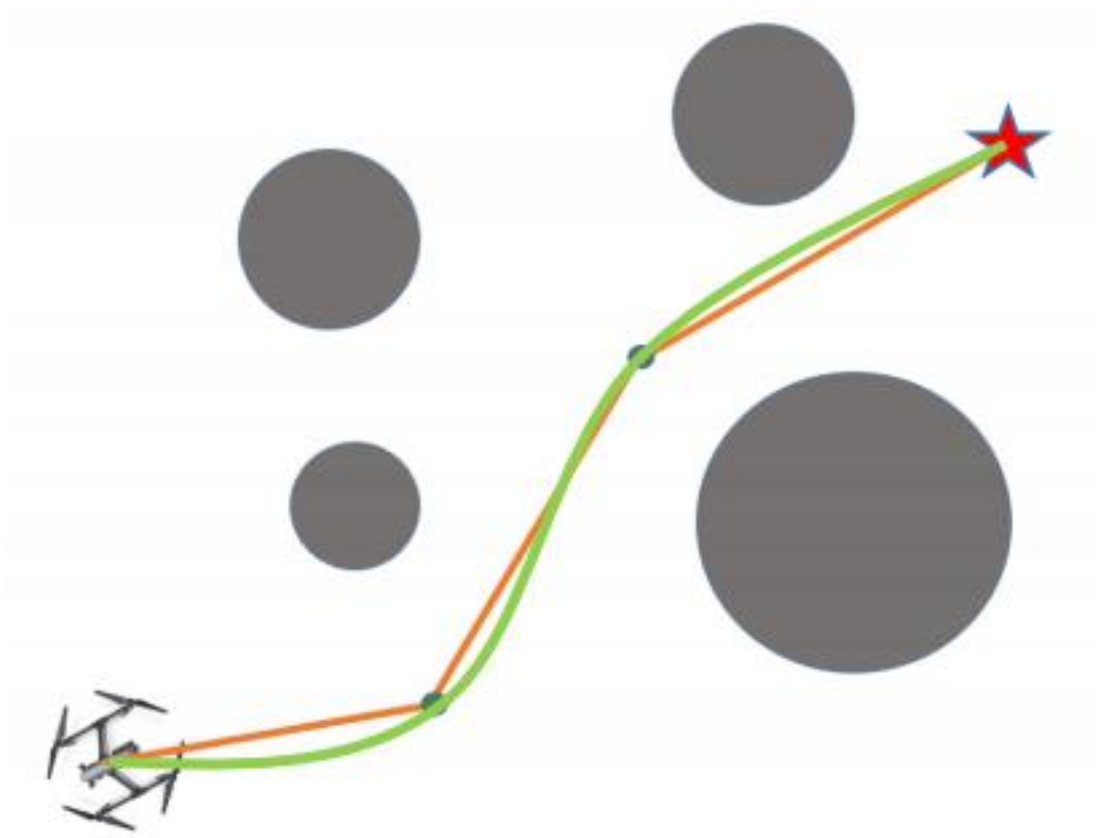

A
B

解耦轨迹规划


Path planning
Trajectory optimization
Trajectory Planning
desired trajectory
+
-
Controller
control inputs
Robot
robot's state
Sensors

