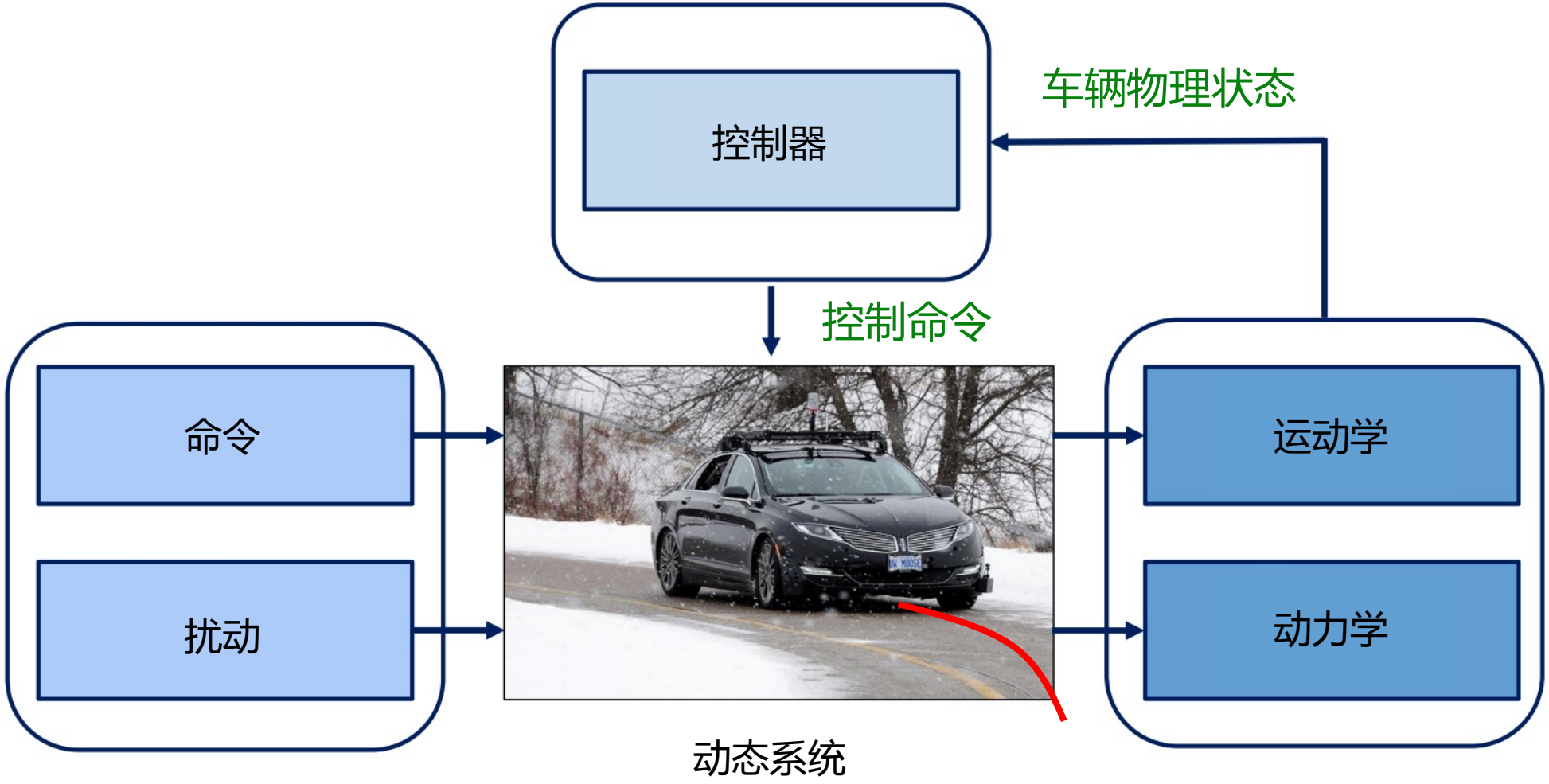
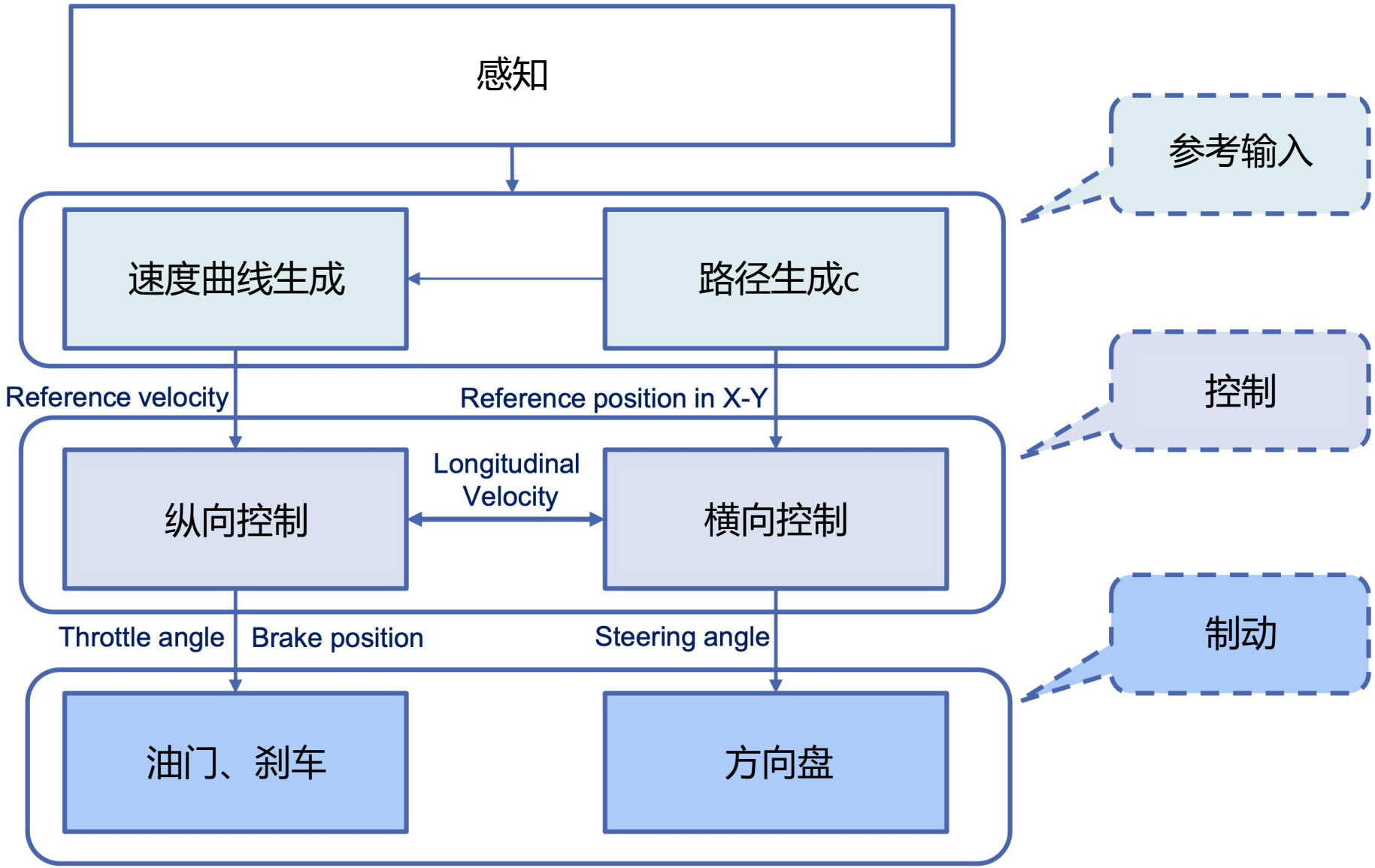
CS7355
自动驾驶前沿技术

2024年4月10日

车辆控制



车辆运动控制总体流程


感知
参考输入
速度曲线生成
路径生成c
Reference velocity
Reference position in X-Y
控制
纵向控制
Longitudinal Velocity
横向控制
Throttle angle
Brake position
Steering angle
制动
油门、刹车
方向盘

车辆最优控制 — MPC

$$\min_{U \triangleq \{u_{t|t}, u_{t+1|t}, \dots\}} J(x(t), U) = \sum_{j=t}^{t+T} C(x_{j|t}, u_{j|t})$$

（二次）目标函数（轨迹追踪误差+控制成本）

优化变量：t 到 t+T时间的控制序列

$$s.t. \quad x_{j+1|t} = f(x_{j|t}, u_{j|t}), \qquad t \le j \le t+T-1$$

车辆动力学公式

$$x_{min} \le x_{j+1|t} \le x_{max}, \qquad t \le j \le t+T-1$$

可行状态限制

$$u_{min} \le u_{j|t} \le u_{max}, \qquad t \le j \le t+T-1$$

可行控制限制

$$g(x_{j|t}, u_{j|t}) \le 0, \qquad t \le j \le t+T-1$$

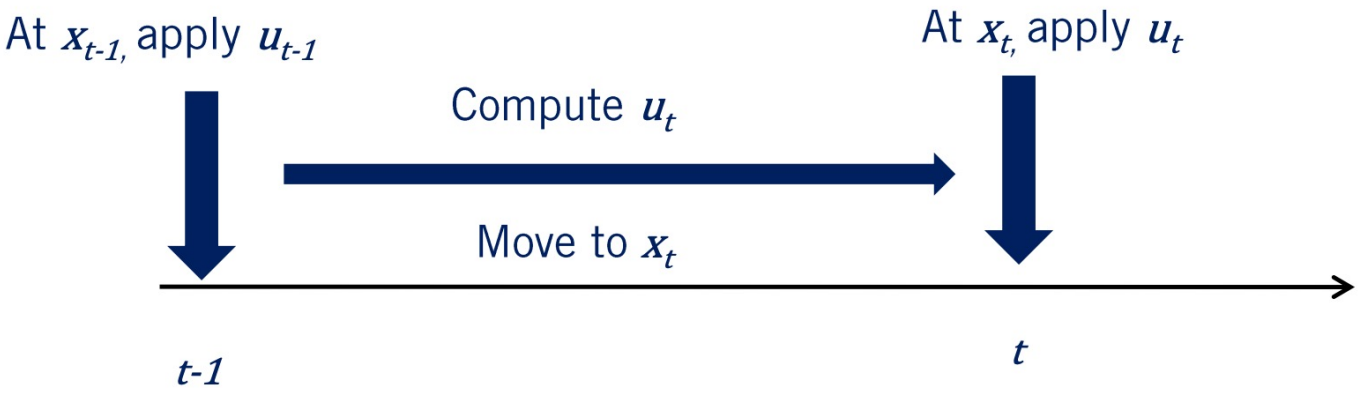
$$h(x_{j|t}, u_{j|t}) = 0, \qquad t \le j \le t+T-1$$

更多底层动力学限制（与打滑、倾倒、引擎动力等相关）

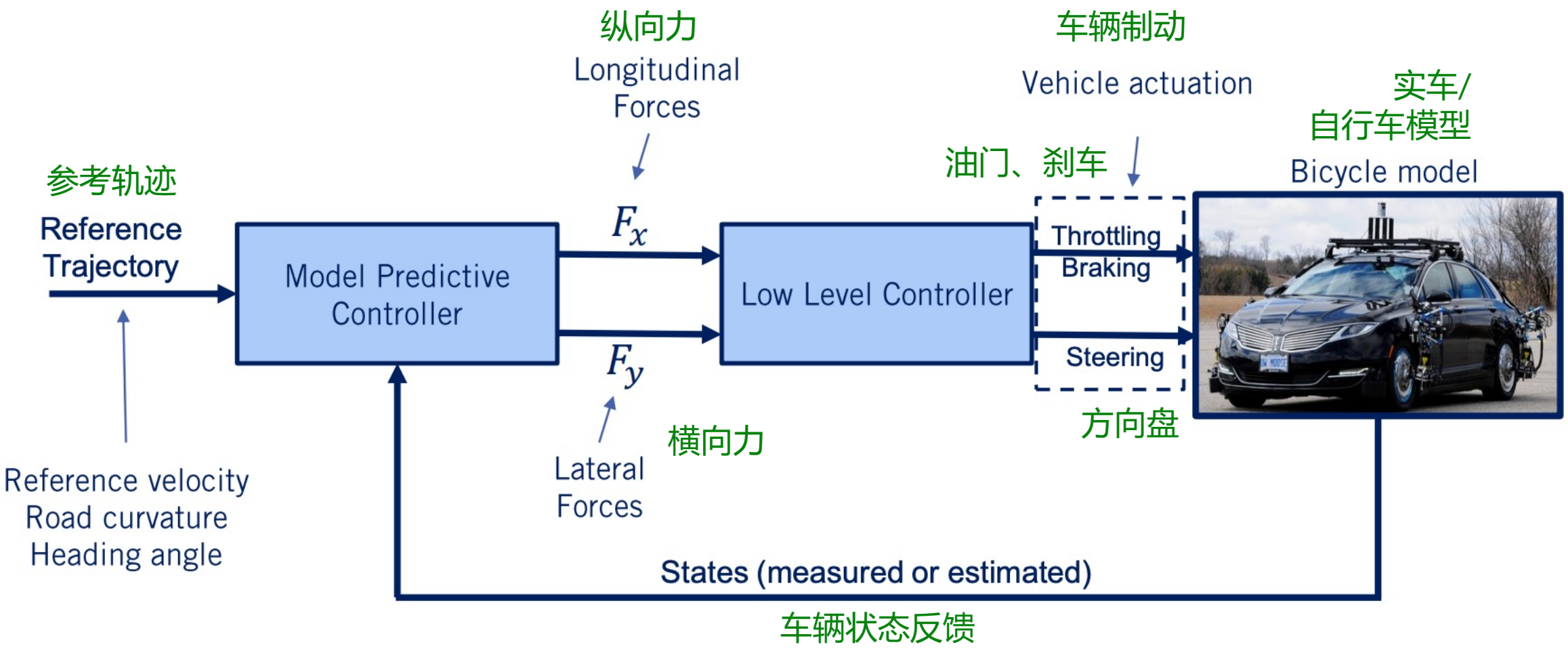
- 上述约束优化问题通常为非线性问题，需要数值方法求解
  - 二次规划（线性约束）、序列二次规划（非线性约束）
  - 梯度下降、共轭梯度下降（局部最优）
  - 对偶法（约束优化）、内点法（大规模问题）

滚动时域控制（RHC）

- 选择时域长度（Horizon，记作 H）
- 对于当前时间步 $t-1$：
  - 利用 u_{t-1}，x_{t-1} 与车辆动力学模型，预测一个时间步之后状态 x_t
  - 同时进行：
    - （Move to x_t）执行 u_{t-1}, 使车辆从状态 x_{t-1} 运动到状态 x_t
    - （Compute u_t）针对 x_t，优化从时间 t 到 $t+H$ 期间的控制序列 $u_t, \ldots, u_{t+H}$，输出第一个 u_t

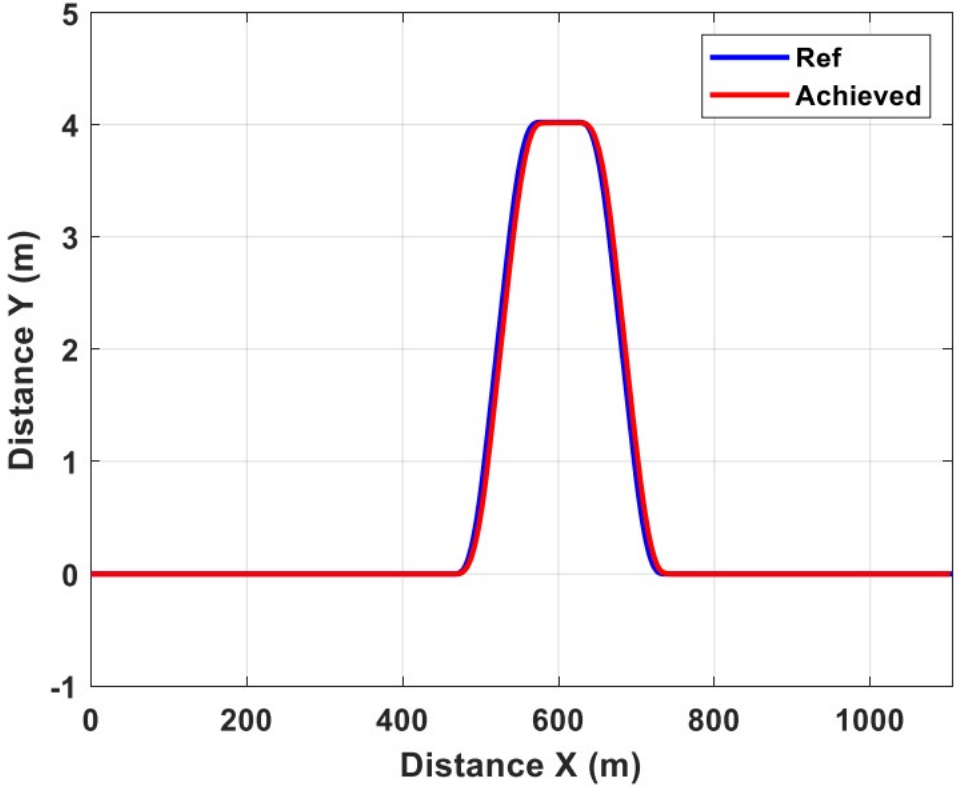


基于MPC的车辆横纵向控制

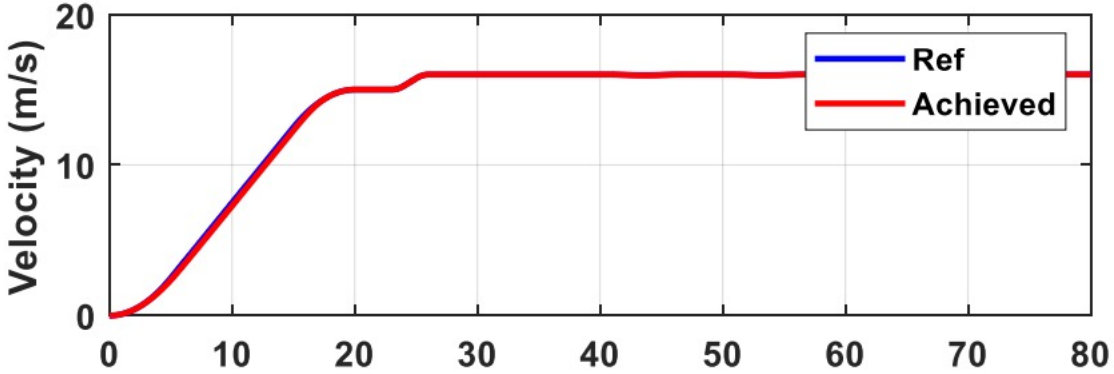


MPC控制效果—来回变道

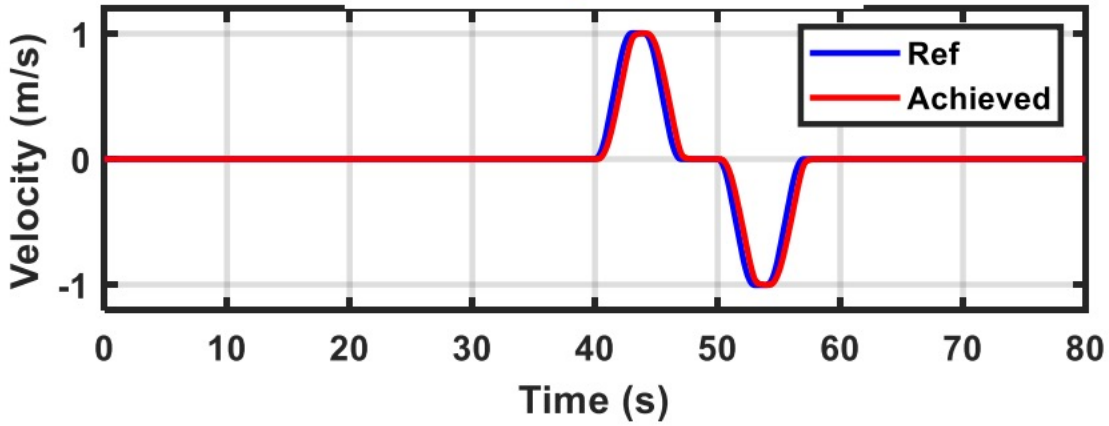
横向运动（位置）
参考vs.实际



纵向运动（速度）
参考vs.实际



横向运动（速度）

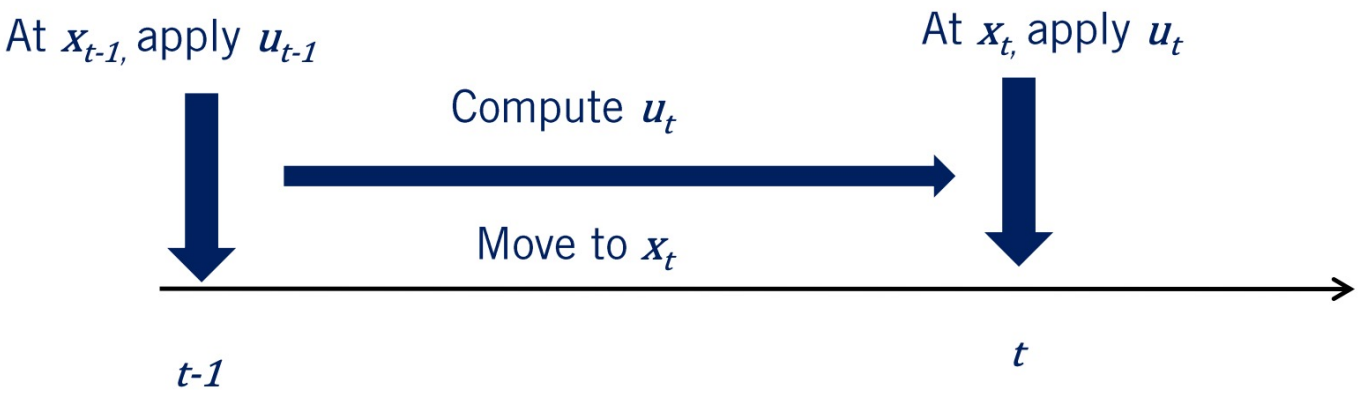


MPC 越野赛车手

Model Predictive Path Integral Control (MPPI)

- **MPPI是基于采样的MPC算法**
- 使用经典的滚动时域控制（RHC）框架

  - 选择时域长度（Horizon，记作 H）
  - 对于当前时间步 $t-1$：
    - 利用 u_{t-1}，x_{t-1} 与车辆动力学模型，预测一个时间步之后状态 x_t
    - 同时进行：
      - （Move to x_t）执行 u_{t-1}, 使车辆从状态 x_{t-1} 运动到状态 x_t
      - **（Compute u_t）**针对 x_t，优化从时间 t 到 $t+H$ 期间的控制序列 $u_t, \dots, u_{t+H}$，输出第一个 u_t



Model Predictive Path Integral Control (MPPI)

- （Compute u_t）MPPI
  - 重复 N 次：
    - 从高斯先验分布随机采样 K 个控制序列: $\boldsymbol{u^k} = \{u_t^k, u_{t+1}^k, \dots, u_{t+H}^k\}$，其中 $u_{t+1}^k \sim \mathcal{N}(\bar{\mu}_t, \Sigma)$
    - 对于每一个序列：
      - 利用模型 $x_{t+1} = F(x_t, u_t)$ 预测未来的状态轨迹: $\boldsymbol{\tau^k} = \{x_t^k, x_{t+1}^k, \dots, x_{t+H}^k\}$
      - 评估每条轨迹的 cost : $\boldsymbol{S}(\boldsymbol{\tau^k}) = \sum_{i=t}^{t+H} C(x_i^k, u_i^k)$
    - 依据 cost 的加权平均，更新高斯分布的均值 $\{\bar{\mu}_t, \overline{\mu_{t+1}}, \dots, \overline{\mu_{t+H}}\}$.
  - 输出 $u_t = \bar{\mu}_t$， $\{\overline{\mu_{t+1}}, \dots, \overline{\mu_{t+H}}\}$ 成为下一步的先验高斯分布

Model Predictive Path Integral Control (MPPI)

- 均值更新规则：

$$\bar{u} = \bar{u} + \frac{\sum_k w_k \delta u_k}{\sum_k w_k}$$

实际采样的是加在均值上的控制噪音序列

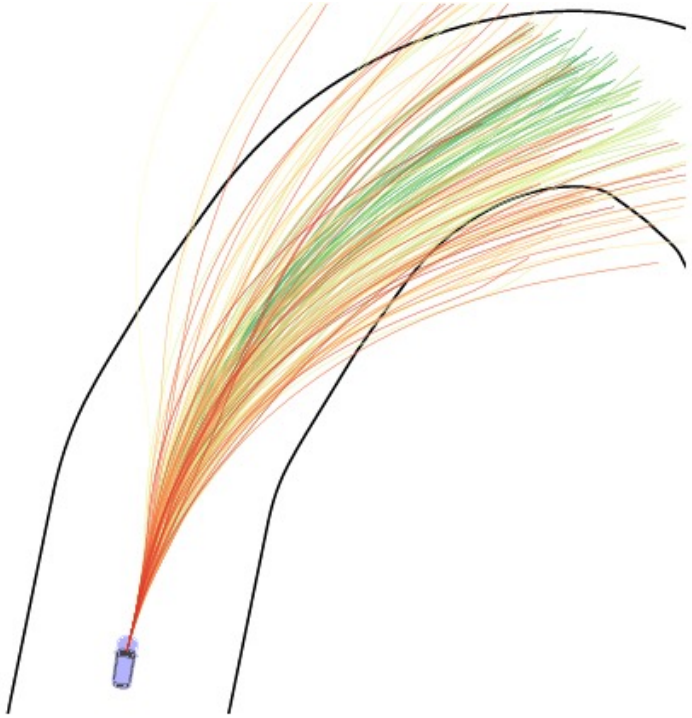
每个噪音序列赋予一个权重，表示其 goodness

- 权重计算：

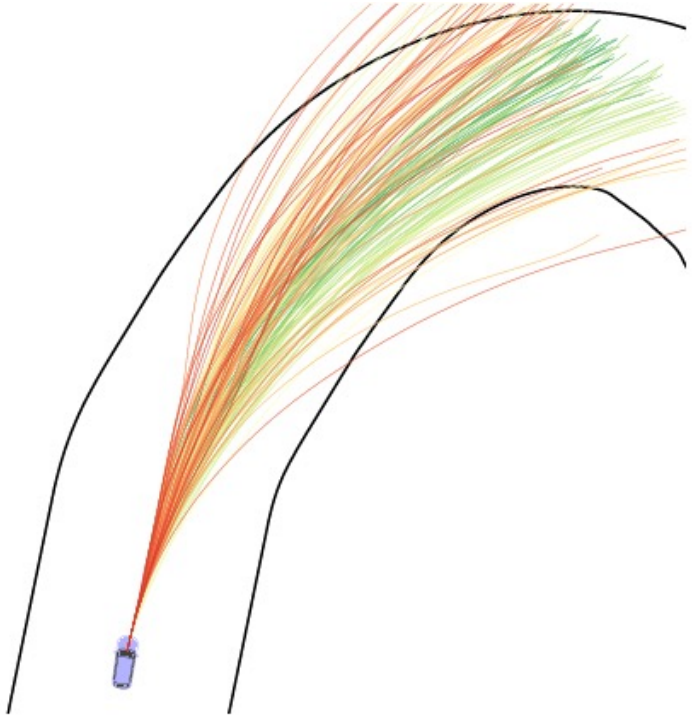
$$w_k = \mathrm{e}^{-\frac{1}{\lambda} S_k}$$

噪音序列的权重与所生成的轨迹的 cost 呈负指数关系

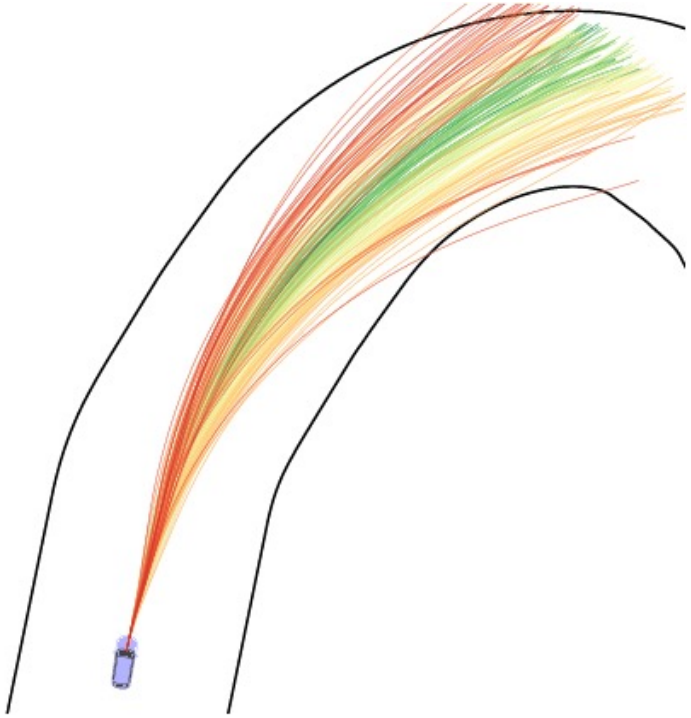
MPPI 最优控制迭代



初始分布



3次迭代之后



8次迭代之后

MPPI中的模型？

- 车辆状态（含一阶导数版）

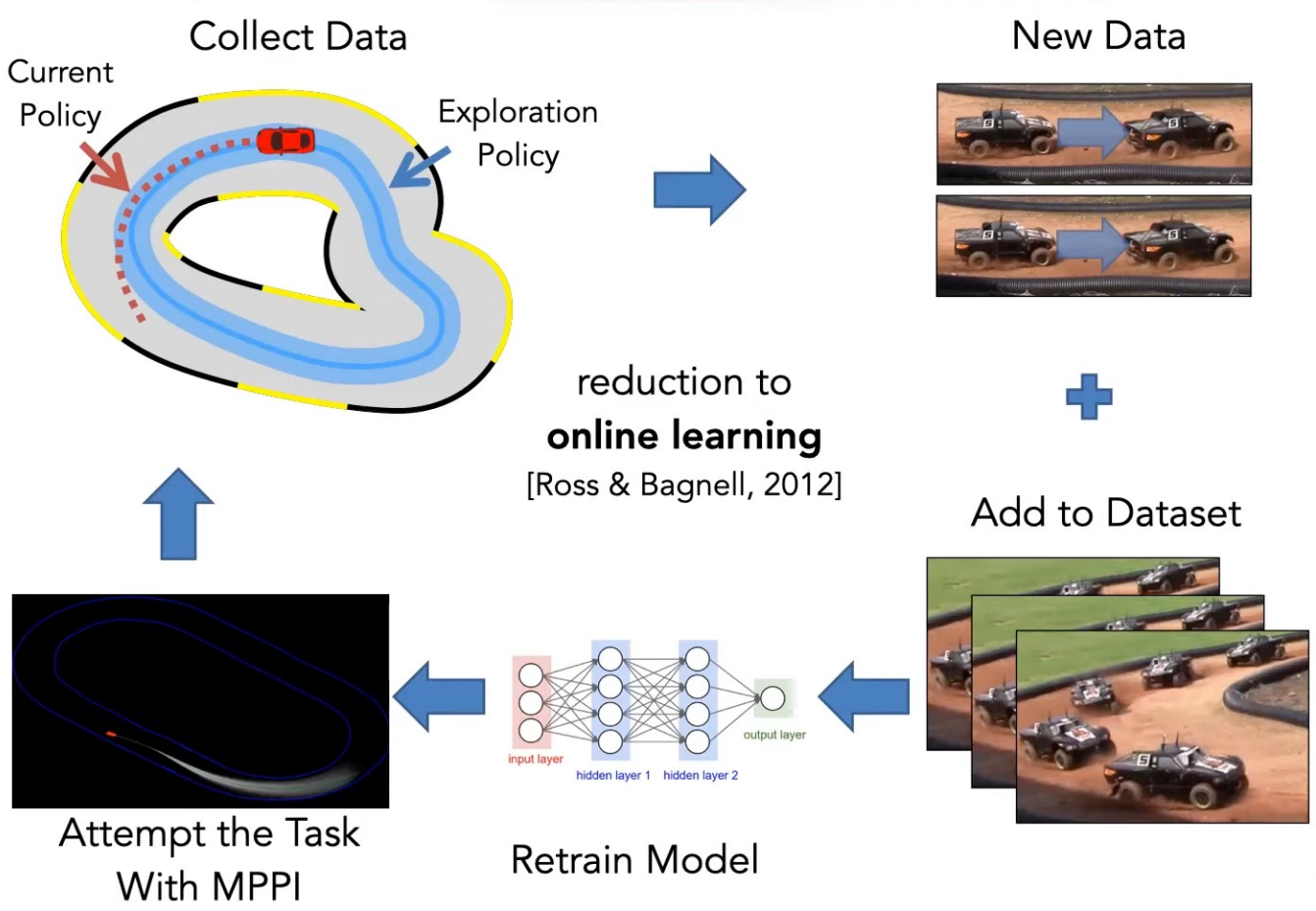
$$\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}})$$

- 车辆动力学模型（离散时间版）

$$\mathbf{x}_{t+1} = \mathbf{F}(\mathbf{x}_t, \mathbf{u}_t) = \begin{bmatrix} \mathbf{q}_t + \dot{\mathbf{q}}_t \Delta t \\ \dot{\mathbf{q}}_t + \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)\Delta t \end{bmatrix}$$

- **$f(x, u)$ 如何获得？**
  - 手写动力学模型？

学习动力学模型



DAgger 在线学习框架

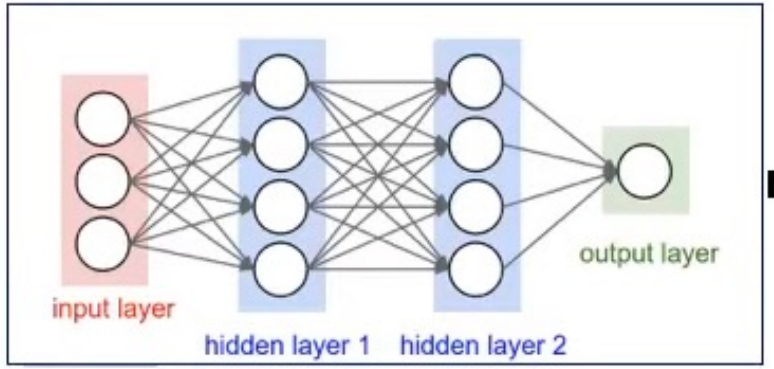
Input: Task, N: Iterations, M: Trials per iteration
$\mathcal{D} \leftarrow$ CollectBootstrapData();
for $i \leftarrow 1$ **to** N **do**
  $\mathbf{F} \leftarrow$ Train($\mathcal{D}$);
  for $j \leftarrow 0$ **to** M **do**
    $\mathcal{D}_j \leftarrow$ MPPI($\mathbf{F}$, Task) ;
    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_j$

MPC + NN 动力学模型

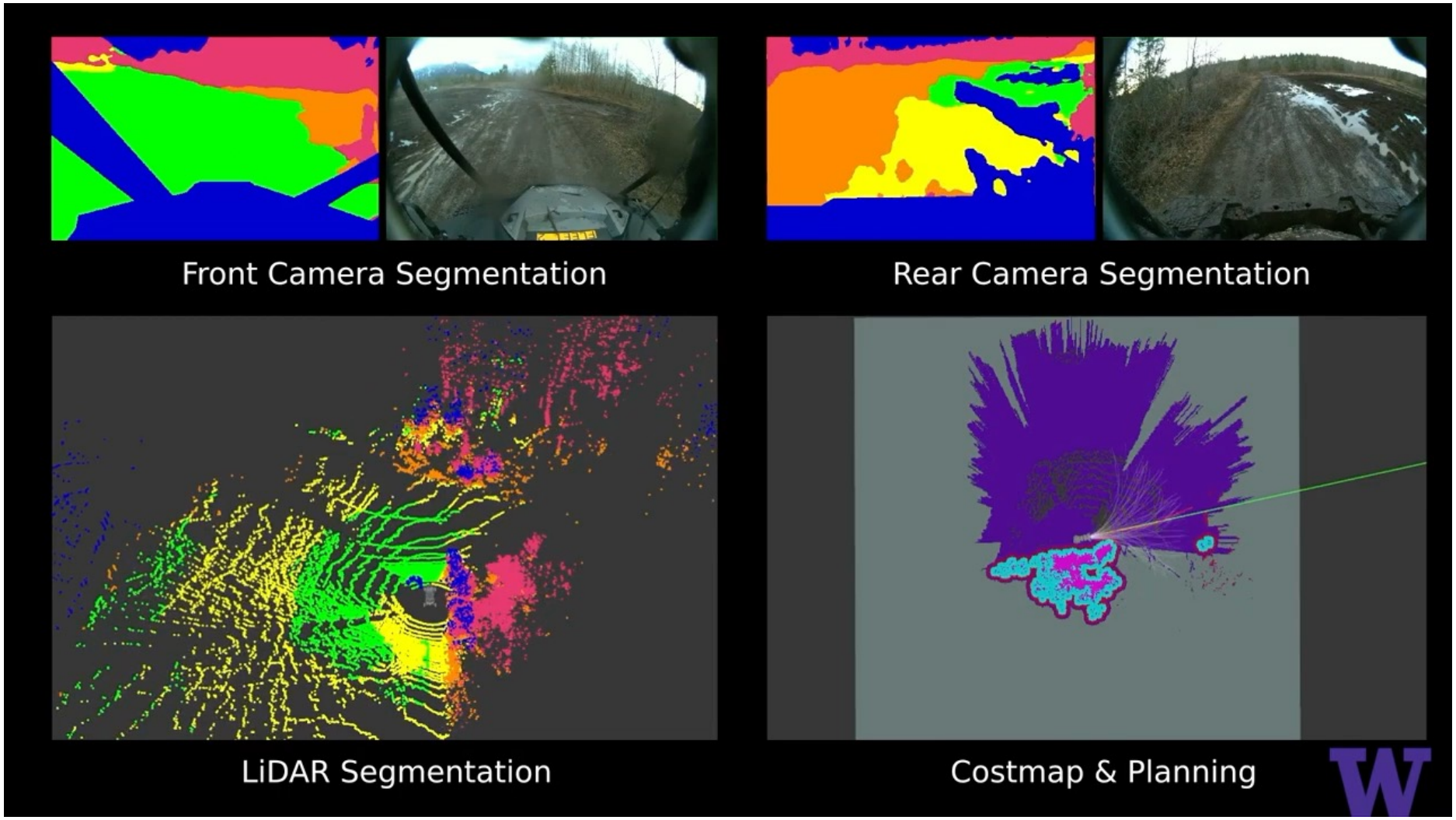
Williams, Grady, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. "Information theoretic mpc for model-based reinforcement learning." In *IEEE international conference on robotics and automation (ICRA)*, 2017.

MPPI运算加速

- 每步需要评估1200个控制序列
- 控制频率为40HZ
- 每秒需要调用4.8百万次动力学模型



MPC 赛车手系统


Front Camera Segmentation
Rear Camera Segmentation
LiDAR Segmentation
Costmap & Planning

第七讲总结

- 车辆运动学
  - 自行车模型（可以前轮、后轮、质心为参考点）
- 车辆动力学
  - 纵向动力学公式：引擎力矩->纵向加速度
  - 横向动力学公式：前轮转向角->横向速度、角速度、角加速度
- 车辆控制：
  - 开环控制、闭环控制流程
  - 纵向：PID
  - 横向：Pure Pursuit、Stanley
  - 高阶：MPC、MPPI

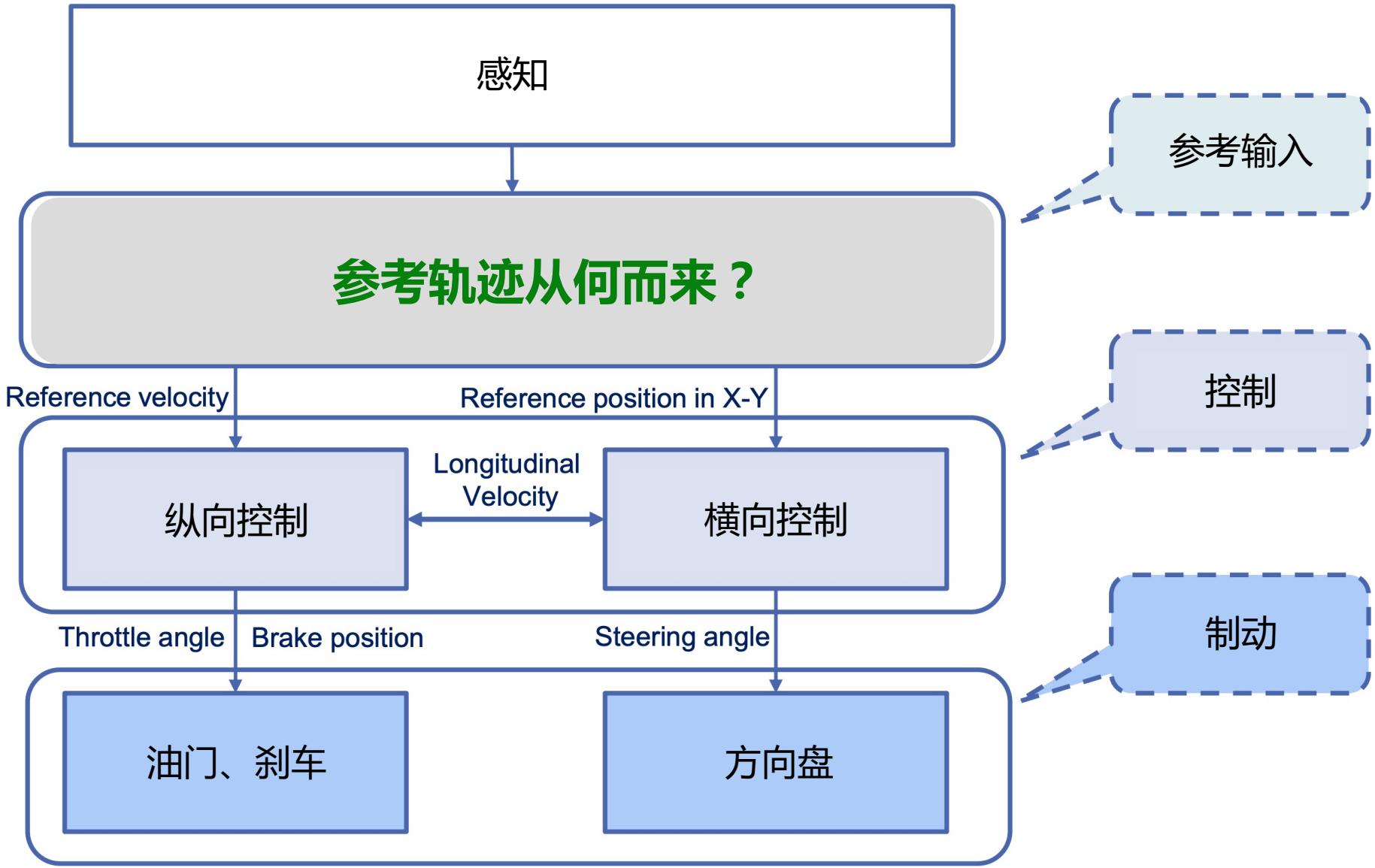
第七讲扩展阅读

- **自行车模型**："Chapter 2, Lateral Vehicle Dynamics", R. Rajamani, *Vehicle Dynamics and Control*, Mechanical Engineering Series, https://www.springer.com/cda/content/document/cda_downloaddocument/9781461414322-c1.pdf?SGWID=0-0-45-1265143-p174267791. (2012)
- **PID纵向控制**：R. Rajamani, "Introduction to Longitudinal Control " In: *Vehicle Dynamics and Control*, Mechanical Engineering Series, https://link.springer.com/chapter/10.1007%2F0-387-28823-6_5 (2006).
- **Pure Pursuit控制**：Snider, J. M., "Automatic Steering Methods for Autonomous Automobile Path Tracking", Robotics Institute, Carnegie Mellon University, Pittsburg (February 2009). https://www.ri.cmu.edu/pub_files/2009/2/Automatic_Steering_Methods_for_Autonomous_Automobile_Path_Tracking.pdf
- **Stanley控制**：Hoffmann, G. et al., "Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing", Stanford University, (2007). http://ai.stanford.edu/~gabeh/papers/hoffmann_stanley_control07.pdf
- **MPC控制**：Falcone, P. et al., "Predictive Active Steering Control for Autonomous Vehicle Systems", IEEE (2007). https://ieeexplore.ieee.org/document/4162483

第七讲扩展阅读

- **MPPI控制**：G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, “Aggressive driving with model predictive path integral control,” in 2016 IEEE International Conference on Robotics and Automation (ICRA), May 2016, pp. 1433–1440.
- **MPPI+NN动力学**：Williams, Grady, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. "Information theoretic mpc for model-based reinforcement learning." In IEEE international conference on robotics and automation (ICRA), 2017.
- **MPPI+Q-Learning**：Bhardwaj, Mohak, Ankur Handa, Dieter Fox, and Byron Boots. "Information theoretic model predictive q-learning." In Learning for Dynamics and Control, pp. 840-850. PMLR, 2020.
- **Differentiable MPC**: Amos, Brandon, Ivan Jimenez, Jacob Sacks, Byron Boots, and J. Zico Kolter. "Differentiable mpc for end-to-end planning and control." Advances in neural information processing systems 31 (2018).

车辆规控总体流程（简化）


感知
参考轨迹从何而来？
参考输入
Reference velocity
Reference position in X-Y
控制
纵向控制
Longitudinal Velocity
横向控制
Throttle angle
Brake position
Steering angle
制动
油门、刹车
方向盘

第八讲：运动规划

蔡盼盼 副教授

上海交通大学清源研究院

研究领域：机器人规划、机器人学习、自动驾驶

邮箱：Cai_panpan@sjtu.edu.cn

网站：https://cindycia.github.io/

机器人的核心能力--运动

一辆自动驾驶汽车应当如何运动？

- What is your mission?
- What is the scenario?
- What are the allowed behaviors?
- How do interact with others?