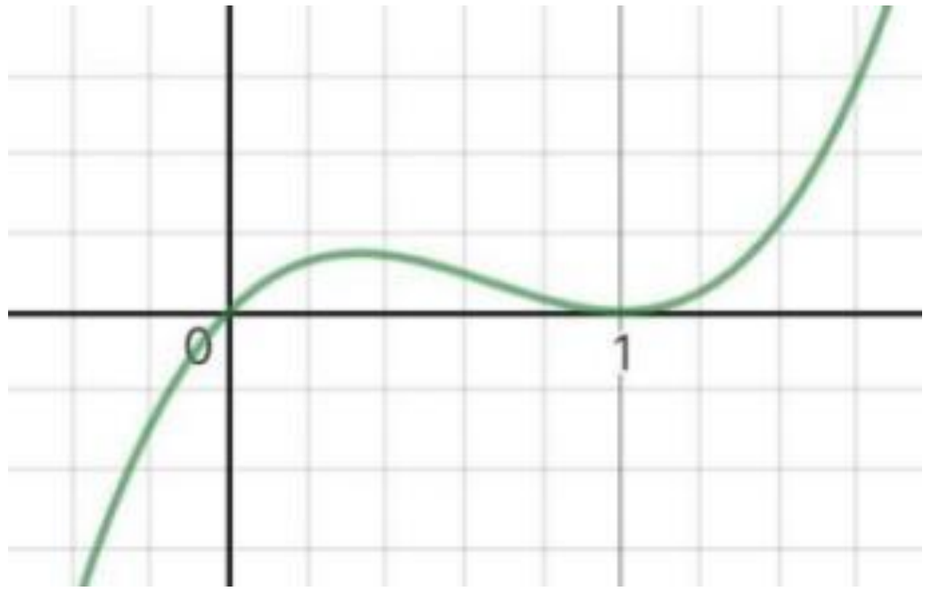
轨迹优化Trajectory Optimization

轨迹优化Trajectory Optimization



路径规划 + 轨迹优化

轨迹: 带时间参数的曲线

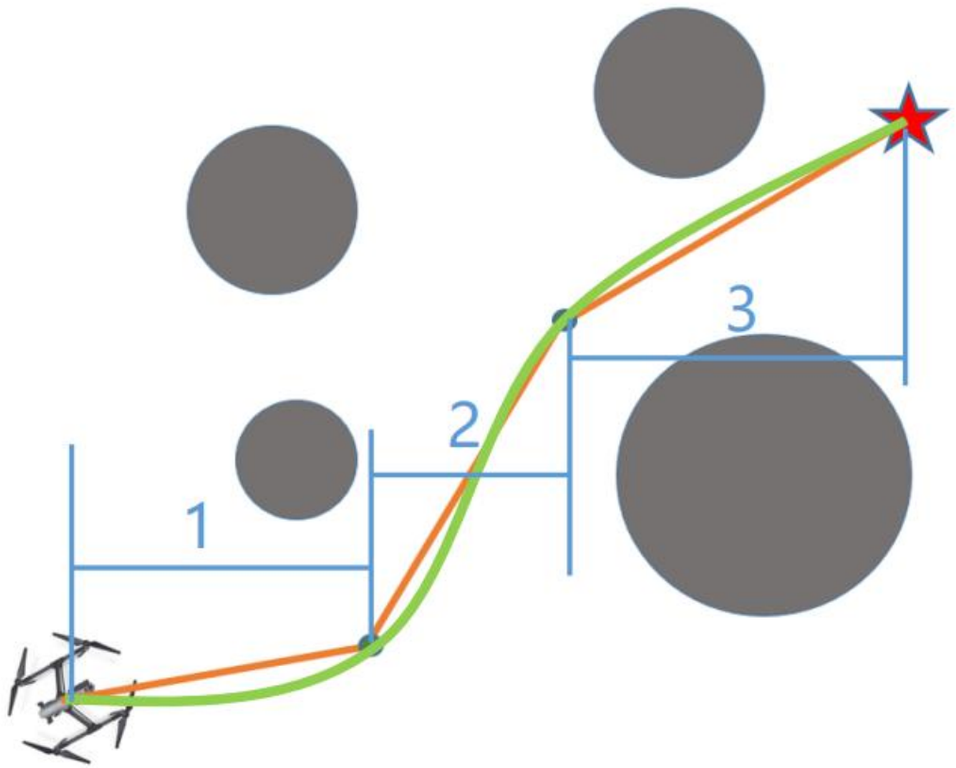


$$t^3 - 2t^2 + t$$

$$\begin{aligned} \min_{\boldsymbol{x}(t),\boldsymbol{u}(t)} \quad & J(\boldsymbol{u}(t),\boldsymbol{x}(t)) \\ \text{subject to} \quad & \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t),\boldsymbol{u}(t)) \\ & \boldsymbol{x}(0) = \bar{\boldsymbol{x}}_0 \\ & \boldsymbol{x}(T) = \bar{\boldsymbol{x}}_T \\ & \boldsymbol{u}_{min} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{max} \\ & \boldsymbol{x}_{min} \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_{max} \end{aligned}$$