自动驾驶任务 (Mission)

定义：从地图上的A点导航到B点

- “将乘客从上交闵行校区东上院送到虹桥机场”

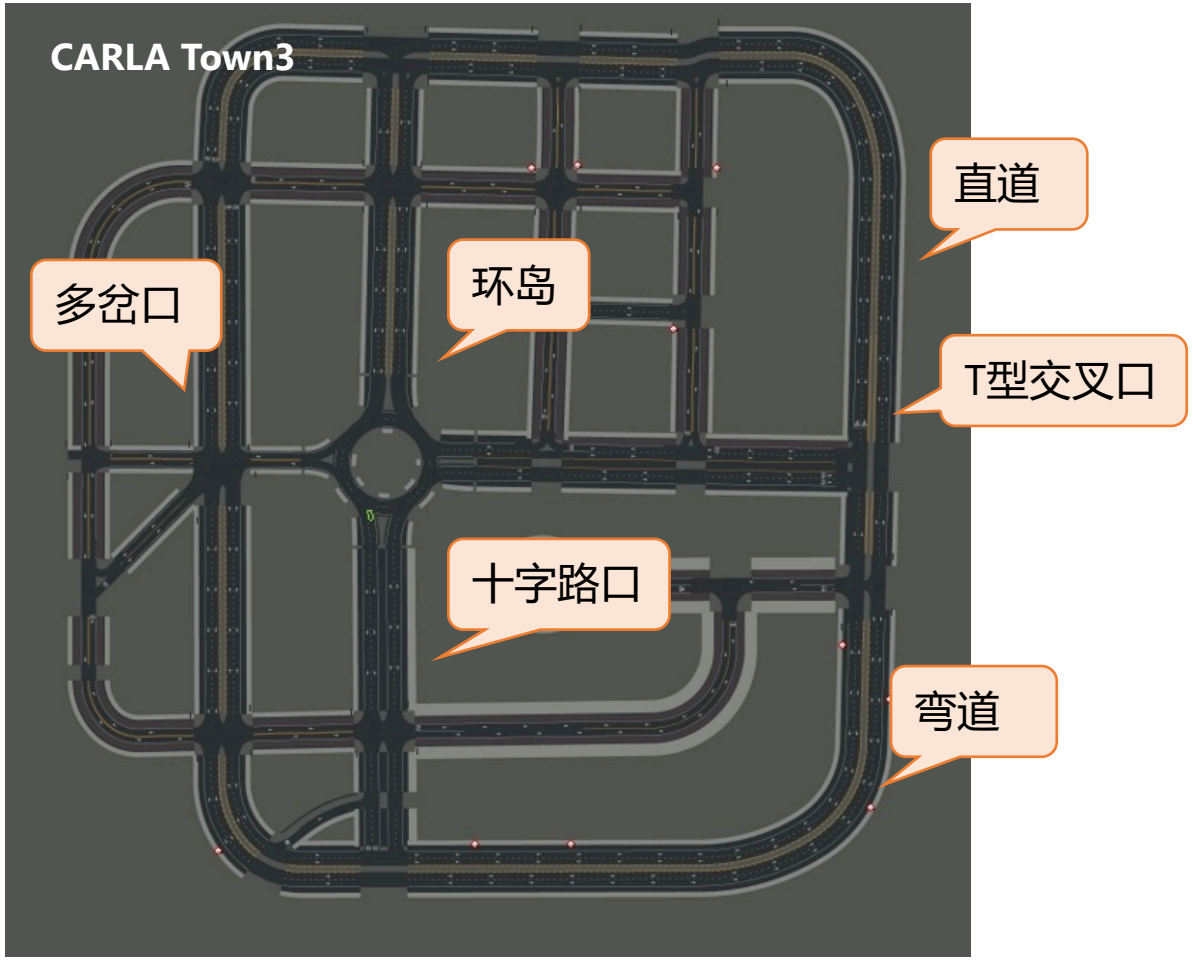
宏观规划

不考虑底层细节

规划目标是最快时间或最短距离到达

自动驾驶场景 (Scenario)

定义：不同地图结构导致的不同交互场景



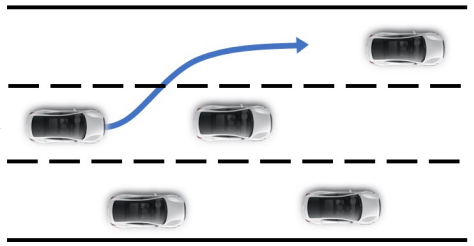
无法命名的场景...

自动驾驶行为 (Behavior)

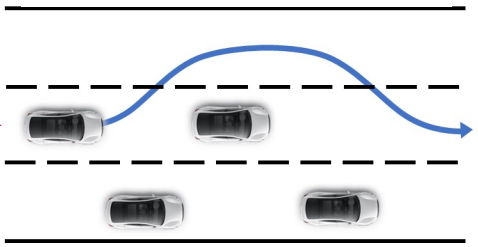
定义：不同场景导致不同的抽象交通行为

例：高速路场景

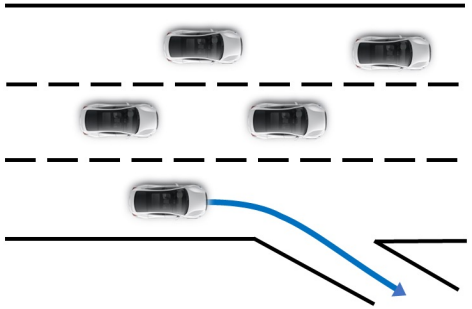
换道



超车



进/出匝道

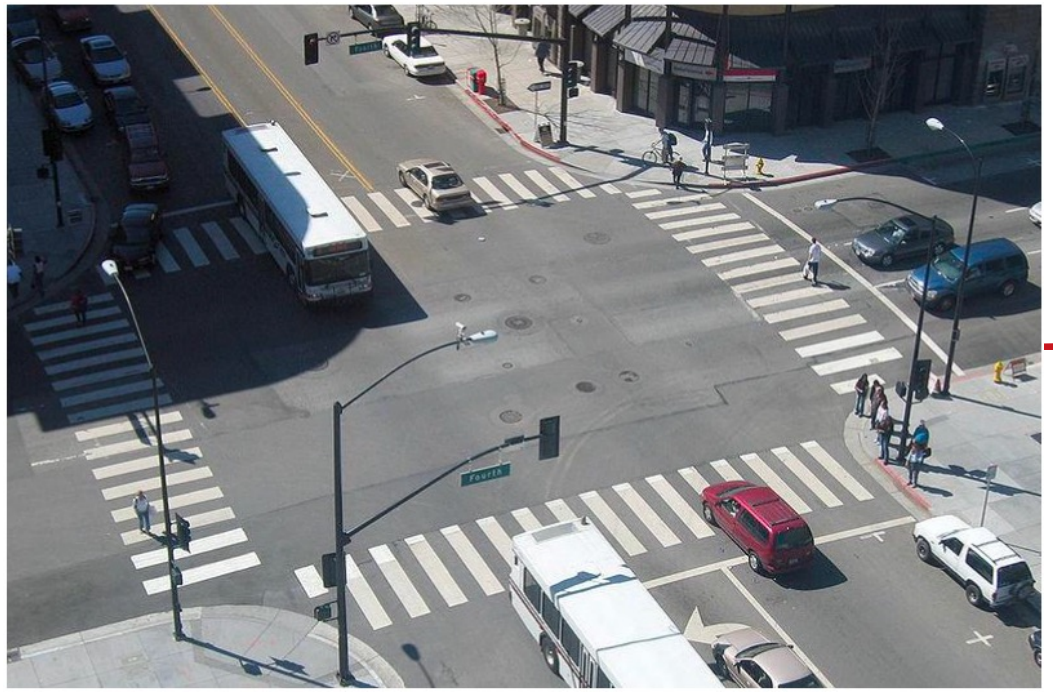


车道保持、汇入、汇出、避让...

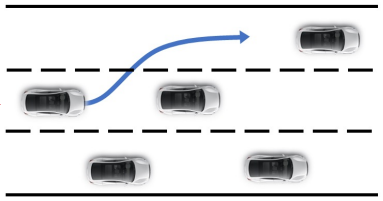
自动驾驶行为 (Behavior)

定义：不同场景导致不同的抽象交通行为

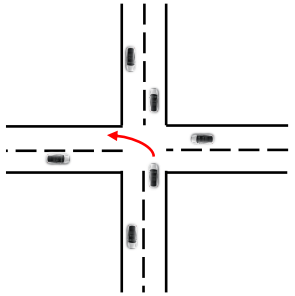
例：十字路口场景



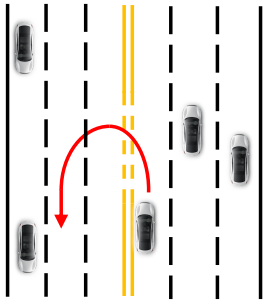
变道



左转



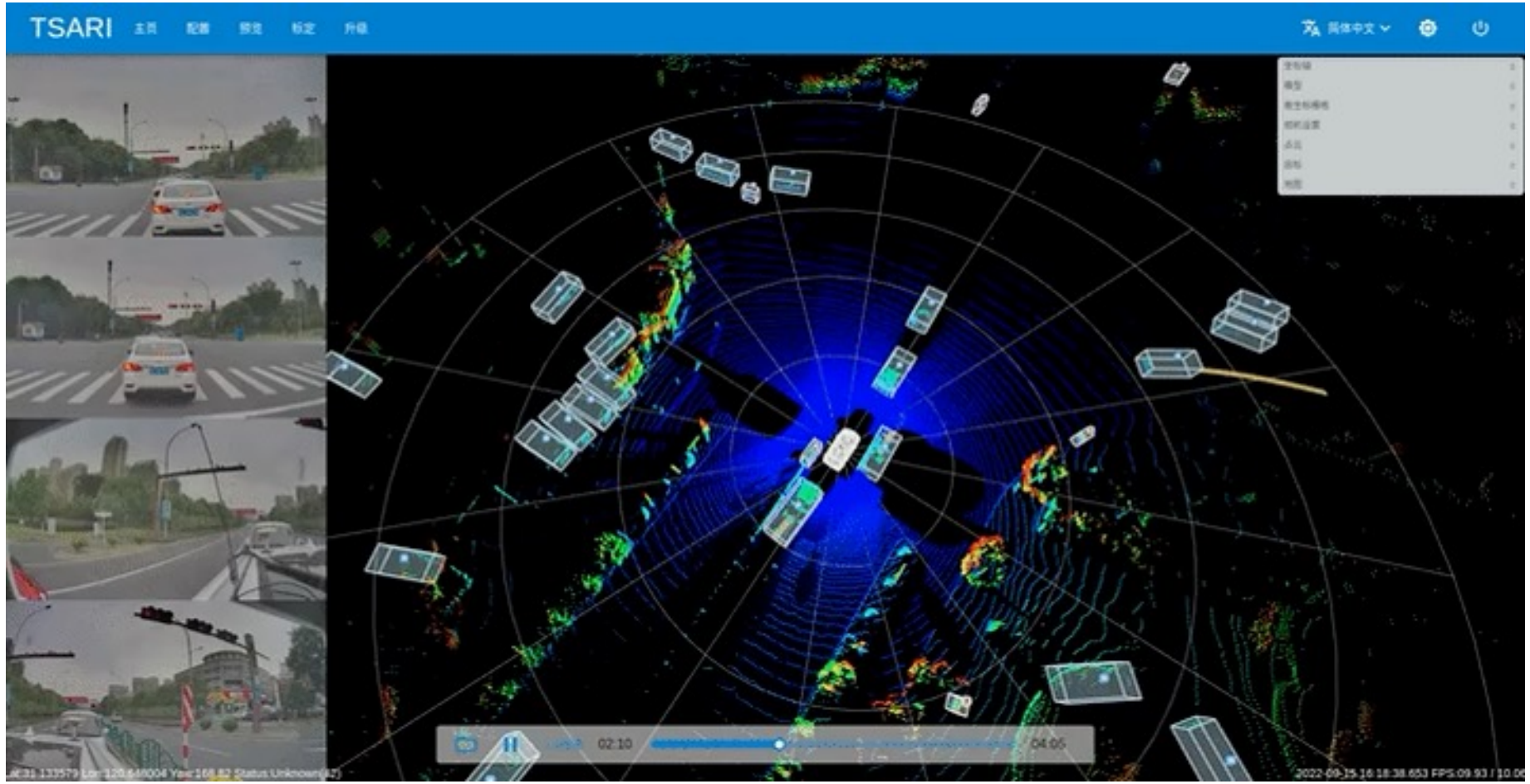
掉头



注：结构化场景中，自车路径可以以车道线作为参考，速度需要根据**实时交互**调整

动态障碍物

定义：车辆、弱势交通参与者、动物等

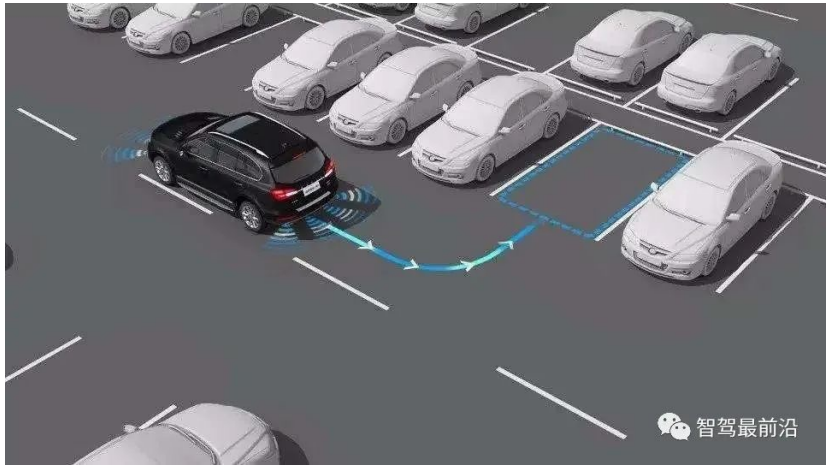


交通管制装置

定义：交通信号灯、交通标志、人行横道和车道标记等

静态障碍物

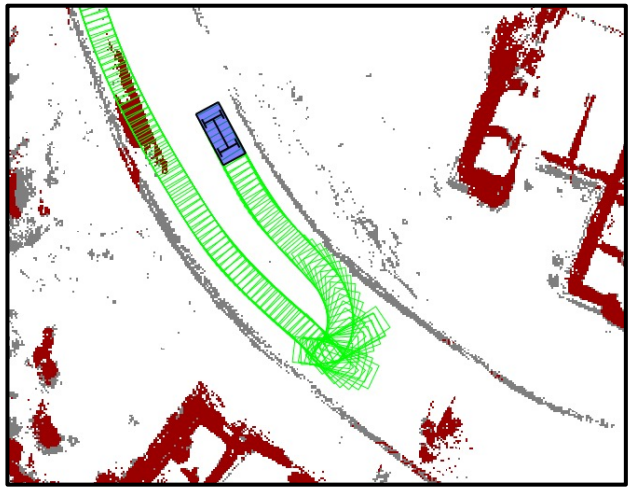
定义：道路边界、停放的车辆、警示牌、 路锥、施工区域等



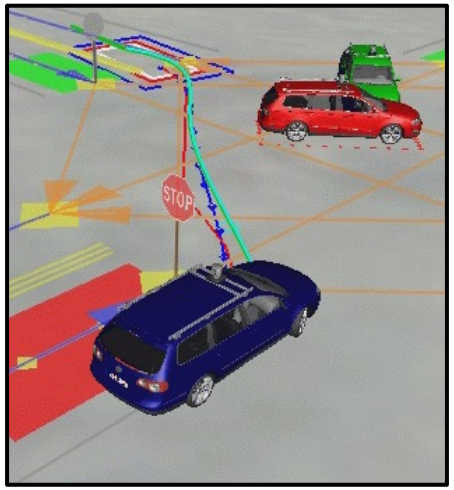
非结构化场景

定义1：交通参与者无法直接按照道路结构运行的场景

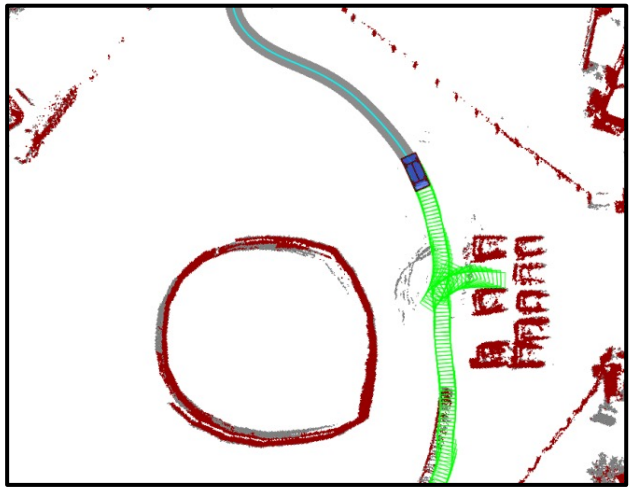
注：非结构化场景中，行车路径需要灵活适应动、静态障碍物的变化，无固定参考路径可用



单行道掉头



绕开非法障碍物



停车场停车

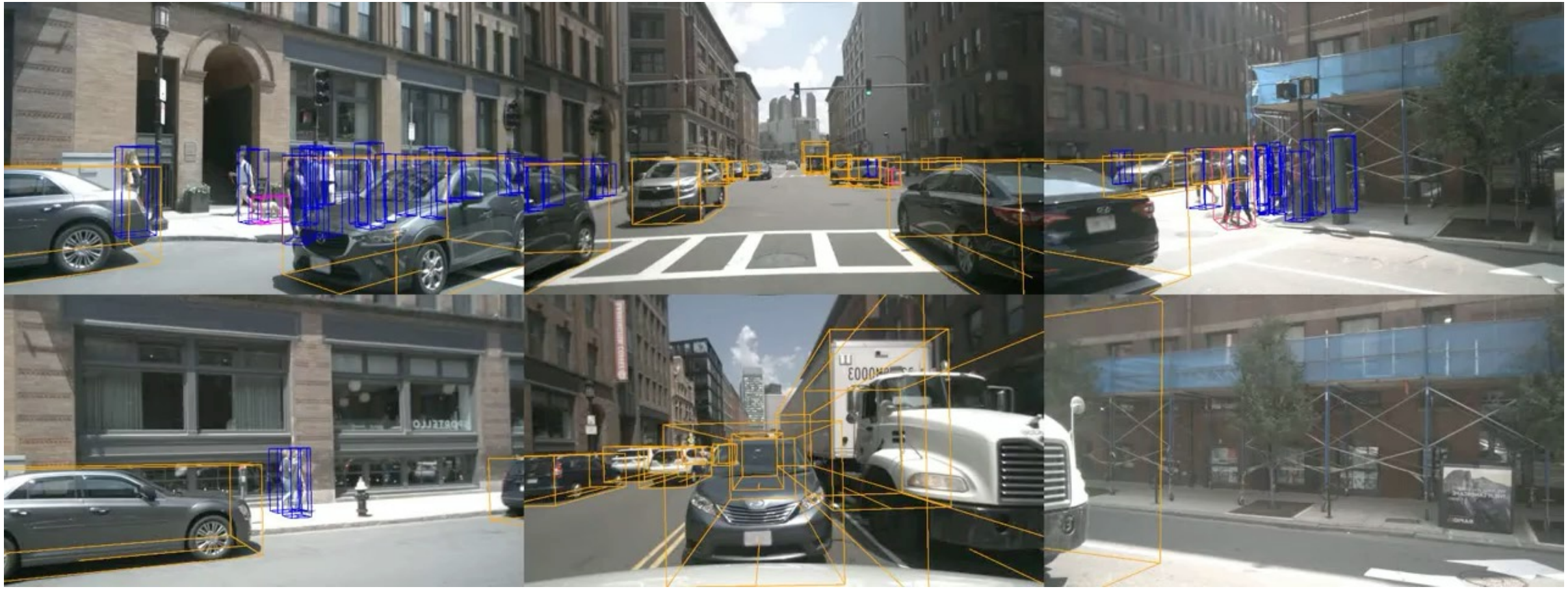
非结构化场景

定义2：交通参与者不按照道路结构与交通规则行驶的场景

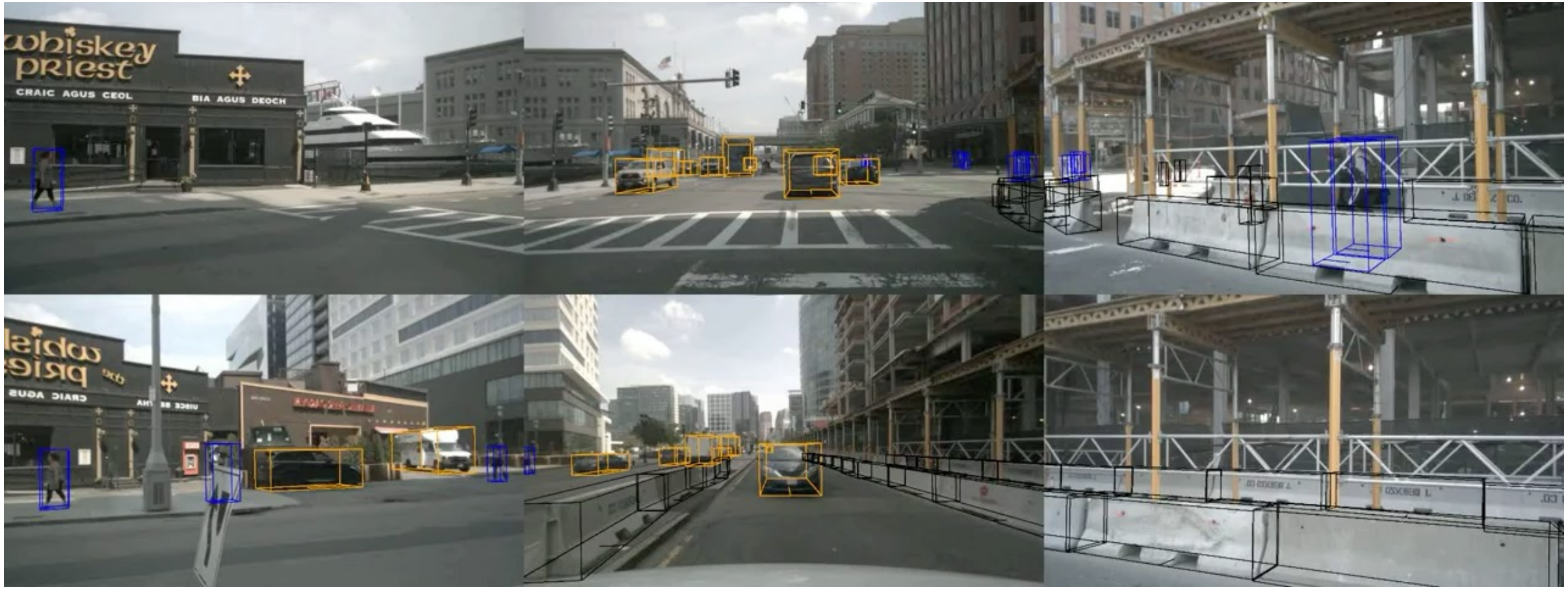
注：非结构化场景中，行车路径需要灵活适应与动、静态障碍物的变化，无固定参考路径可用

End-to-End？大模型？

End-to-End 测试场景一（NuScene）



End-to-End 测试场景二（NuScene）



End-to-End 测试场景三（NuScene）

决策与运动规划难度？

数据集

现实（非洲）

决策与运动规划难度？

数据集

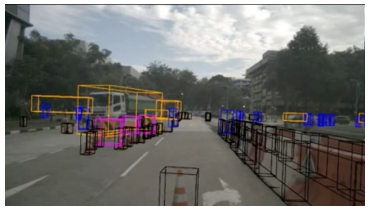
现实（国内）

注：常见数据集仅包含常见的（交互通常简单的）驾驶场景，而真正的L5自动驾驶系统需要能够解决上述所有情形

端到端SOTA在数据集中的效果？

ID	Method	Ego Status in BEV	Ego Status in Planer	L2 (m) ↓ 1s	L2 (m) ↓ 2s	L2 (m) ↓ 3s	L2 (m) ↓ Avg.	Collision (%) ↓ 1s	Collision (%) ↓ 2s	Collision (%) ↓ 3s	Collision (%) ↓ Avg.	Intersection (%) ↓ 1s	Intersection (%) ↓ 2s	Intersection (%) ↓ 3s	Intersection (%) ↓ Avg.	ckpt. source
0	ST-P3	✗	✗	1.59†	2.64†	3.73†	2.65†	0.69†	3.62†	8.39†	4.23†	2.53†	8.17†	14.4†	8.37†	Official
1	UniAD	✗	✗	0.59	1.01	1.48	1.03	0.16	0.51	1.64	0.77	0.35	1.46	3.99	1.93	Reproduce
2	UniAD	✓	✗	0.35	0.63	0.99	0.66	0.16	0.43	1.27	0.62	0.21	**1.32**	3.63	1.72	Official
3	UniAD	✓	✓	0.20	0.42	0.75	0.46	0.02	**0.25**	0.84	0.37	**0.20**	1.33	**3.24**	**1.59**	Reproduce
4	VAD-Base	✗	✗	0.69	1.22	1.83	1.25	0.06	0.68	2.52	1.09	1.02	3.44	7.00	3.82	Reproduce
5	VAD-Base	✓	✗	0.41	0.70	1.06	0.72	0.04	0.43	1.15	0.54	0.60	2.38	5.18	2.72	Official
6	VAD-Base	✓	✓	0.17	0.34	0.60	0.37	0.04	0.27	**0.67**	**0.33**	0.21	2.13	5.06	2.47	Official
7	GoStright	-	✓	0.38	0.79	1.33	0.83	0.15	0.60	2.50	1.08	2.07	8.09	15.7	8.62	-
8	Ego-MLP	-	✓	**0.15**	**0.32**	0.59	**0.35**	**0.00**	0.27	0.85	0.37	0.27	2.52	6.60	2.93	
9	BEV-Planner*	✗	✗	0.27	0.54	0.90	0.57	0.04	0.35	1.80	0.73	0.63	3.38	7.93	3.98	-
10	BEV-Planner	✗	✗	0.30	0.52	0.83	0.55	0.10	0.37	1.30	0.59	0.78	3.79	8.22	4.26	-
11	BEV-Planner+	✓	✗	0.28	0.42	0.68	0.46	0.04	0.37	1.07	0.49	0.70	3.77	8.15	4.21	-
12	BEV-Planner++	✓	✓	0.16	0.32	**0.57**	**0.35**	**0.00**	0.29	0.73	0.34	0.35	2.62	6.51	3.16	-

每75秒碰撞1次



自动驾驶规划系统的评价指标

子任务数据集验证：
- 目标检测准确度
- 轨迹预测准确度
- 地图重建准确度
- …

数据集开环验证：
- 类人度
- 事故率
- 任务效率
- 任务失败率
- 轨迹舒适度
- …

仿真闭环验证：
- 事故率
- 任务效率
- 任务失败率
- 轨迹舒适度
- …

真车实验验证：
- 接管率
- 任务效率
- 任务失败率
- 轨迹舒适度
- 能耗
- …

开放真车验证：
- 事故率
- 任务效率
- 任务失败率
- 客户舒适度
- 性价比
- …

问题：

$\sum$ **子任务性能**

$\neq$ **任务性能**

问题：

环境与自车的交互被忽略

问题：

仿真模型不真实

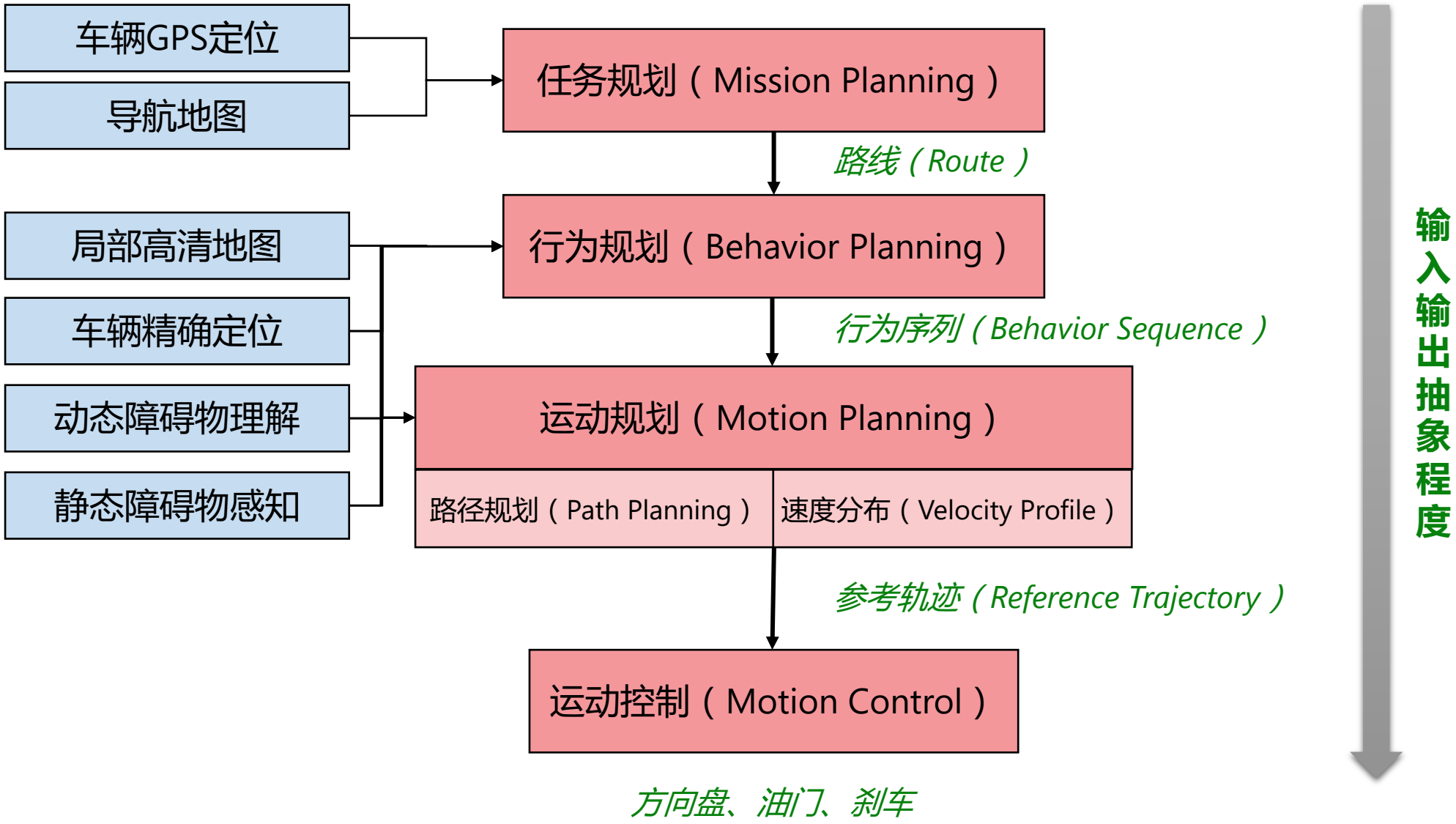
问题：

安全司机接管具有主观性
测试量小

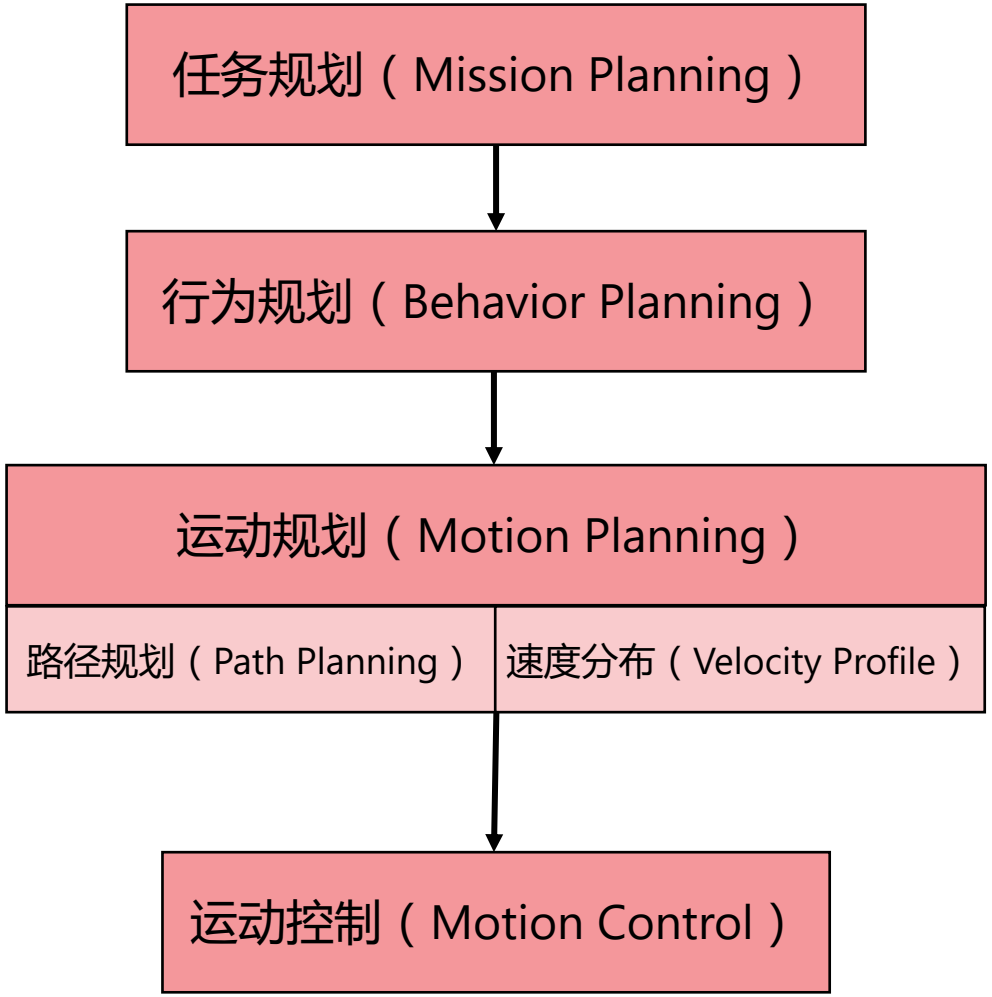
问题：

对系统安全性、稳定性要求极高

层级式自动驾驶规划架构



层级式自动驾驶规划架构



运行频率增高

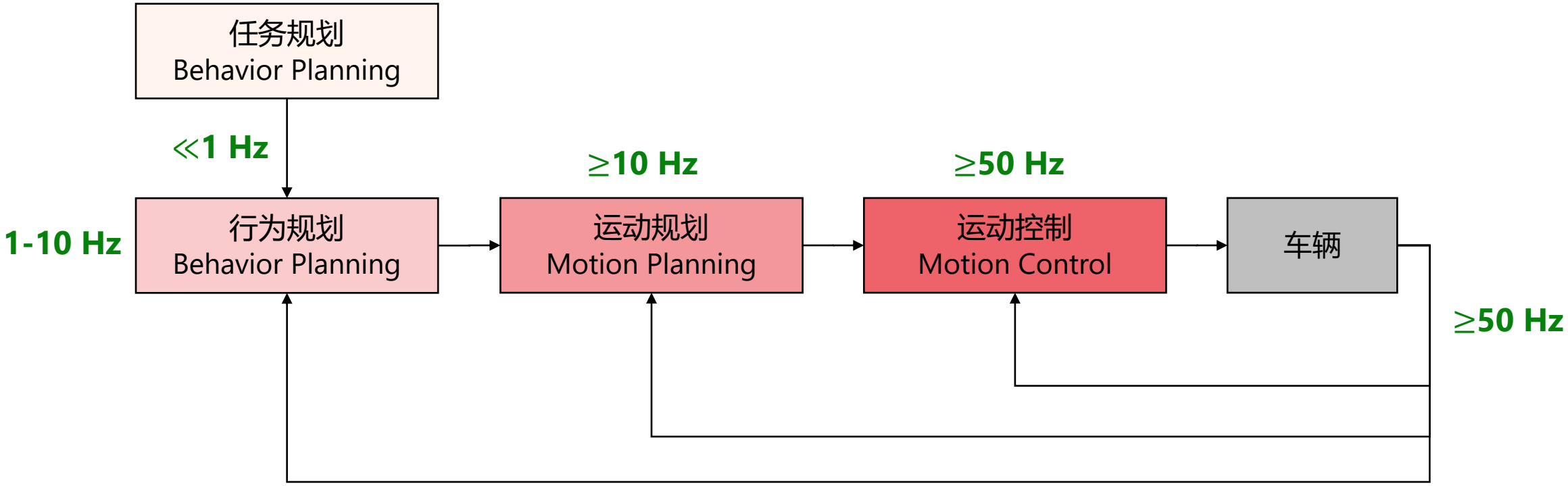
(≪ 1 Hz) 情景式规划（Episodic Planning）
在任务开始时、任务进展受阻时、环境显著改变时进行

(1-10 Hz) 在线决策规划（Online Decision Making）
每时间步执行一个“规划”-“执行”-“感知”闭环

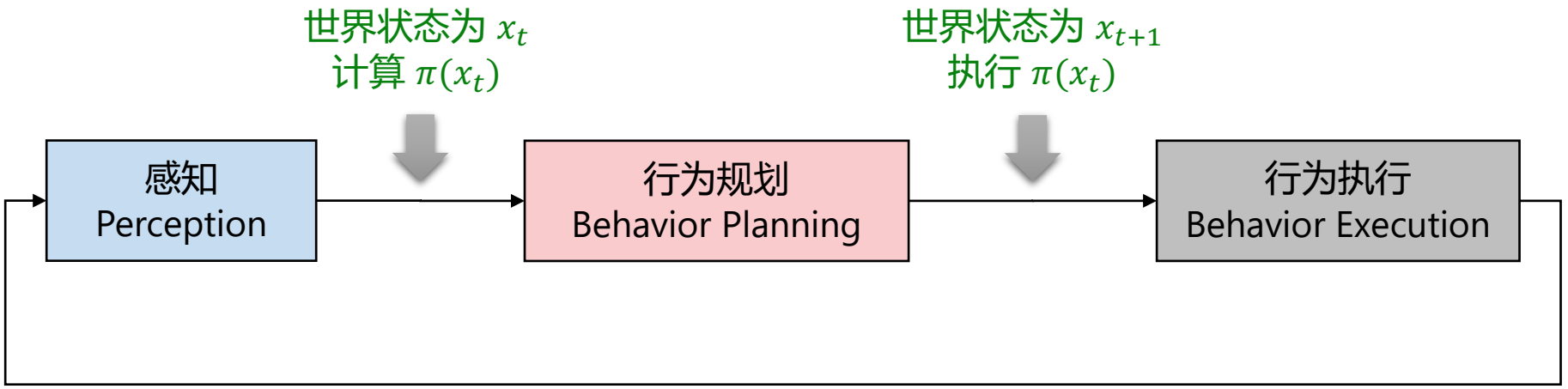
(≥10 Hz) 实时规划（Real-Time Planning）
每时间步执行一个“规划”-“执行”-“感知”闭环

(≥ 30 Hz) 实时控制（Real-Time Control）
每时间步执行一个“优化”-“执行”-“反馈”闭环

“规划”-“执行”-“感知”闭环

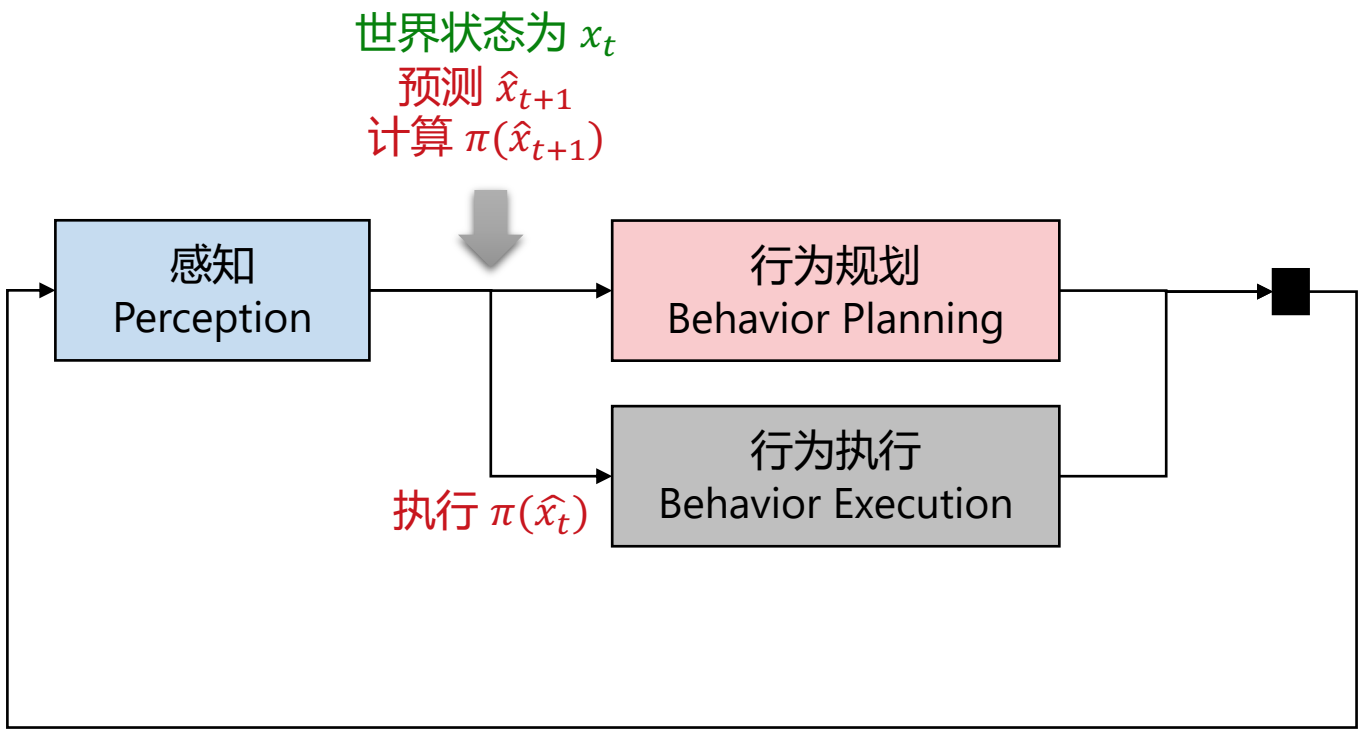

任务规划
Behavior Planning
≪1 Hz
1-10 Hz
行为规划
Behavior Planning
≥10 Hz
运动规划
Motion Planning
≥50 Hz
运动控制
Motion Control
车辆
≥50 Hz

规划-执行轮替（Alternating Planning and Execution）



- 规划-执行-感知闭环的直接实现
- **问题**：规划需要时间（0.1s-1s），而此时环境会发生改变
  - 若世界改变缓慢：
  - 若世界改变迅速：事故风险增大（规划时间=反应时间）
    - *人的反应时间最快可以小于1s*

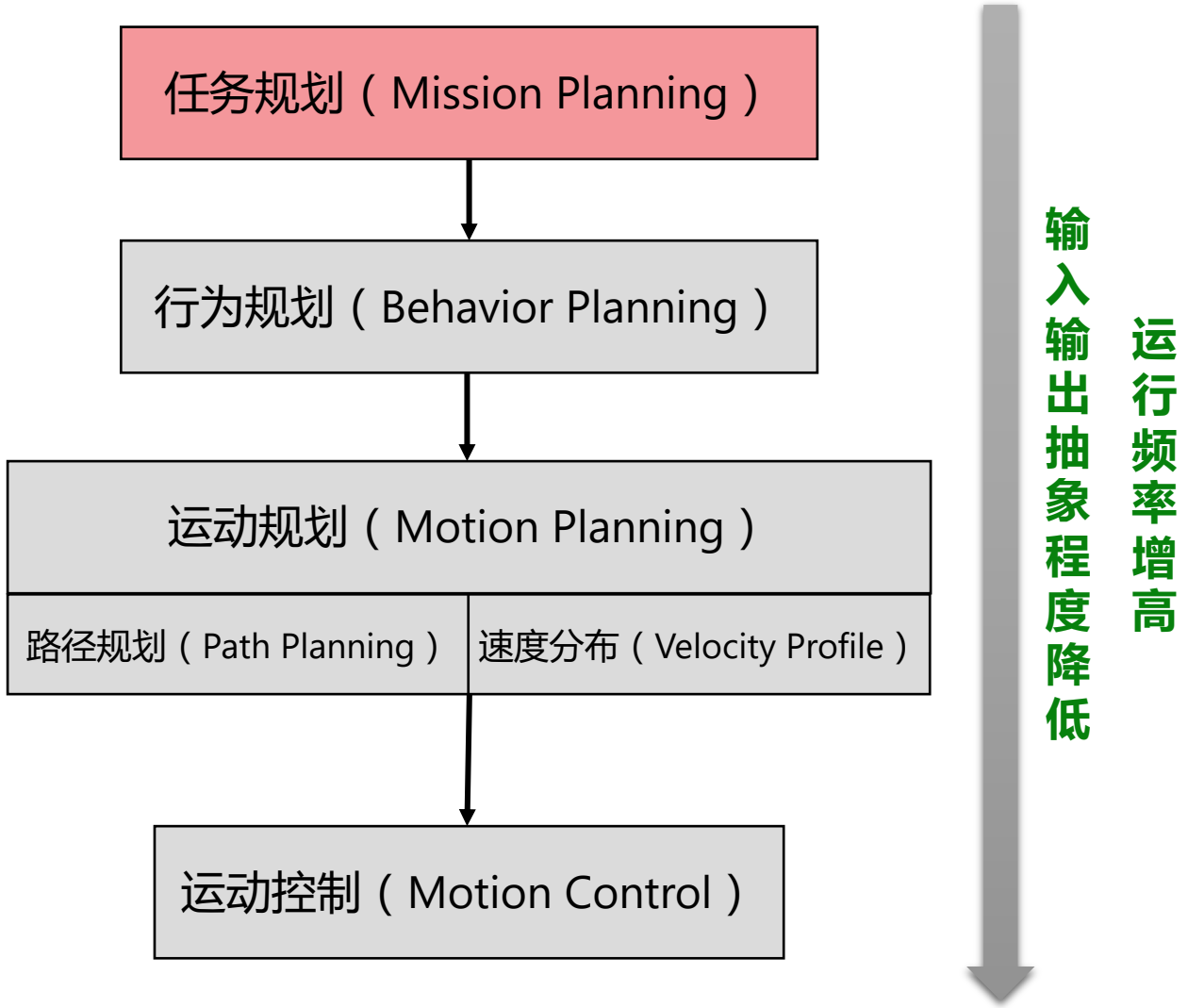
规划-执行交错（Interleaving Planning and Execution）



- **好处**：缩短反应时间
- **问题**：
  - 模型不一定准确
    - 在1s之内，匀速预测模型可以给出非常准确的预测
  - 突发事件
    - 规划时间越短越好

Schöller, Christoph, Vincent Aravantinos, Florian Lay, and Alois Knoll. "What the constant velocity model can teach us about pedestrian motion prediction." *IEEE Robotics and Automation Letters* 5, no. 2 (2020): 1696-1703.