轨迹优化Trajectory Optimization

多项式轨迹



时间t根据路径长度和平均速度分配

$$p(t) = p_0 + p_1 t + p_2 t^2 \ldots + p_n t^n = \sum_{i=0}^{n} p_i t^i$$

$$p(t) = [1, t, t^2, \ldots, t^n] \cdot p$$

$$p = [p_0, p_1, \ldots, p_n]^T$$

$$v(t) = p'(t) = [0, 1, 2t, 3t^2, 4t^3, \ldots, nt^{n-1}] \cdot p$$

$$a(t) = p''(t) = [0, 0, 2, 6t, 12t^2, \ldots, n(n-1)t^{n-2}] \cdot p$$

$$jerk(t) = p^{(3)}(t) = [0, 0, 0, 6, 24t, \ldots, \frac{n!}{(n-3)!} t^{n-3}] \cdot p$$

$$snap(t) = p^{(4)}(t) = [0, 0, 0, 0, 24, \ldots, \frac{n!}{(n-4)!} t^{n-4}] \cdot p$$

https://blog.csdn.net/q597967420/article/details/76099491

轨迹优化Trajectory Optimization

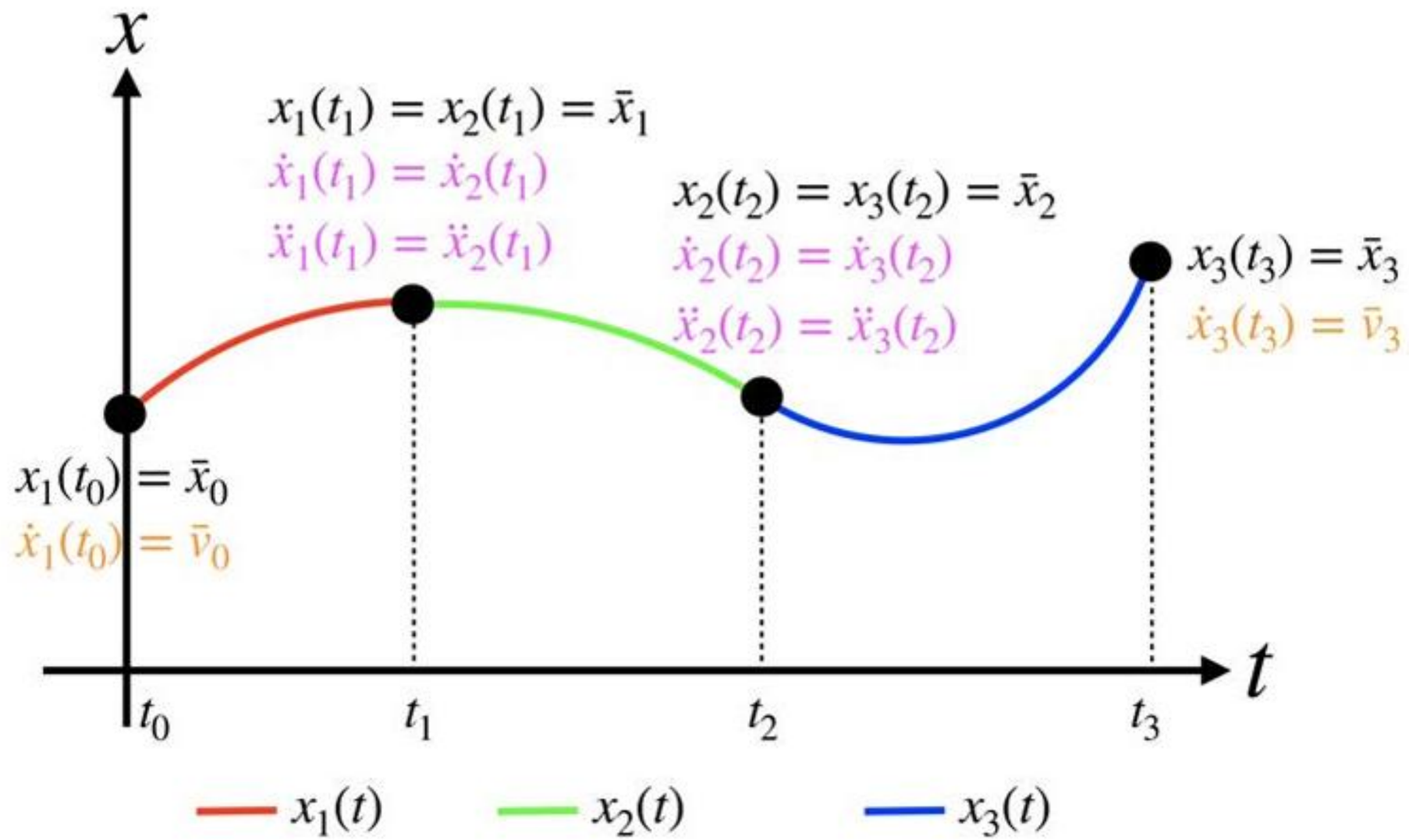
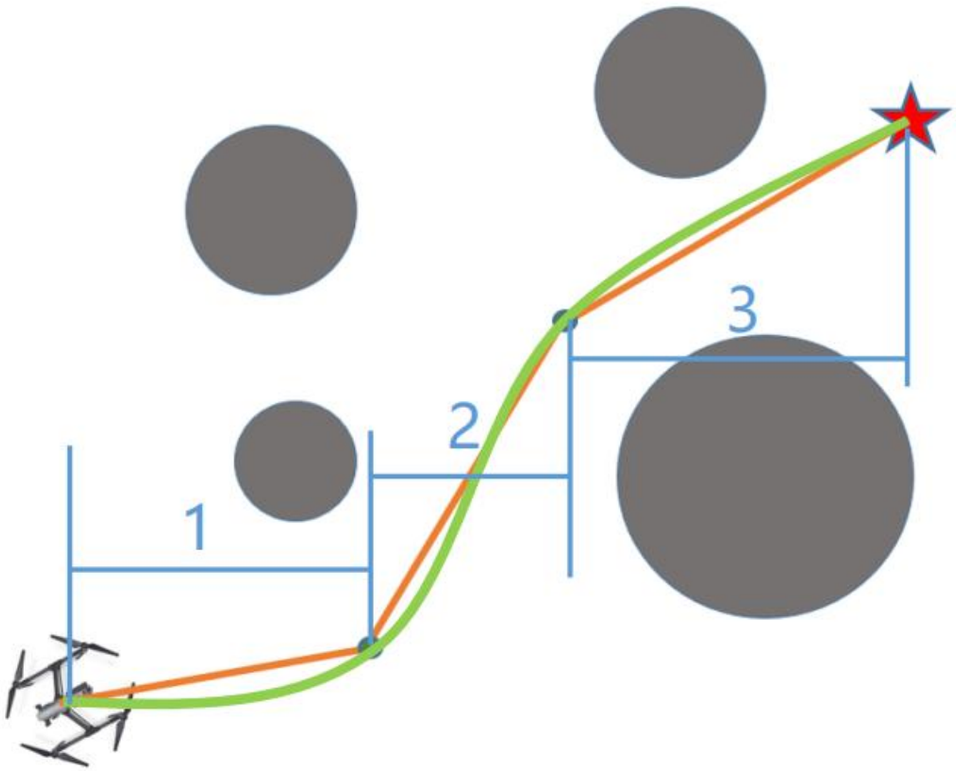


Figure 9.1: Multi-segment minimum acceleration ($r = 2$) trajectory optimization with 3 segments.