层级式自动驾驶规划架构

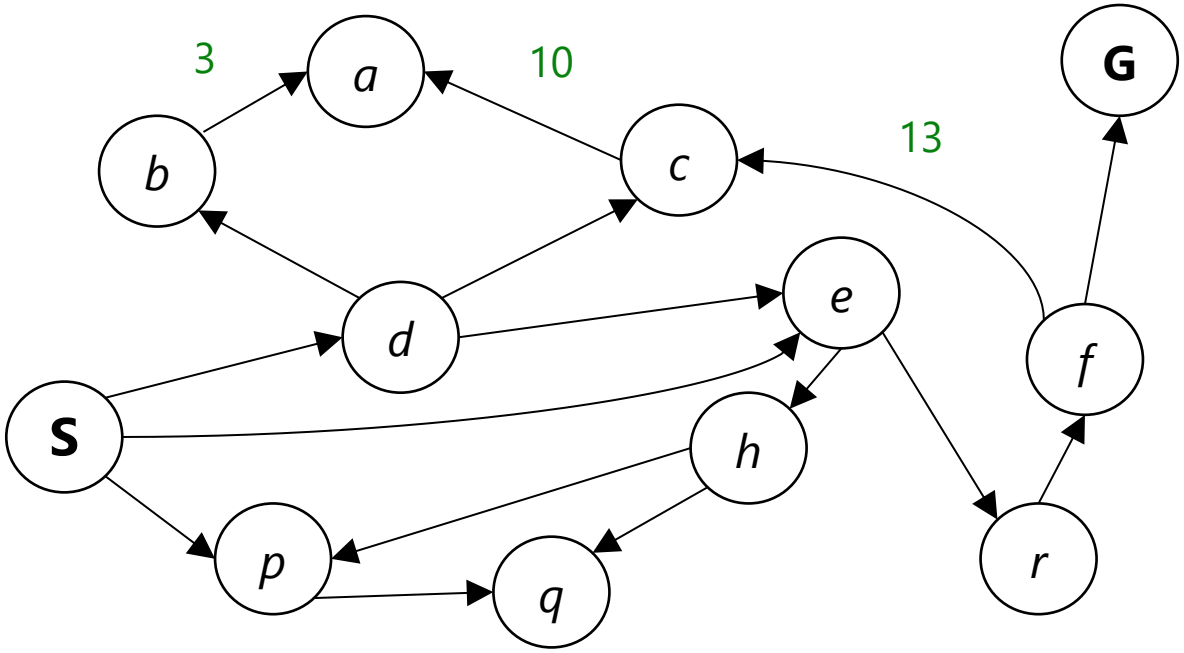


任务规划（Mission Planning）

- 输入：
  - 路网图（road network graph ）
  - 起始点（start position）
  - 终点（goal position）
- 输出：
  - 路径：一系列的途径节点
- 优化目标：
  - 最短距离路径
  - 最短时长路径

路网图（road network graph ）

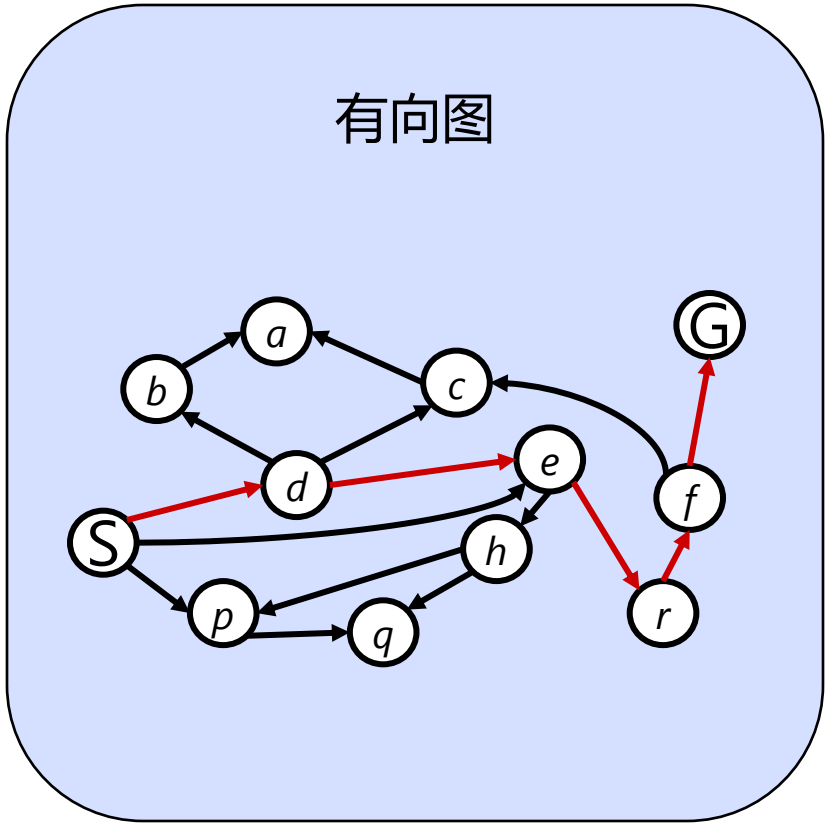
- **边（Edge）**
  - 道路
- **顶点（Vertex）**
  - 道路的起点、终点
- **成本（Cost）**
  - 道路的长度
  - 行驶通过道路的时间



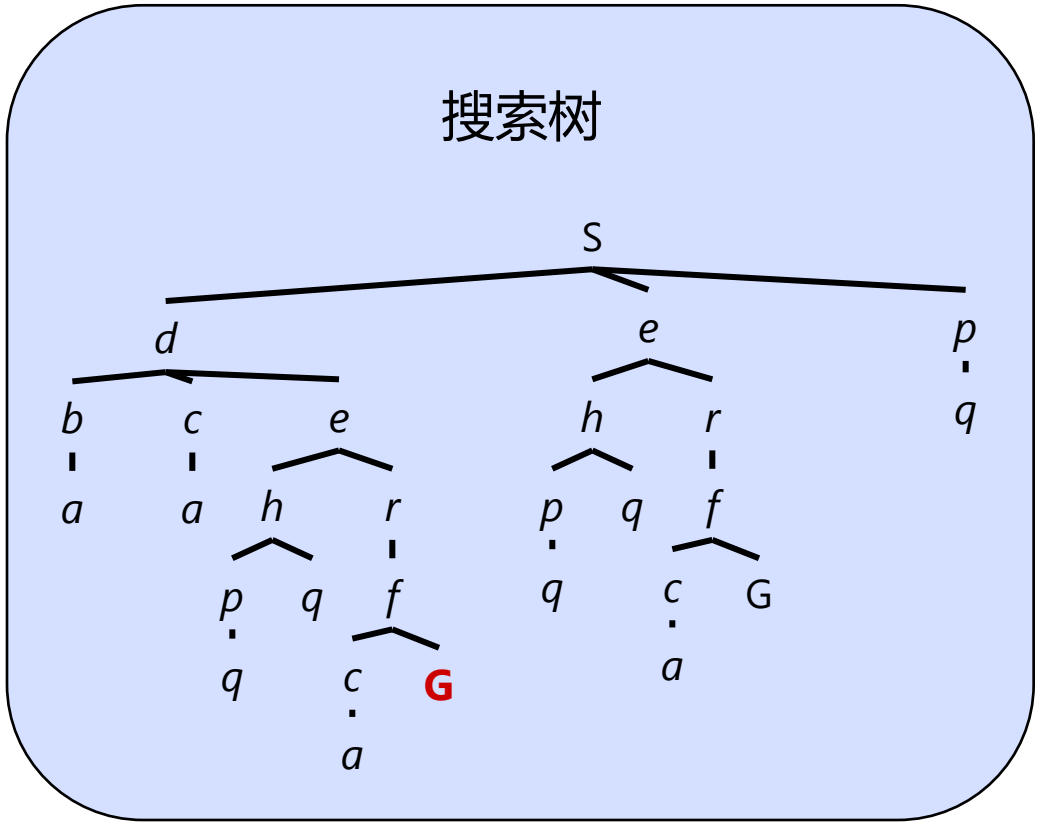
- **性质：**
  - 路网图是有向图
  - 起始点、终点不一定在顶点，但可以被添加为顶点

图片来自 UC Berkeley CS188

图中的路径规划

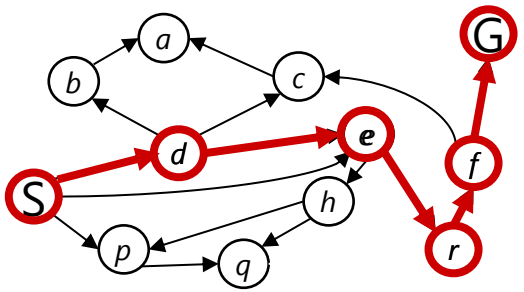


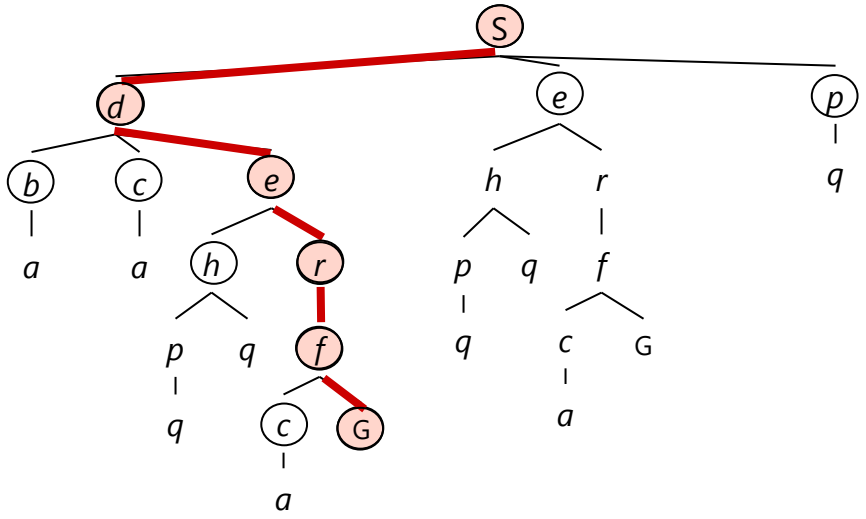
- 搜索树的根节点是起点S
- 搜索树中的每个节点对应图中的一条路径
- **树搜索算法**：逐渐扩展搜索树，直至找到需要的路径
- **基本扩展规律**：每一步扩展一个现有节点（后继顶点创建为子节点）
- **不同的算法**：不同的树扩展策略（优先扩展什么节点）



图片来自 UC Berkeley CS188

树搜索算法展示



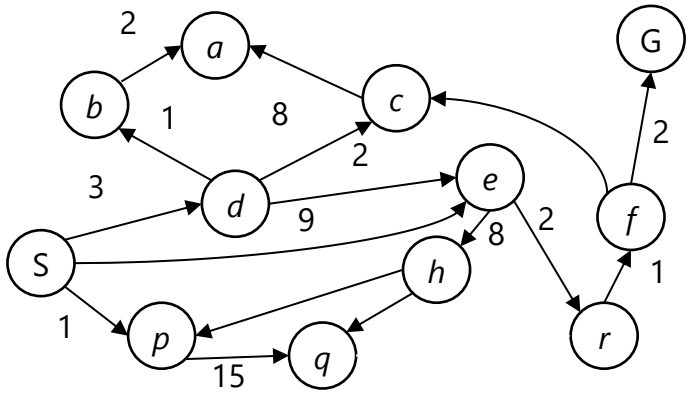


~~s~~
~~s → d~~
s → e
s → p
s → d → b
s → d → c
~~s → d → e~~
s → d → e → h
~~s → d → e → r~~
~~s → d → e → r → f~~
s → d → e → r → f → c
~~s → d → e → r → f → G~~

图片来自 UC Berkeley CS188

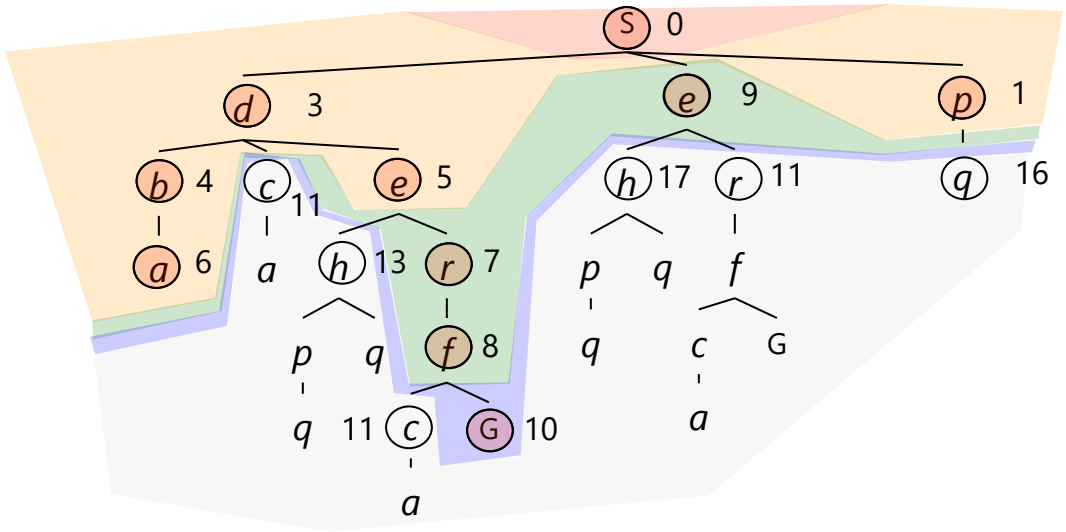
Dijkstra 树搜索算法

- **树扩展策略：**
  - 优先扩展“成本”最低的节点
  - **“成本”**：从根节点到该节点路径的成本（Cost-to-come）



- **实现：**
  - 优先队列（Priority queue）
    - 每一行数据包含：<路径：cost-to-come>
    - 从低到高排序
    - 永远选择最顶部的路径扩展

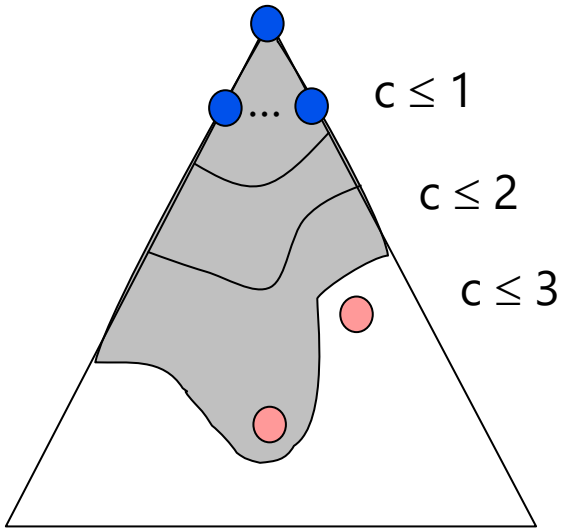
成本等高线

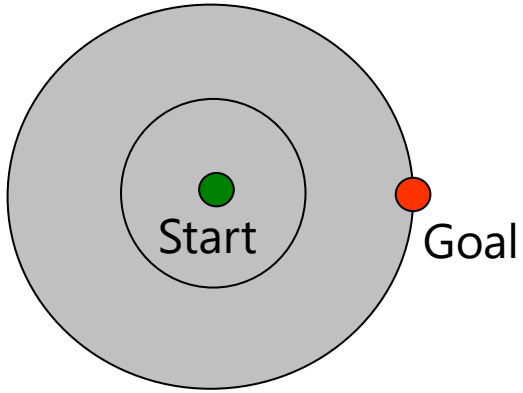


图片来自 UC Berkeley CS188

Dijkstra 树搜索算法

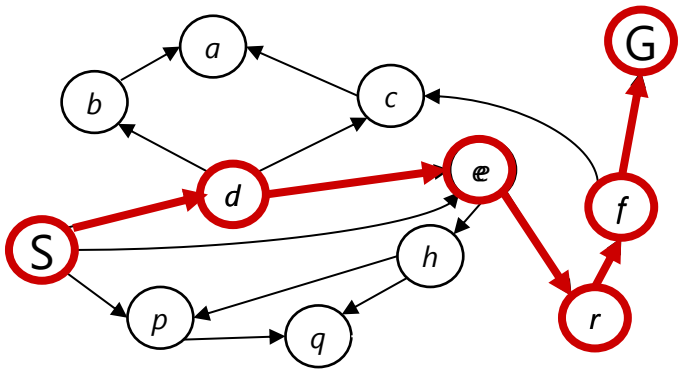
- **理论性质：**
  - 保证能找到最优路径
- **缺点：**
  - 沿着成本等高线均匀向外扩展，不受终点的指引
  - 计算效率低下

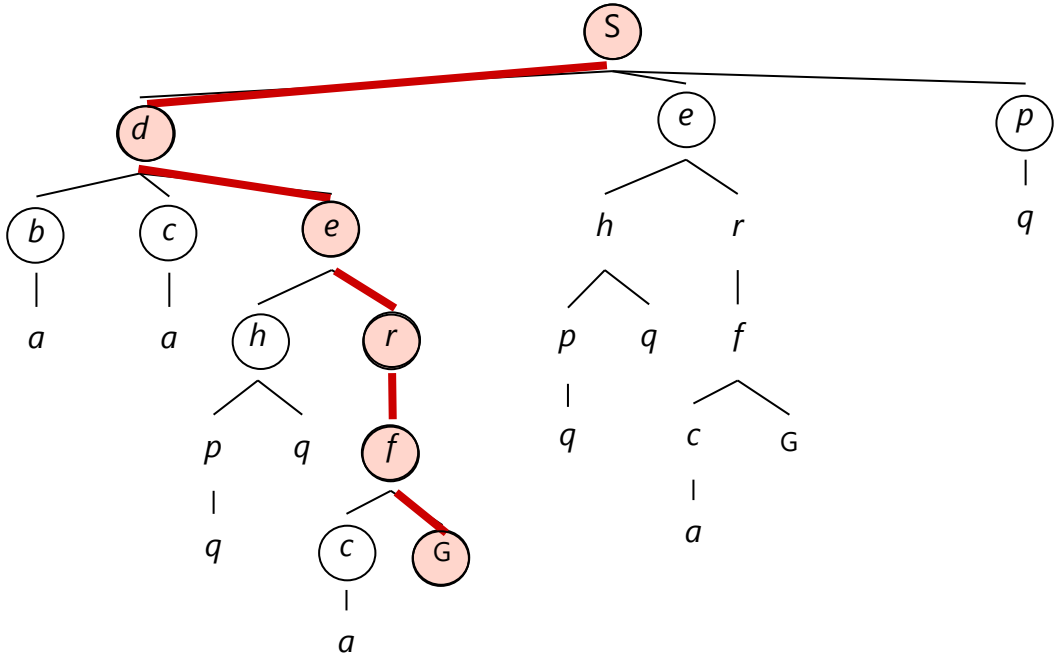




图片来自 UC Berkeley CS188

在优先值中考虑终点信息—搜索启发





优先序列

$< n:\ f(n) >$

优先值：$f(n) = g(n) + h(n)$

$g(n)$: *Cost-to-come*

- n所代表路径的成本

$h(n)$: *Heuristic cost-to-go*

- n到终点的估计成本

$h(n) \equiv 0$ ➡ **Djikstra**

$h(n) \neq 0$ ➡ **A***

图片来自 UC Berkeley CS188

A* 算法性质

- A* 在搜索启发 h 满足以下条件时，可以保证最优性：
  - **Admissibility**: heuristic cost-to-goal ≤ actual cost-to-goal

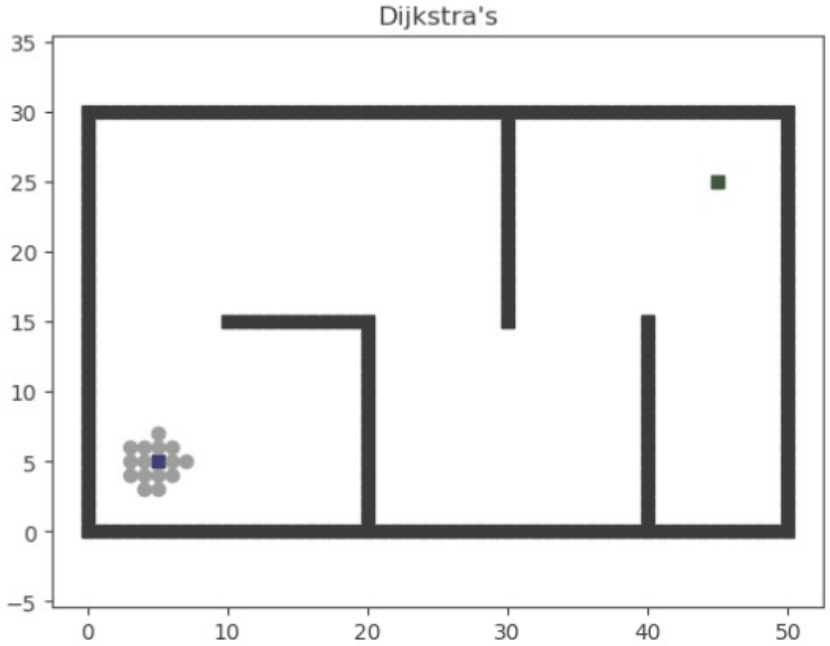
    h(A) ≤ actual cost from A to G
  - **Consistency**: heuristic “edge” cost ≤ actual edge cost

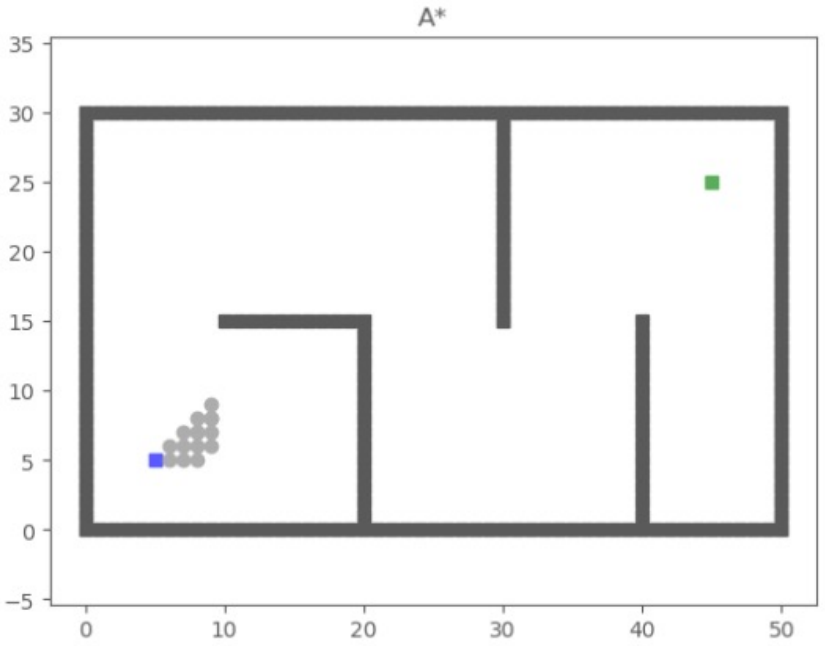
    h(A) – h(C) ≤ cost(A to C)

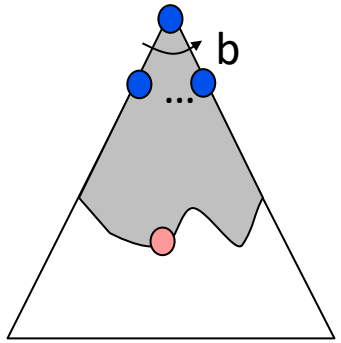
- **实用性质一**：Relaxed problem 的解一定是 admissible heuristic，大概率是consistent heuristic
- **实用性质二**：两个 admissible heuristics 的最大值仍然是 admissible heuristic

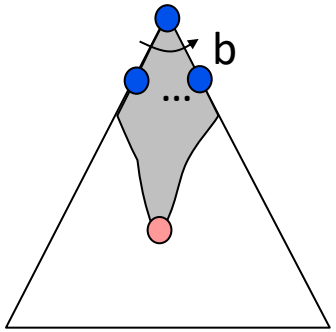
A* 与Dijkstra对比

图片来自 zhm-real









任务规划（Mission Planning）-> A*

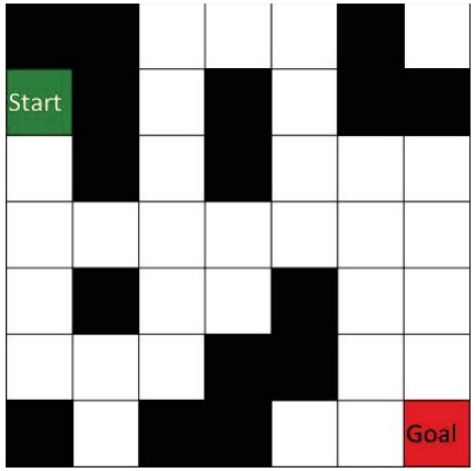
- 输入：
  - 路网图（road network graph ）
  - 起始点（start position）
  - 终点（goal position）
- 输出：
  - 路径：一系列的途径节点
- 优化目标：
  - 最短距离路径
  - 最短时长路径

任务规划（Mission Planning）

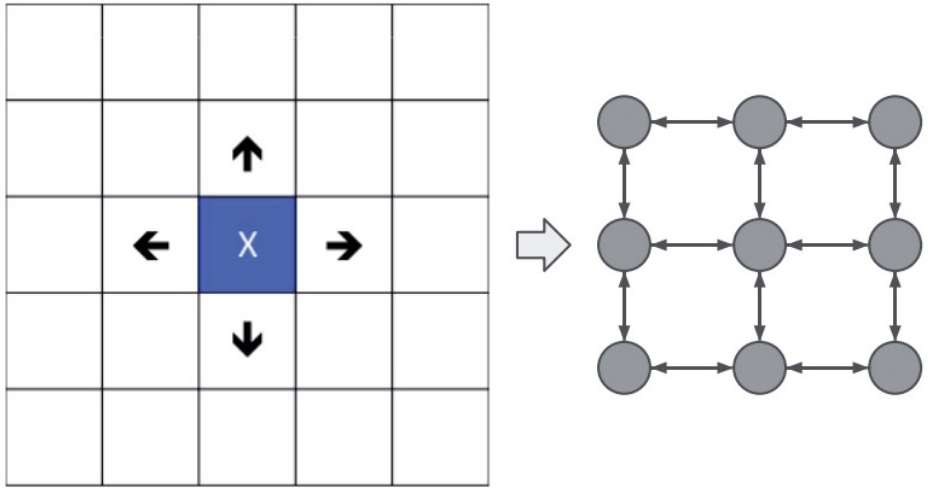
- 输入：
  - 路网图（road network graph ）
  - 起始点（start position）
  - 终点（goal position）
- 输出：
  - 路径：一系列的途径节点
- 优化目标：
  - 最短距离路径
  - 最短时长路径

A* ?