轨迹优化Trajectory Optimization

多项式轨迹



$$p(t) = \begin{cases} [1, t, t^2, \ldots, t^n] \cdot p_1 & t_0 \le t < t_1 \\ [1, t, t^2, \ldots, t^n] \cdot p_2 & t_1 \le t < t_2 \\ \ldots \\ [1, t, t^2, \ldots, t^n] \cdot p_k & t_{k-1} \le t < t_k \end{cases}$$

$$p_i = [p_{i0}, p_{i1}, \ldots, p_{in}]^T$$

$$p = [p_1^T, p_2^T, \ldots, p_k^T]^T$$

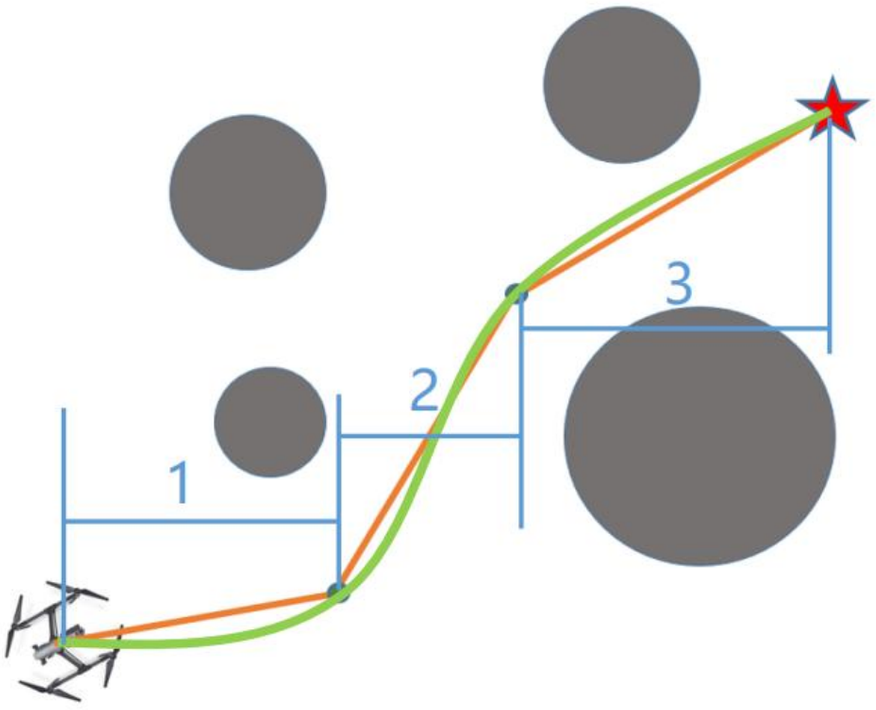
求解参数p，确定轨迹

时间t根据路径长度和平均速度分配

https://blog.csdn.net/q597967420/article/details/76099491

轨迹优化Trajectory Optimization

Minimum Snap



基本要求：

1. 两段轨迹之间连续
2. 轨迹经过固定点
3. 轨迹无碰撞

高级要求：

轨迹最顺滑、能量最优等

$$\min f(p)$$
$$s.t. \quad A_{eq}p = b_{eq},$$
$$A_{ieq}p \leq b_{ieq}$$

https://blog.csdn.net/q597967420/article/details/76099491

轨迹优化Trajectory Optimization

Minimum Snap

$$minimum\ snap:\ \ \min f(p) = \min(p^{(4)}(t))^2$$

$$minimum\ jerk:\ \ \min f(p) = \min(p^{(3)}(t))^2$$

$$minimum\ acce:\ \ \min f(p) = \min(p^{(2)}(t))^2$$

Minimum Snap Trajectory Generation and Control for Quadrotors (2011 ICRA)

轨迹优化Trajectory Optimization

Minimum Snap

！！索引从0开始

$$
\begin{aligned}
&\min \int_0^T (p^{(4)}(t))^2 \mathrm{d}t \\
&= \min \sum_{i=1}^{k} \int_{t_{i-1}}^{t_i} (p^{(4)}(t))^2 \mathrm{d}t \\
&= \min \sum_{i=1}^{k} \int_{t_{i-1}}^{t_i} ([0,0,0,0,24,\ldots,\frac{n!}{(n-4)!}t^{n-4}]\cdot p)^T [0,0,0,0,24,\ldots,\frac{n!}{(n-4)!}t^{n-4}]\cdot p \, \mathrm{d}t \\
&= \min \sum_{i=1}^{k} p^T \int_{t_{i-1}}^{t_i} [0,0,0,0,24,\ldots,\frac{n!}{(n-4)!}t^{n-4}]^T [0,0,0,0,24,\ldots,\frac{n!}{(n-4)!}t^{n-4}] \, \mathrm{d}t \, p \\
&= \min \sum_{i=1}^{k} p^T Q_i p
\end{aligned}
$$

$$
\begin{aligned}
Q_i &= \int_{t_{i-1}}^{t_i} [0,0,0,0,24,\ldots,\frac{n!}{(n-4)!}t^{n-4}]^T [0,0,0,0,24,\ldots,\frac{n!}{(n-4)!}t^{n-4}] \, \mathrm{d}t \\
&= \begin{bmatrix} 0_{4\times 4} & 0_{4\times(n-3)} \\ 0_{(n-3)\times 4} & \frac{r!}{(r-4)!}\frac{c!}{(c-4)!}\frac{1}{(r-4)+(c-4)+1}(t_i^{(r+c-7)} - t_{i-1}^{(r+c-7)}) \end{bmatrix}_{(n+1)*(n+1)}
\end{aligned}
$$

$$
Q = \begin{bmatrix} Q_1 & & & \\ & Q_2 & & \\ & & \ddots & \\ & & & Q_k \end{bmatrix}
$$

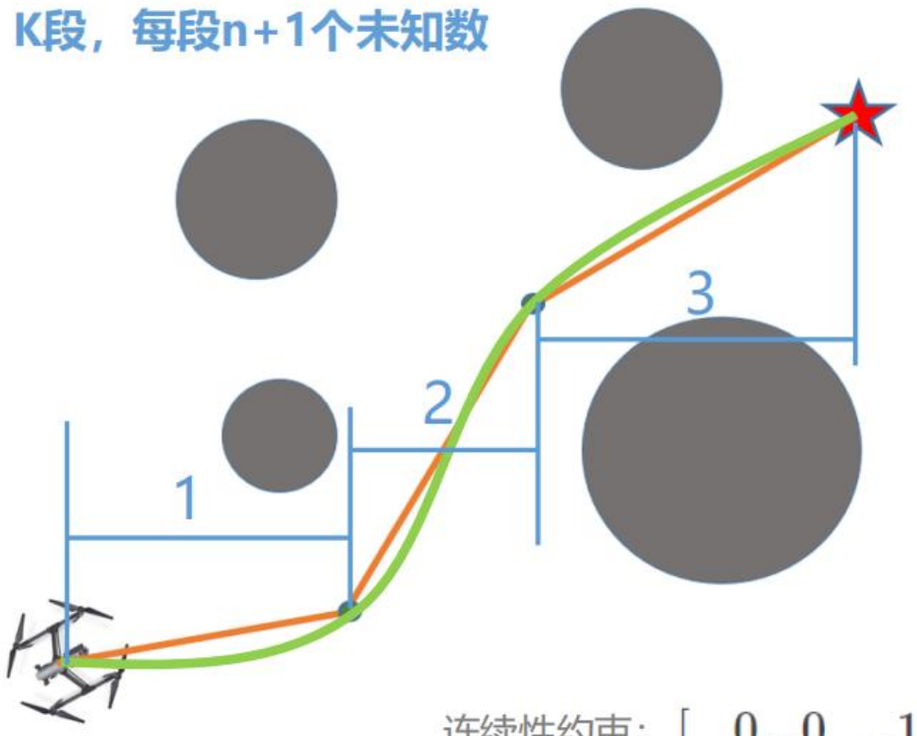
$$
\min p^T Q p
$$

Minimum Snap Trajectory Generation and Control for Quadrotors (2011 ICRA)