A* 路径规划

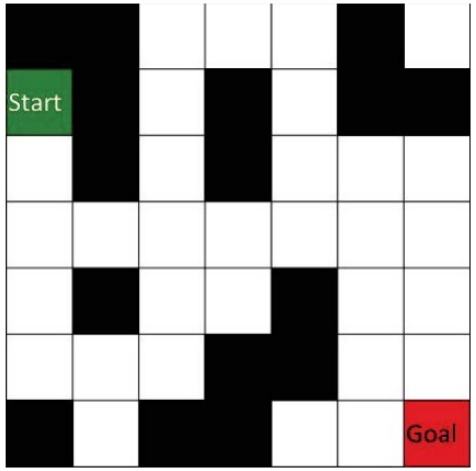


4-Connected Graph

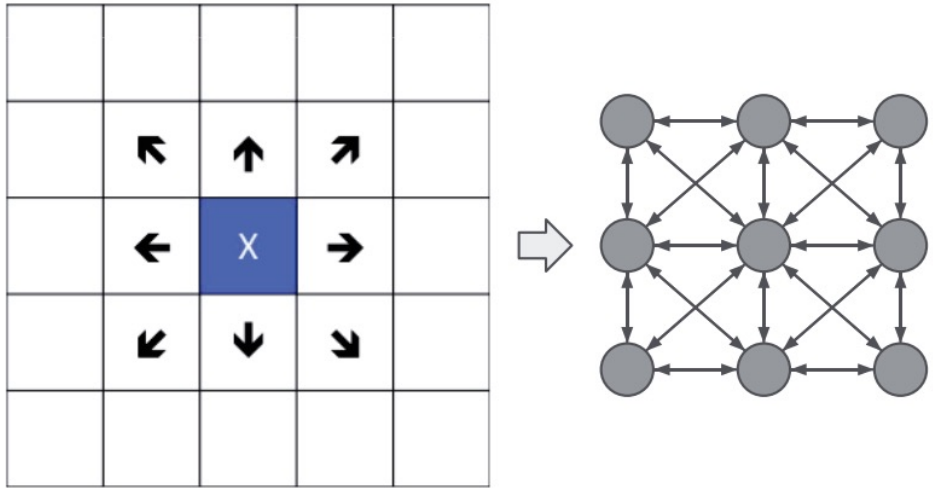


Manhattan Distance
$= \Delta x + \Delta y$

A* 路径规划

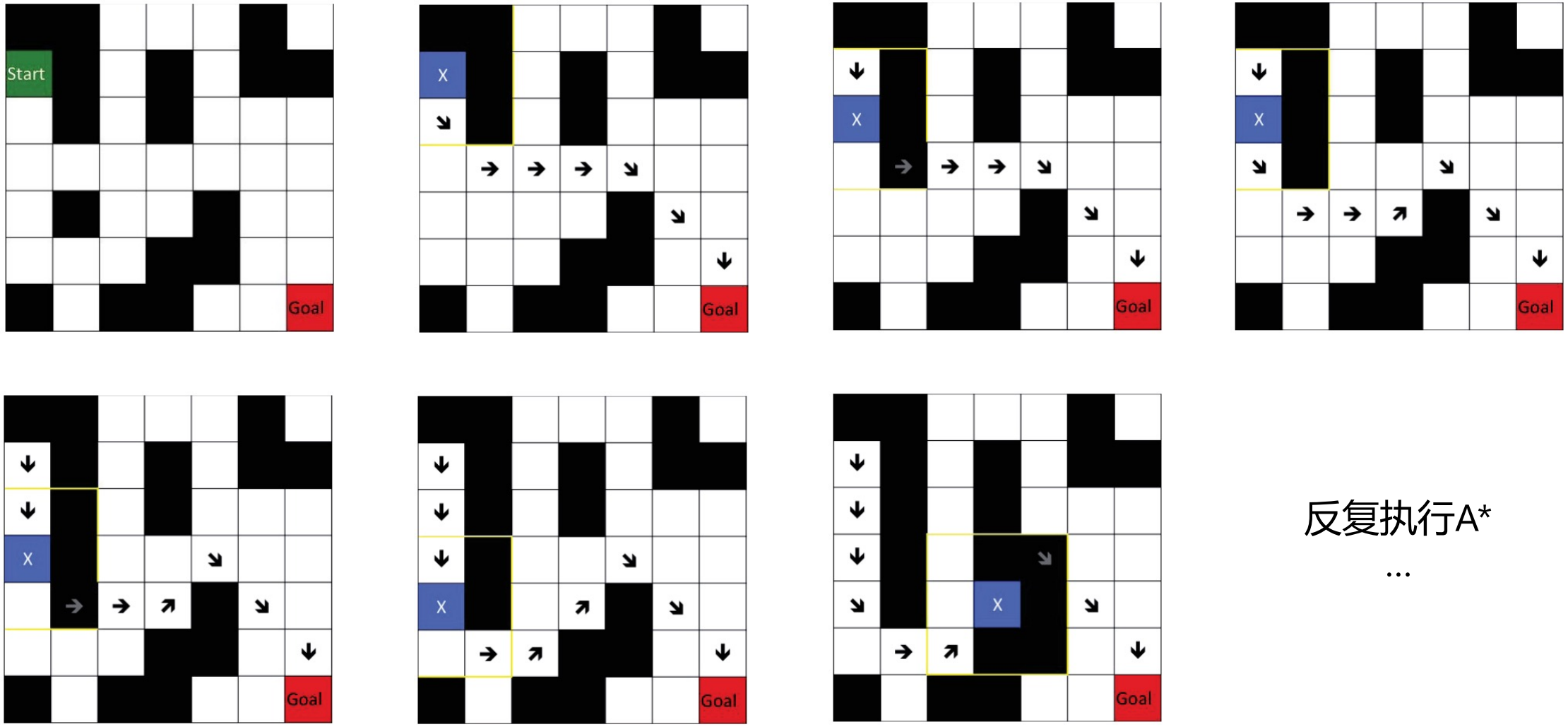


8-Connected Graph



max(Δx, Δy)
(with unit edge weights)

动态图中的路径规划

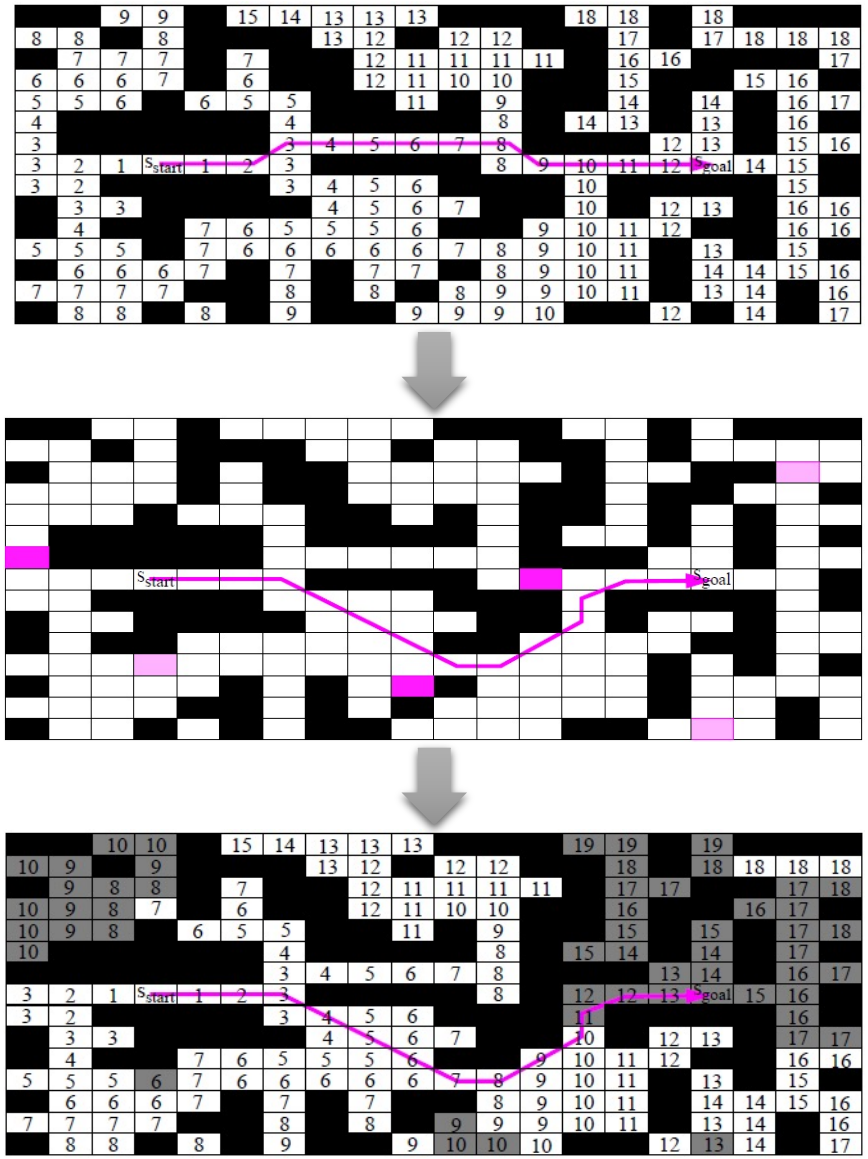


反复执行A*

…

渐进式搜索（Incremental Search）

- 进行图搜索，得到初始路径
  - 保存搜索的结果
    - 最优路径
    - 所有已搜索节点的 Cost-to-come
    - 优先队列
- 重复：
  - 执行路径
  - 接收图的局部变化
  - （尽可能少地）更新之前的搜索结果



Lifelong Planning A* (LPA*)

- 数学符号：
  - S 所有节点的集合
  - 节点 s 的后继节点（successors）： $succ(s) \subseteq S$
  - 节点 s 的前驱节点（predecessor）： $pred(s) \subseteq S$
  - 有向边 (s, s') 的成本 ： $0 < c(s, s') \leqslant \infty$
  - 起点： $s_{start} \in S$
  - 终点： $s_{goal} \in S$

LPA* – 实际/估计来路成本

- 实际的 cost-to-come：起点到节点 s 最优路径的成本

$$g^*(s) = \begin{cases} 0 & \text{if } s = s_{start}, \\ \min_{s' \in pred(s)} (g^*(s') + c(s', s)) & \text{otherwise.} \end{cases}$$

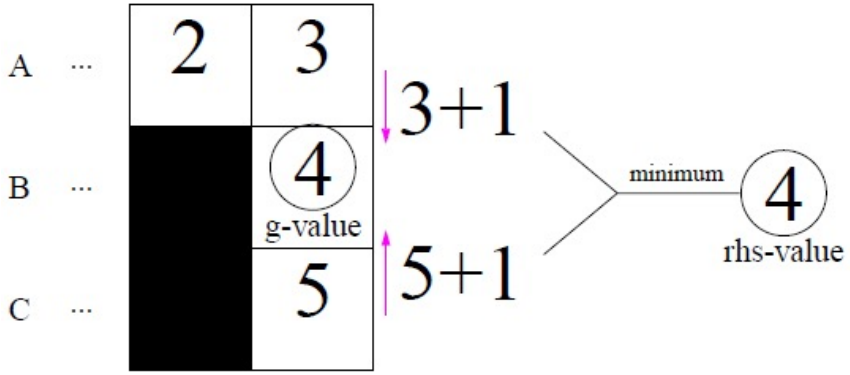
- 估计的 cost-to-come:

$$g(s) \approx g^*(s)$$

LPA* – Right Hand Side Value

- **右手边值 rhs-value** 是基于估计的 g-value **前瞻一步**之后的值：

$$rhs(s) = \begin{cases} 0 & \text{if } s = s_{start}, \\ \min_{s' \in pred(s)} (g(s') + c(s', s)) & \text{otherwise.} \end{cases} \quad (1)$$



- 对照 cost-to-come 真值：

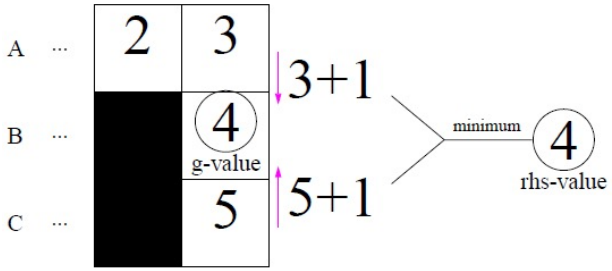
$$g^*(s) = \begin{cases} 0 & \text{if } s = s_{start}, \\ \min_{s' \in pred(s)} (g^*(s') + c(s', s)) & \text{otherwise.} \end{cases} \quad (2)$$

- Rhs计算公式（1）是最优性条件（2）的迭代形式

局部一致性

$$rhs(s) = \begin{cases} 0 & \text{if } s = s_{start}, \\ \min_{s' \in pred(s)} (g(s') + c(s', s)) & \text{otherwise.} \end{cases}$$

- g-value = rhs-value: 节点是**局部一致**的（locally consistent）
- g-value ≠ rhs-value: 节点是**局部不一致**的（locally inconsistent）



- **局部不一致时**：前瞻后的 rhs-value 有望比 g-value 更加趋近真值 g*

LPA* 关键思想

- **优先队列**（priority queue）只包含**局部不一致**的顶点
- **优先值**（priority value，key）根据 **rhs-value**、g-value、h-value 计算
- **优先队列与 g-value 更新规则：**
  - **if** g(s) < rhs(s)，**then:**
    - g(s) = ∞
    - 将 s 和 s 所有的 successor 加入优先队列
  - **If** g(s) > rhs(s)，**then:**
    - g(s) = rhs(s)
    - 将 s 所有的 successor 加入优先队列
  - 第一次规划的优先队列包含所有节点
  - 优先队列始终被保存，每次重新规划时沿用

LPA* 关键思想

终止条件：

- s_{goal} 达到局部一致，并且
- s_{goal} 的估计路径成本（优先值） <= 优先队列中所有路径的估计成本（优先值）
  - 该条件满足 = s_{goal} 被扩展

性质：

- LPA* 收敛后**不保证**所有顶点 g-value 的一致性
- LPA* 只更新 priority queue 中的 g-value （只关心局部一致性）
- LPA* 不保证最优性，但环境变化缓慢时，通常可以达到最优解

LPA* 优先值

- LPA* 优先队列中的优先值由**两部分**组成：

$$k(s) = [\, k1(s);\ k2(s) \,]$$

$$k1(s) = \min(g(s),\ rhs(s)) + h(s)$$

$$k2(s) = \min(g(s),\ rhs(s))$$

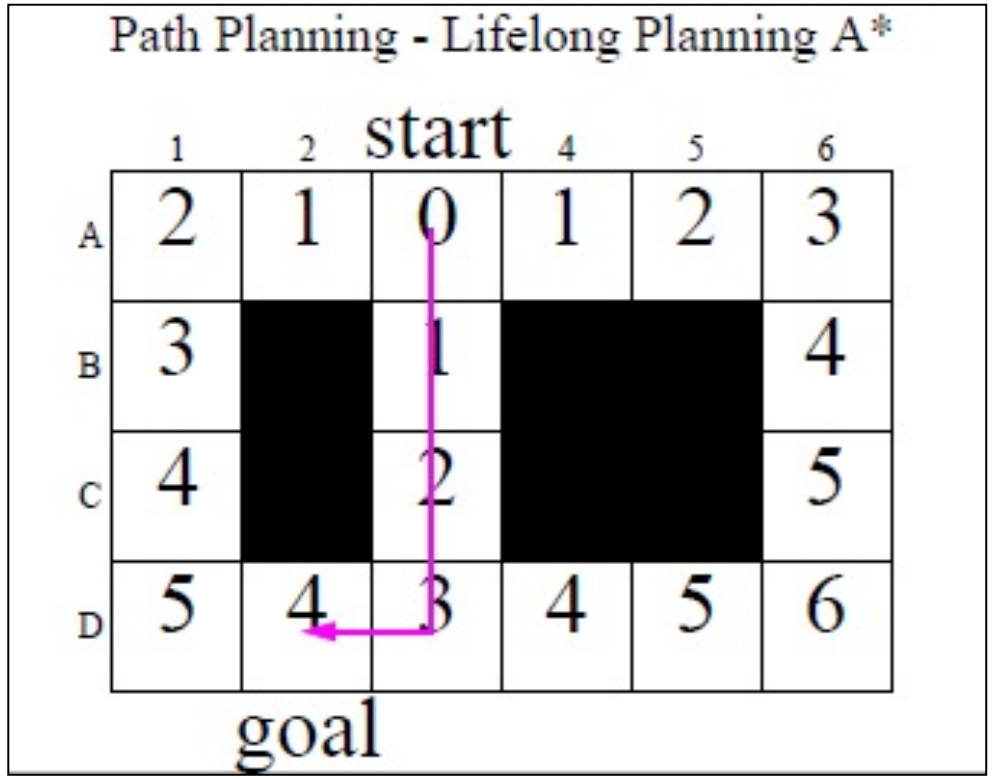
- 节点排序：字典序

$k(s) \le k(s')$ iff : $(\, k1(s) < k1(s') \,)$

or $(\, k1(s) == k1(s') \,)$ and

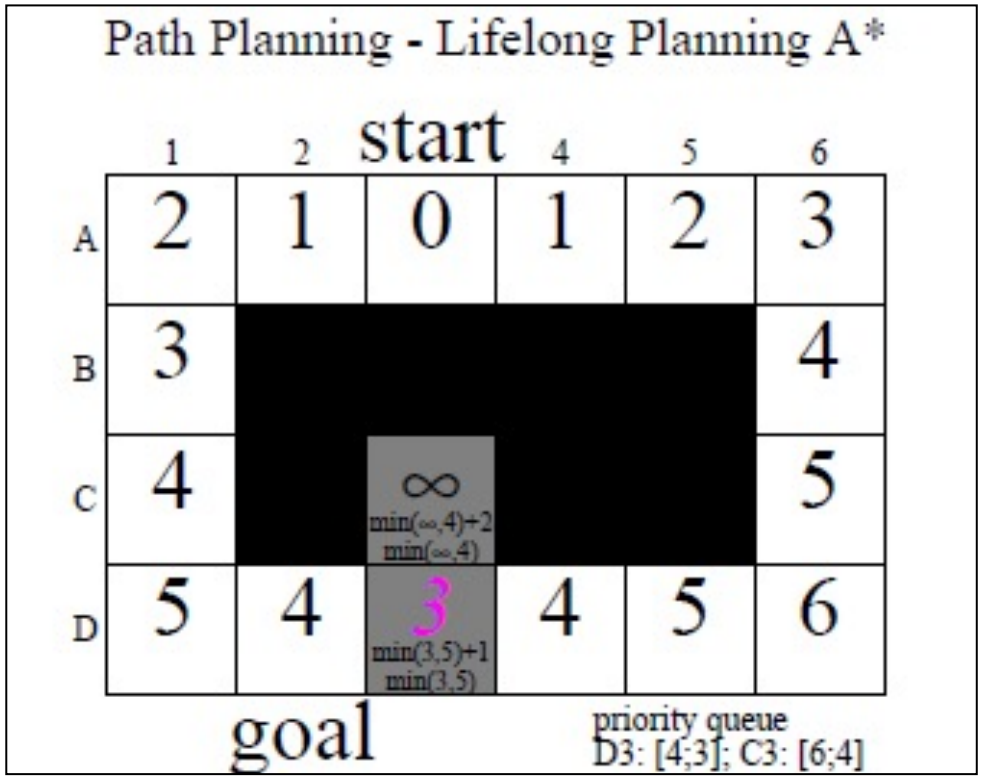
$(\, k2(s) \le k2(s') \,)$

LPA* 演示

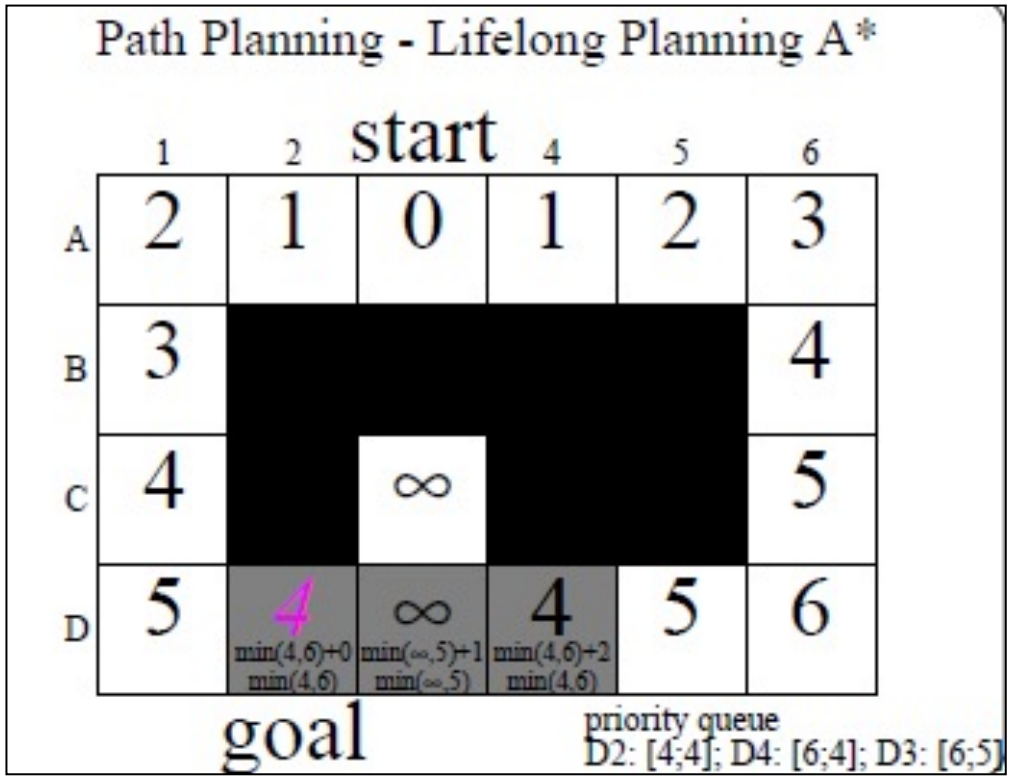


- 发现 C3 局部不一致，加入优先队列，计算优先值
- 优先队列最优节点是 C3: [4,2]
- $k(Goal) > k(\mathrm{C3})$，继续搜索