轨迹优化Trajectory Optimization

Minimum Snap

K段，每段n+1个未知数



$$\min f(p)$$

$$s.t.\quad A_{eq}p = b_{eq},$$

$$A_{ieq}p \le b_{ieq}$$

位置约束：$[1, t_0, t_0^2, \ldots, t_0^n, \underbrace{0 \ldots 0}_{(k-1)(n+1)}]p = p_0$

速度约束：$[0, 1, 2t_0, \ldots, nt_0^{n-1}, \underbrace{0 \ldots 0}_{(k-1)(n+1)}]p = v_0$

加速度约束：$[0, 0, 2, \ldots, n(n-1)t_0^{n-2}, \underbrace{0 \ldots 0}_{(k-1)(n+1)}]p = a_0$

连续性约束：$[\underbrace{0 \ldots 0}_{(i-1)(n+1)}, 1, t_i, t_i^2, \ldots, t_i^n, -1, -t_i, -t_i^2, \ldots, -t_i^n, \underbrace{0 \ldots 0}_{(k-i-1)(n+1)}]p = 0$

https://blog.csdn.net/q597967420/article/details/76099491

轨迹优化Trajectory Optimization

$$
\begin{bmatrix}
1, t_0, t_0^2, \ldots, t_0^n, & \underbrace{0 \ldots 0}_{(k-1)(n+1)} \\
0, 1, 2t_0, \ldots, nt_0^{n-1}, & \underbrace{0 \ldots 0}_{(k-1)(n+1)} \\
0, 0, 2, \ldots, n(n-1)t_0^{n-2}, & \underbrace{0 \ldots 0}_{(k-1)(n+1)} \\
\vdots & \\
\underbrace{0 \ldots 0}_{(i-1)(n+1)}, 1, t_i, t_i^2, \ldots, t_i^n, & \underbrace{0 \ldots 0}_{(k-i)(n+1)} \\
\vdots & \\
\underbrace{0 \ldots 0}_{(k-1)(n+1)}, 1, t_k, t_k^2, \ldots, t_k^n & \\
\underbrace{0 \ldots 0}_{(k-1)(n+1)}, 0, 1, 2t_k, \ldots, nt_k^{n-1} & \\
\underbrace{0 \ldots 0}_{(k-1)(n+1)}, 0, 0, 2, \ldots, n(n-1)t_k^{n-2} & \\
\underbrace{0 \ldots 0}_{(i-1)(n+1)}, 1, t_i, t_i^2, \ldots, t_i^n, -1, -t_i, -t_i^2, \ldots, -t_i^n, & \underbrace{0 \ldots 0}_{(k-i-1)(n+1)} \\
\underbrace{0 \ldots 0}_{(i-1)(n+1)}, 0, 1, 2t_i, \ldots, nt_i^{n-1}, -0, -1, -2t_i, \ldots, -nt_i^{n-1}, & \underbrace{0 \ldots 0}_{(k-i-1)(n+1)} \\
\underbrace{0 \ldots 0}_{(i-1)(n+1)}, 0, 0, 2, \ldots, \frac{n!}{(n-2)!}t_i^{n-2}, -0, -0, -2, \ldots, -\frac{n!}{(n-2)!}t_i^{n-2}, & \underbrace{0 \ldots 0}_{(k-i-1)(n+1)}
\end{bmatrix}_{(4k+2)\times(n+1)k}
$$

$$
p = \begin{bmatrix} p_0 \\ v_0 \\ a_0 \\ \vdots \\ p_i \\ \vdots \\ p_k \\ v_k \\ a_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}
$$

等式约束个数：

3 (起点PVA)

k-1 (中间固定点的P)

3 (终点PVA)

3(k-1) (中间点PVA连续)

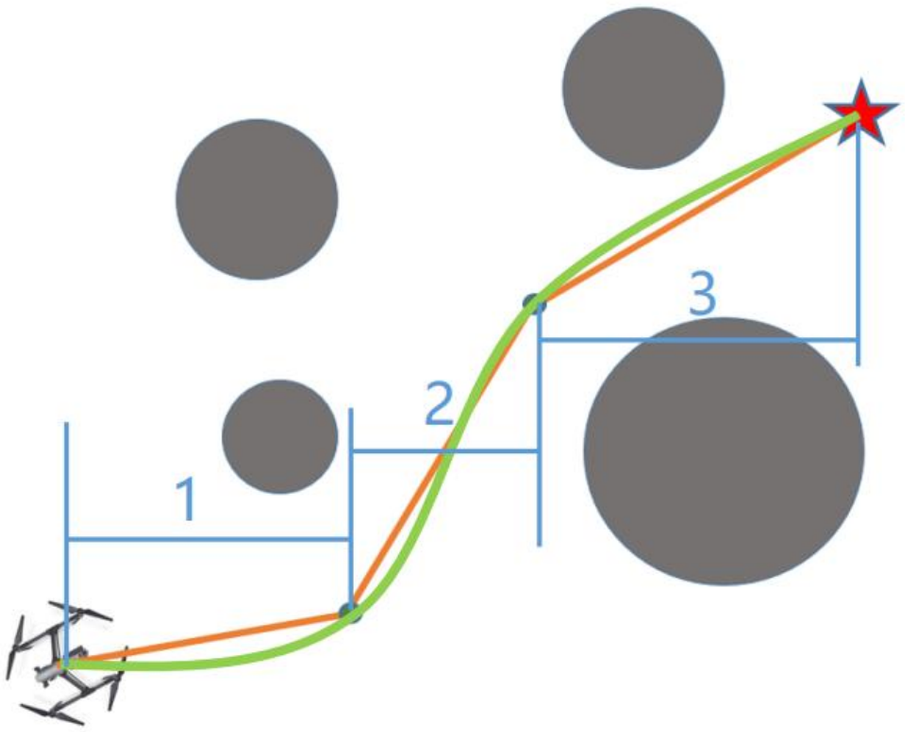
4k+2

$$A_{eq}p = b_{eq}$$

Minimum Snap Trajectory Generation and Control for Quadrotors (2011 ICRA)

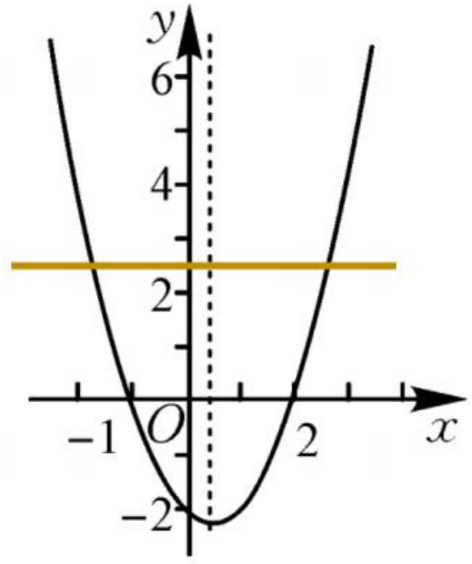
轨迹优化Trajectory Optimization

从理论到代码



$$\min p^T Qp$$
$$s.t. \quad A_{eq}p = b_{eq},$$

二次规划问题



https://blog.csdn.net/u011231598/article/details/79615321

16.485 Visual Navigation for Autonomous Vehicles (VNAV)
Fall 2020