LPA* 演示

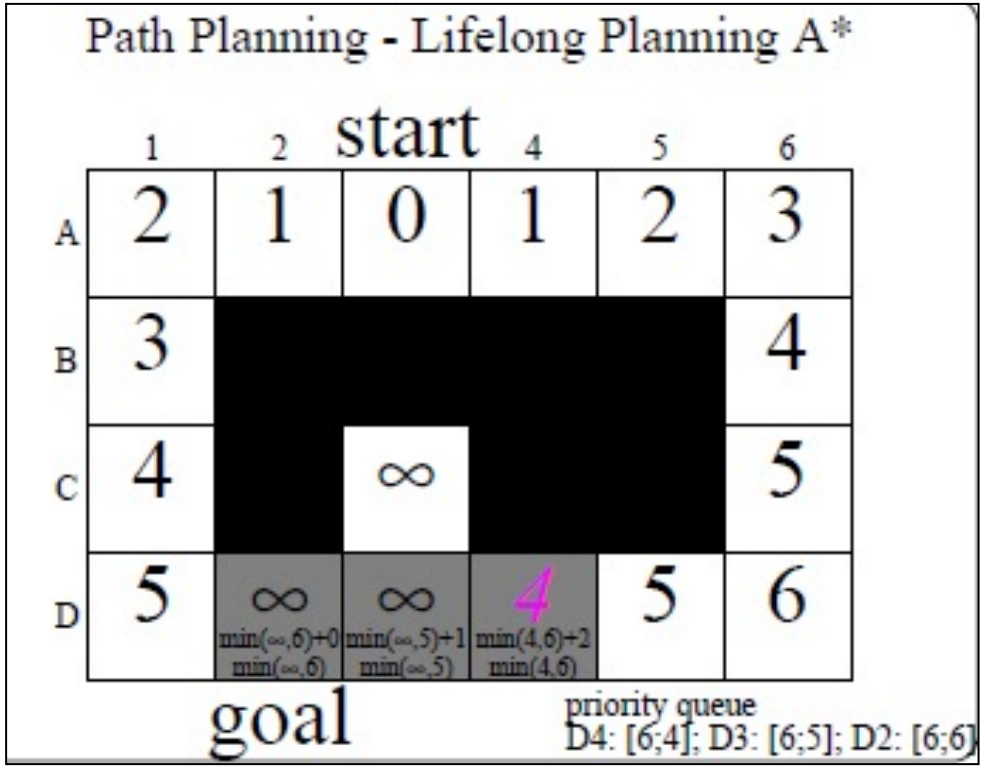


- 由于 C3 的 g-value < rhs-value, 将 g-value 重置为 ∞
- 将重置后的 C3 加入优先队列，计算优先值
- 扩展 C3，将所有后继结点加入优先队列，计算优先值
- 此时优先队列最优节点是 D3: [4,3]
- $k(Goal) > k(D3)$，继续

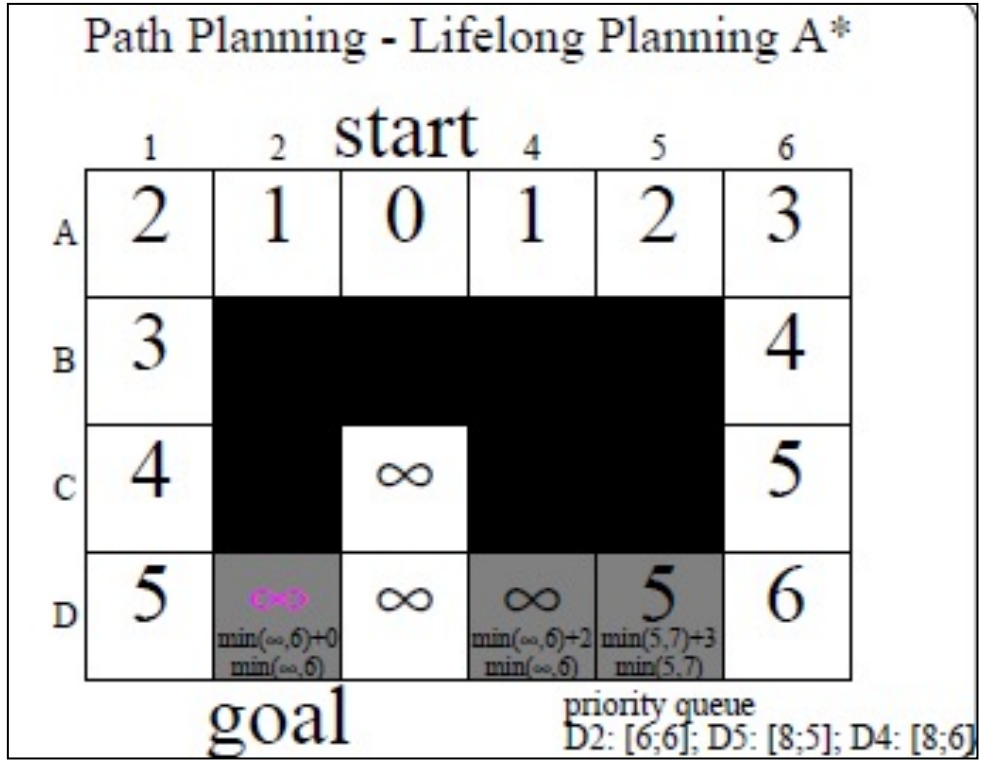


- 由于 D3 的 g-value < rhs-value, 将 g-value 重置为 ∞
- 将重置后的 D3 加入优先队列，计算优先值
- 扩展 D3，将所有后继结点加入优先队列，计算优先值
- 此时优先队列最优节点是 D2 (goal): [4,4]
- Goal 局部不一致，继续

LPA* 演示

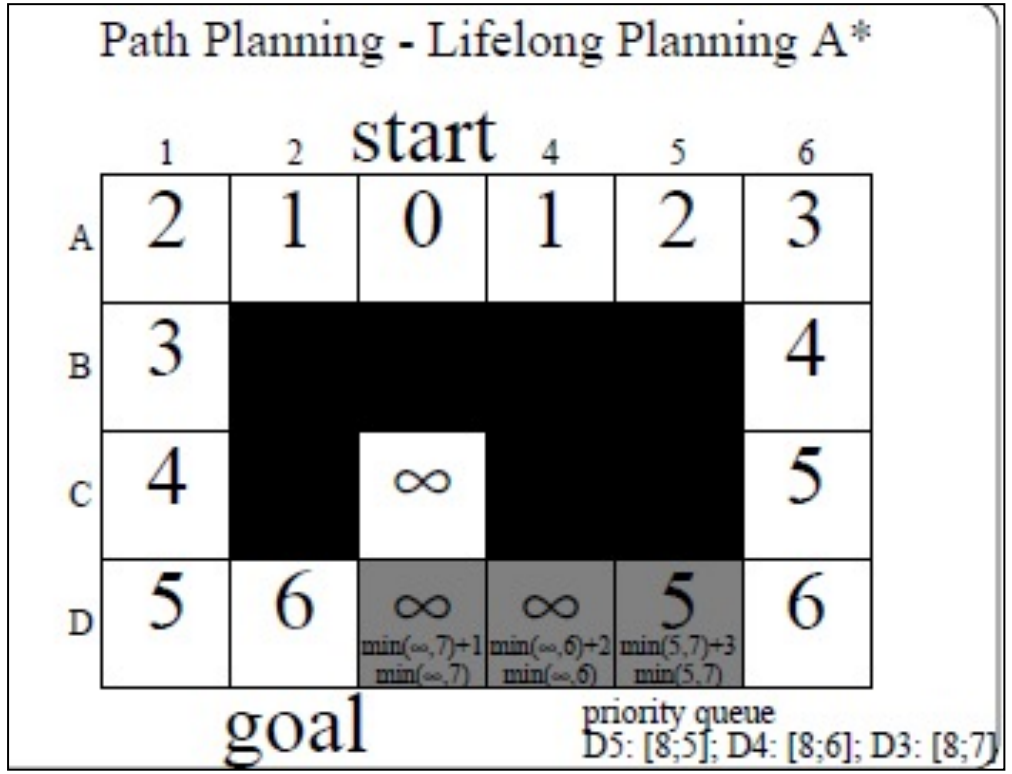


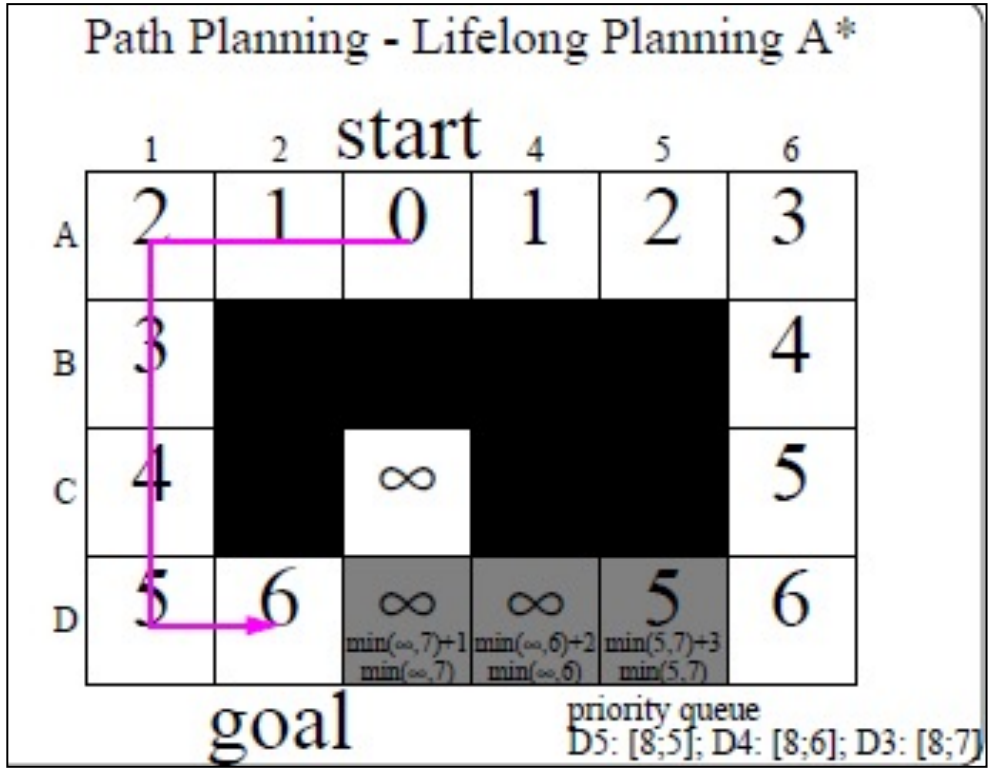
- D2 的 g-value < rhs-value, 将 g-value 重置为 ∞
- D2 是 goal，没有后继结点加入队列
- 将重置后的 D2 加入优先队列，计算优先值
- 此时优先队列最优节点是 D4: [6,4]
- Goal 局部不一致，继续



- 依上述流程扩展完成 D4, D3后，队列顶部回到D2
- D2 的 g-value > rhs-value, 更新 g-value = rhs-value
- D2 是 goal，没有后继结点加入队列
- Goal 局部不一致，继续

LPA* 演示



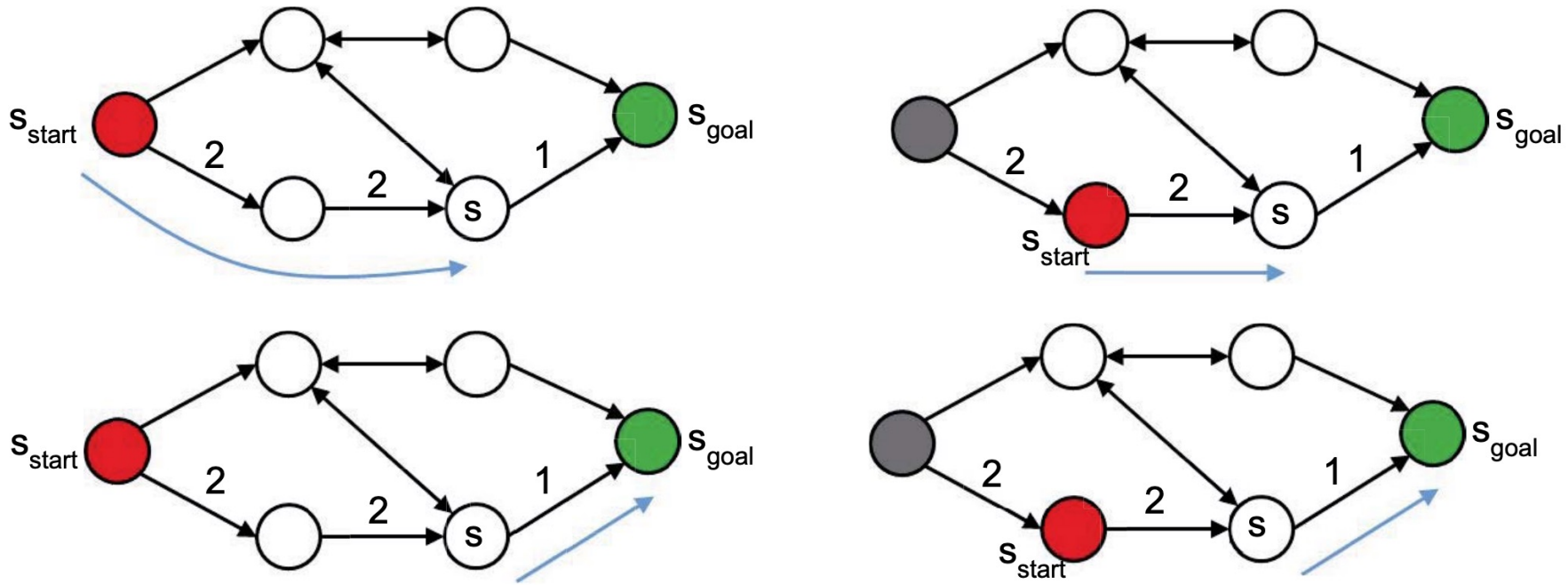


- 达到终止条件：
  - Goal 局部一致
  - $k(\text{Goal}) <= k(\text{D5})$

- 回溯得到最优路径
- 保存优先队列，供下次使用

如果起始点（Start）变化呢？

- 机器人在图上是不断运动的，规划的起始点始终在发生变化，导致原本的 Cost-to-come 失效



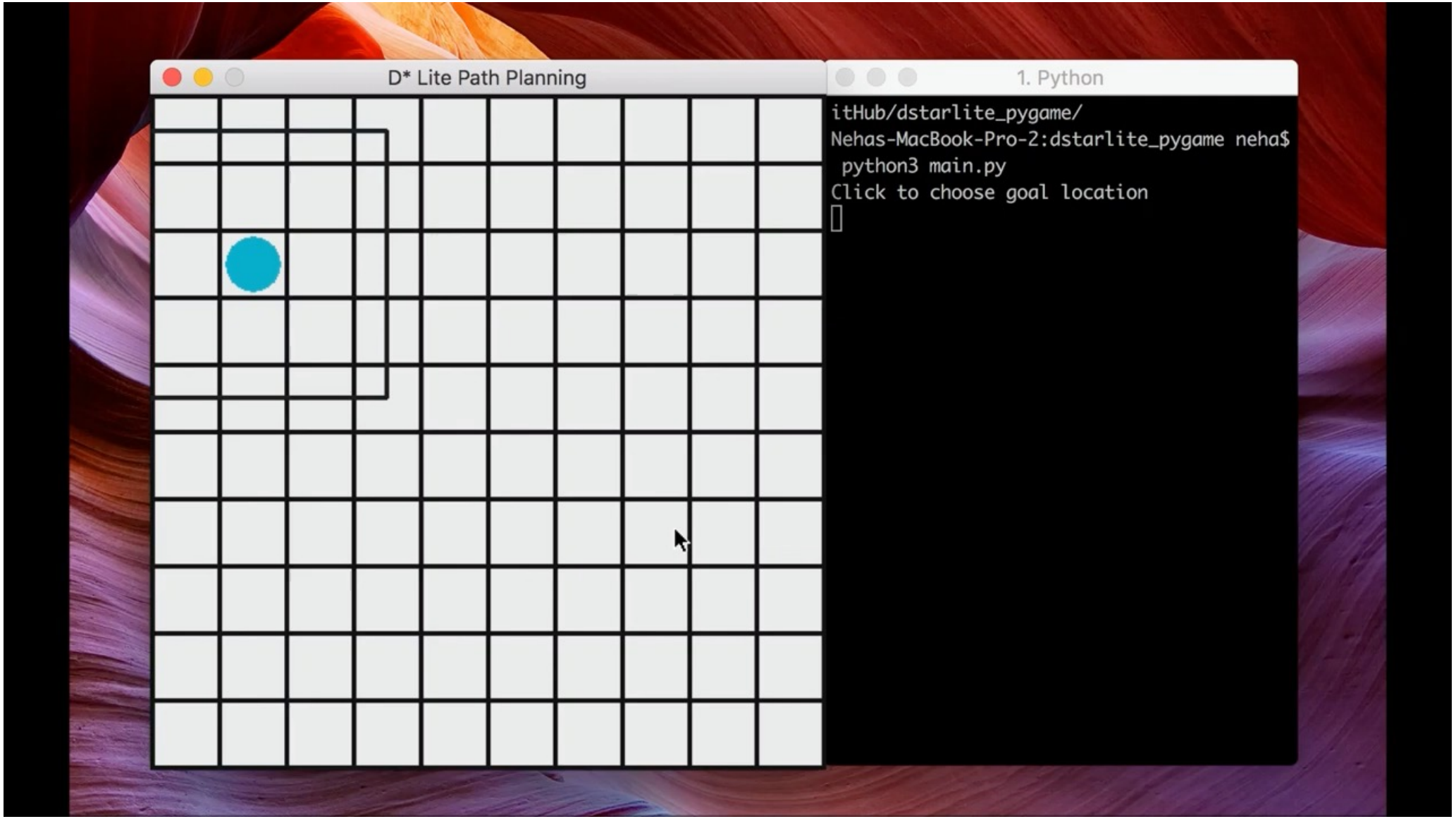
- 但 cost-to-goal 始终不变！
- 将搜索问题逆转，看作**终点->起点**的寻路问题，让 g-value 表示 cost-to-go

D* Lite 算法

- 将所有顶点加入优先队列
- 重复：
  - 进行逆向 LPA* 搜索，直至 Start 局部一致并已经被扩展
  - 从当前节点按最优路径运动下一个节点
  - 如果检测到环境变化：
    - 更新各个边的 cost
    - 更新发生变化节点的前驱节点（predecessors）信息

D* Lite 算法效果

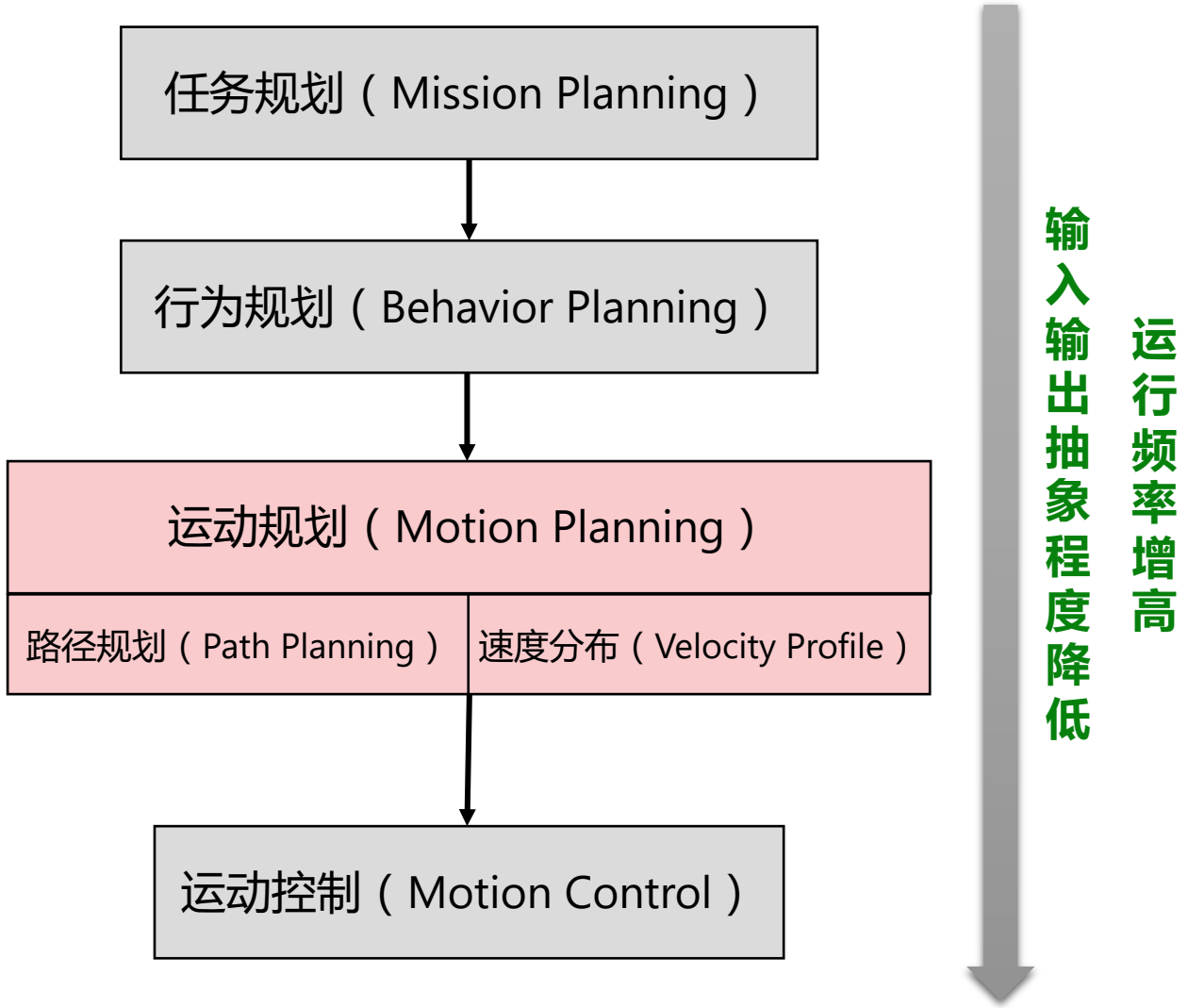
https://youtu.be/FgIIM-QPk-Y



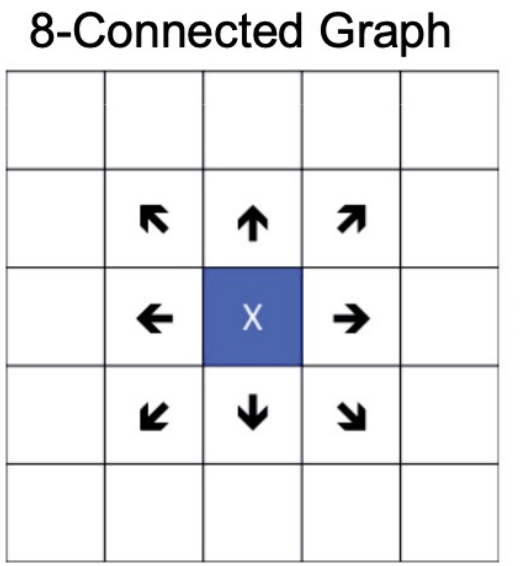
任务规划（Mission Planning）->D* Lite

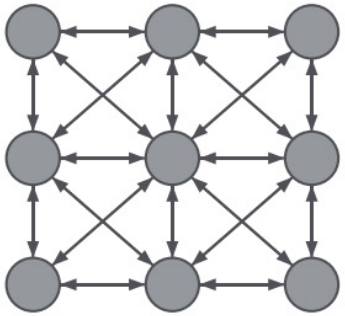
- 输入：
  - 路网图（road network graph ）
  - 起始点（start position）
  - 终点（goal position）
- 输出：
  - 路径：一系列的途径节点
- 优化目标：
  - 最短距离路径
  - 最短时长路径

层级式自动驾驶规划架构



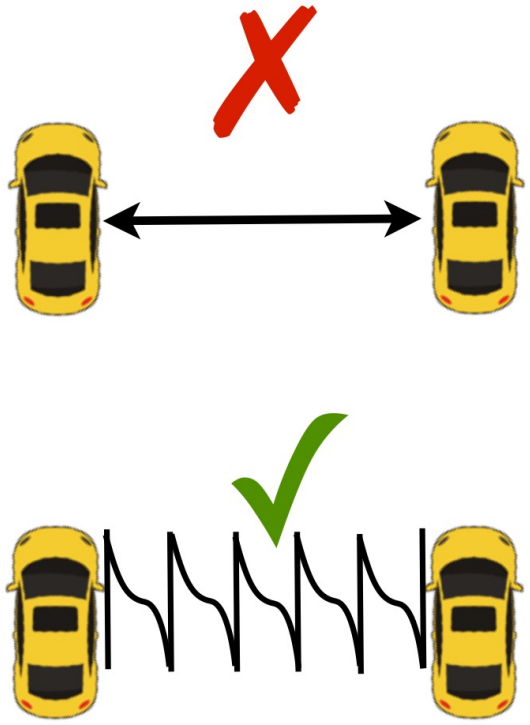
用 A* 或者 D* Lite 实现车辆路径规划？

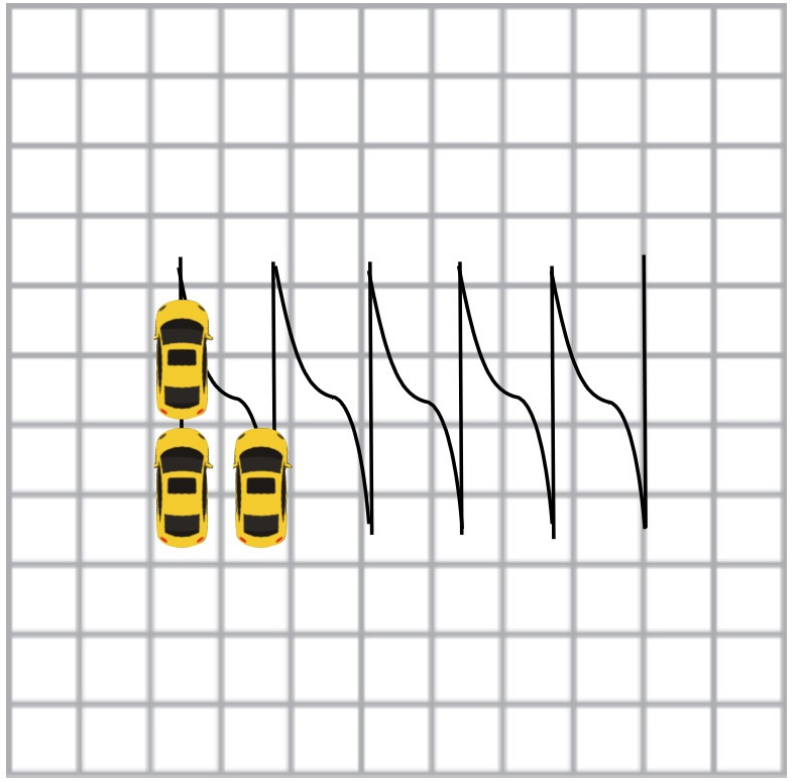




车辆运动无法直接应用 A* 与 D* Lite！

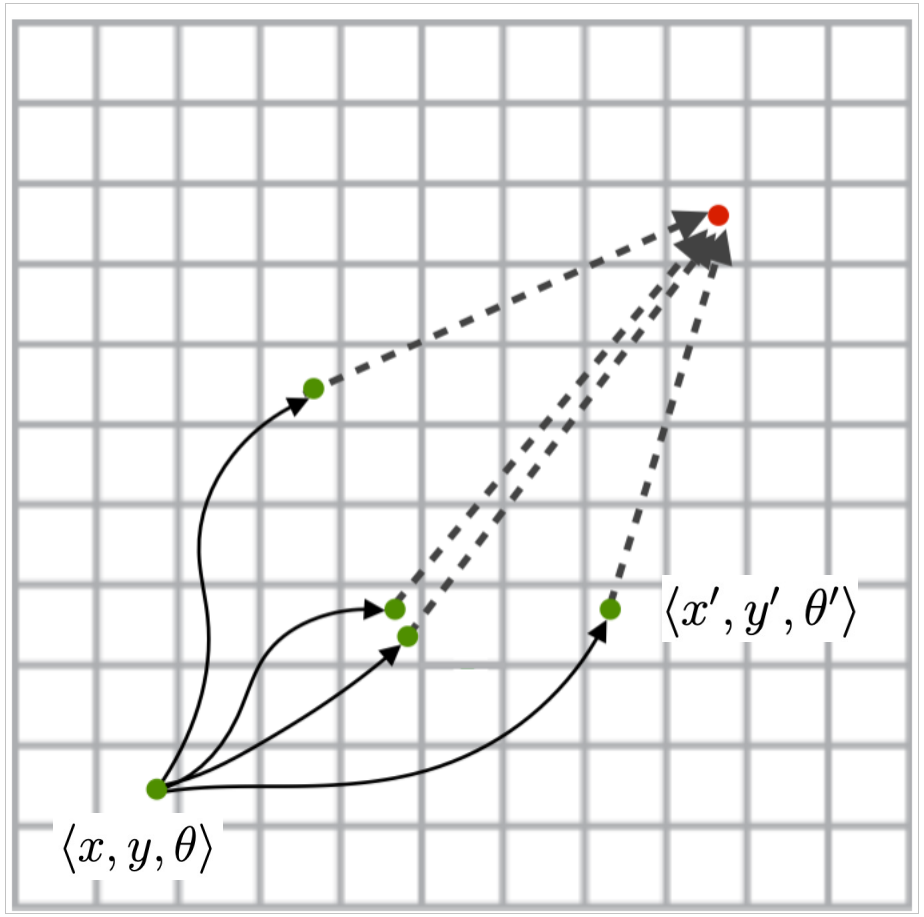
- 车辆运动学具有 nonholonomic constraint，无法按任意方向平移
- Occupancy grid 的连通性质非常复杂





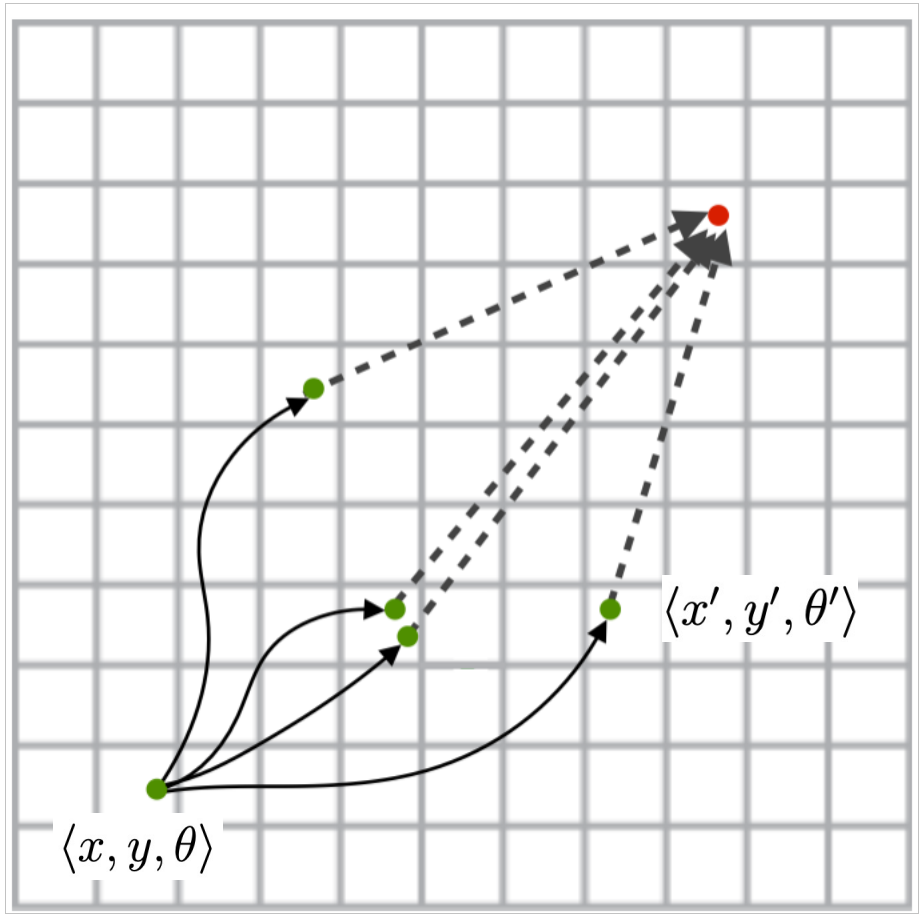
Hybrid A* 算法

- **栅格**：状态(x, y, θ, dir) 构成的空间被离散化为4D 栅格， 每个**离散**的格子同时保存了一个**连续**的车辆状态
  - 格子中心很难被车辆到达
  - 格子中所储存连续状态是车辆可以到达的
  - "Hybrid" 的由来



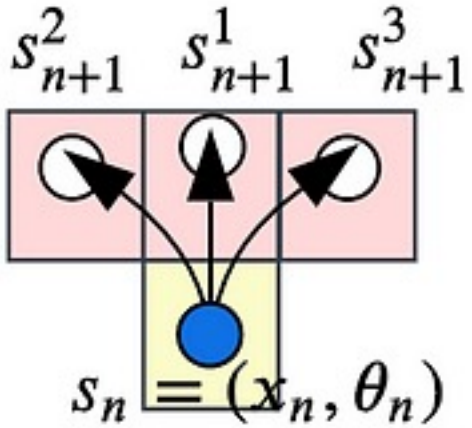
Hybrid A* 算法

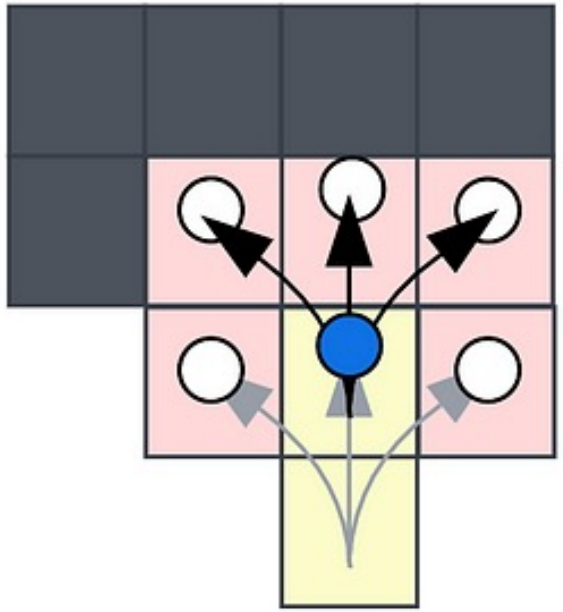
- **节点扩展**：
  - 当前格子 c_i 包含连续状态 q
  - 运行控制 u 可得到不同的后继状态 $q' = f(q,u)$
  - q' 落入另一个 c_j:
    - 若路径 (c_i, c_j) 检测到碰撞：
      - 丢弃 c_j
    - 若 c_j 未被访问：
      - 赋值 q'
      - 加入搜索树
    - 若 c_j 已经被访问 ：
      - c_j 中储存了另一个 $\widehat{q'}$
      - 比较$f(\widehat{q'})$ 与 $f(q')$，取最优的状态

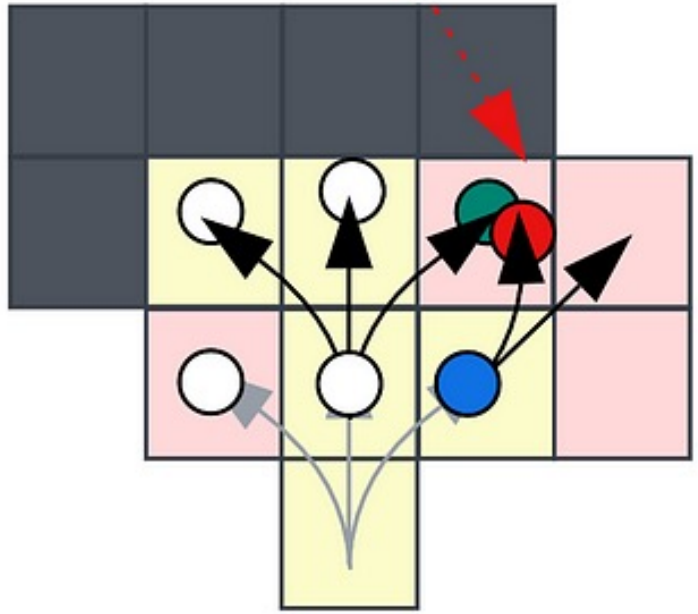


Hybrid A* 树扩展

比较 $f(q) = g(q) + h(q)$
只保留最优的 q







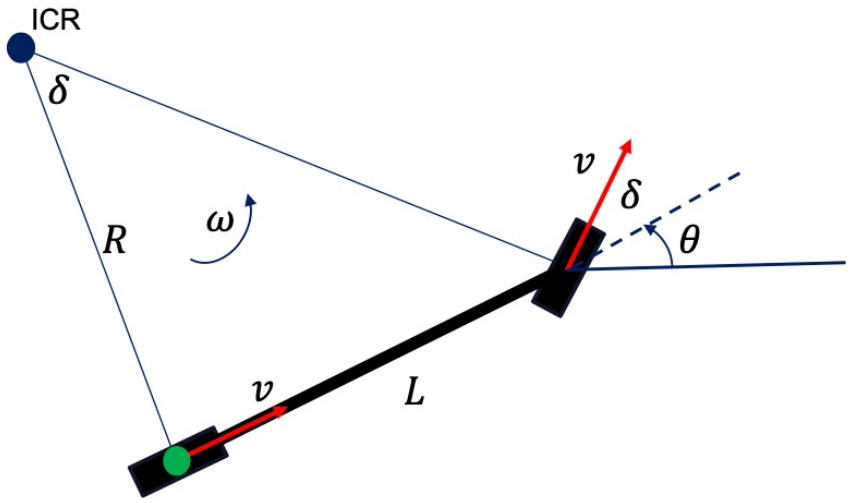
轨迹扩展 $q' = f(q, u)$

- 离散时间的自行车模型展开

$$x_n = \sum_{i=0}^{n-1} v_i \cos(\theta_i)\Delta t = x_{n-1} + v_{n-1}\cos(\theta_{n-1})\,\Delta t$$

$$y_n = \sum_{i=0}^{n-1} v_i \sin(\theta_i)\Delta t = x_{n-1} + v_{n-1}\sin(\theta_{n-1})\,\Delta t$$

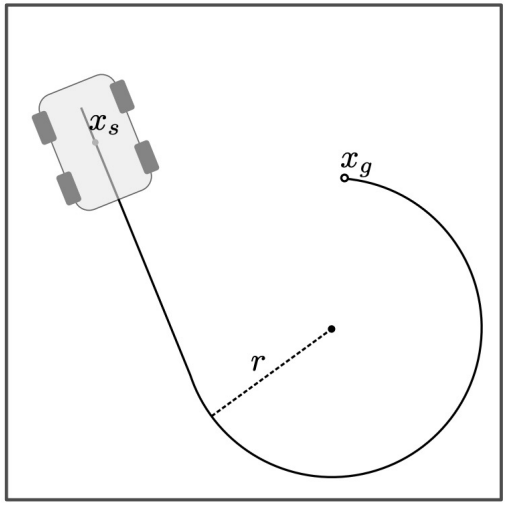
$$\theta_n = \sum_{i=0}^{n-1} \frac{v_i \tan(\delta_i)}{L}\Delta t = \theta_{n-1} + \frac{v_{n-1}\tan(\delta_{n-1})}{L}\Delta t$$

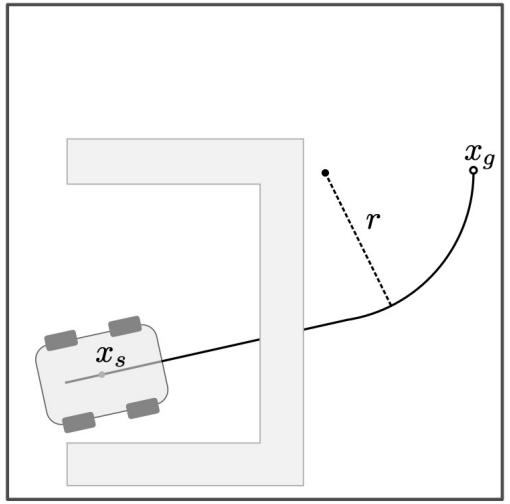


$$\dot{x} = v\cos(\theta)$$
$$\dot{y} = v\sin(\theta)$$
$$\dot{\theta} = \frac{v\tan(\delta)}{L}$$
$$\delta_{min} \le \delta \le \delta_{max}$$
$$v_{min} \le v \le v_{max}$$

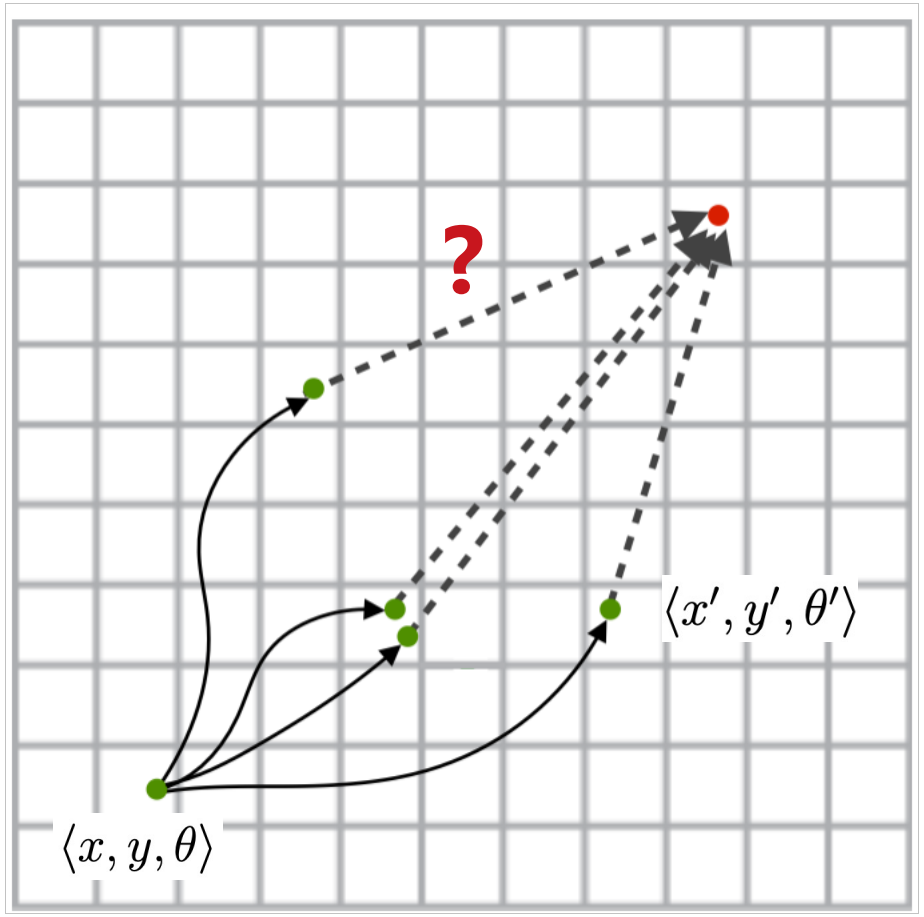
Hybrid A* Heuristics

- 启发函数 <- relaxed problem
- **启发1**：Nonholonomic without obstacles $h_1(q)$
  - 保持非完整约束
  - 去掉所有障碍物





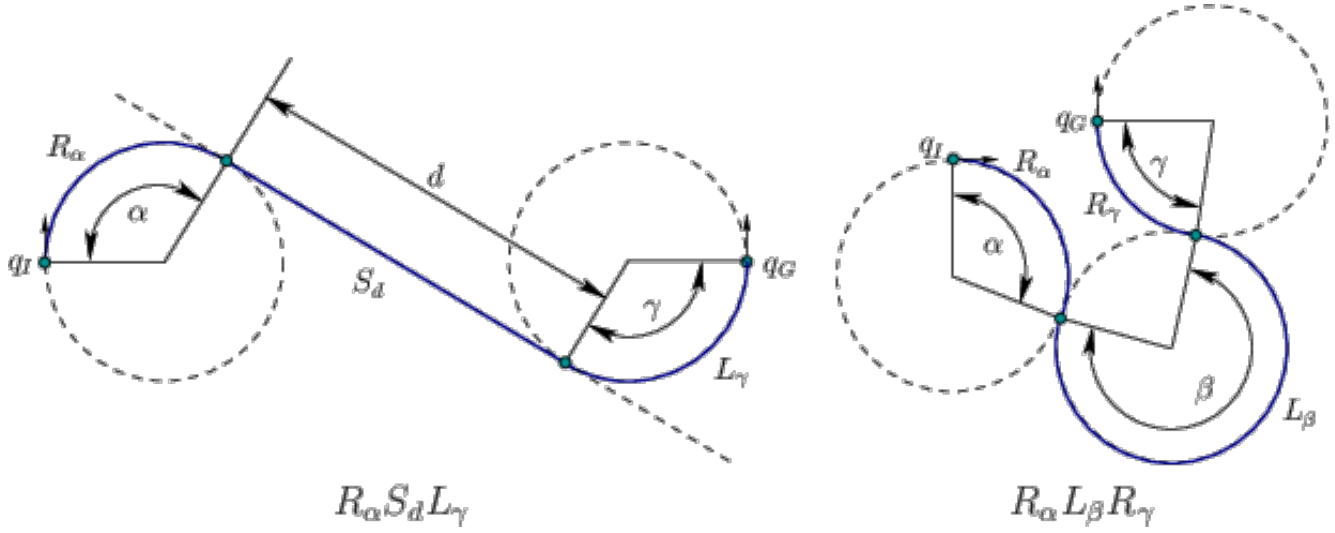
图片来自 Karl's master thesis



Dubins Curves / Reeds-Shepp Curves

- 无障碍物时，车辆最优路径一定取以下六个形式之一：

$$\{LRL,\ RLR,\ LSL,\ LSR,\ RSL,\ RSR\}.$$

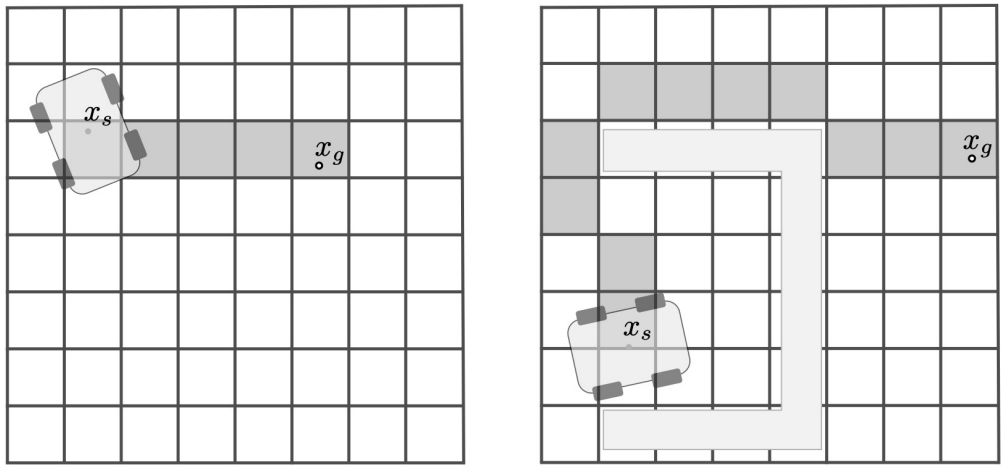


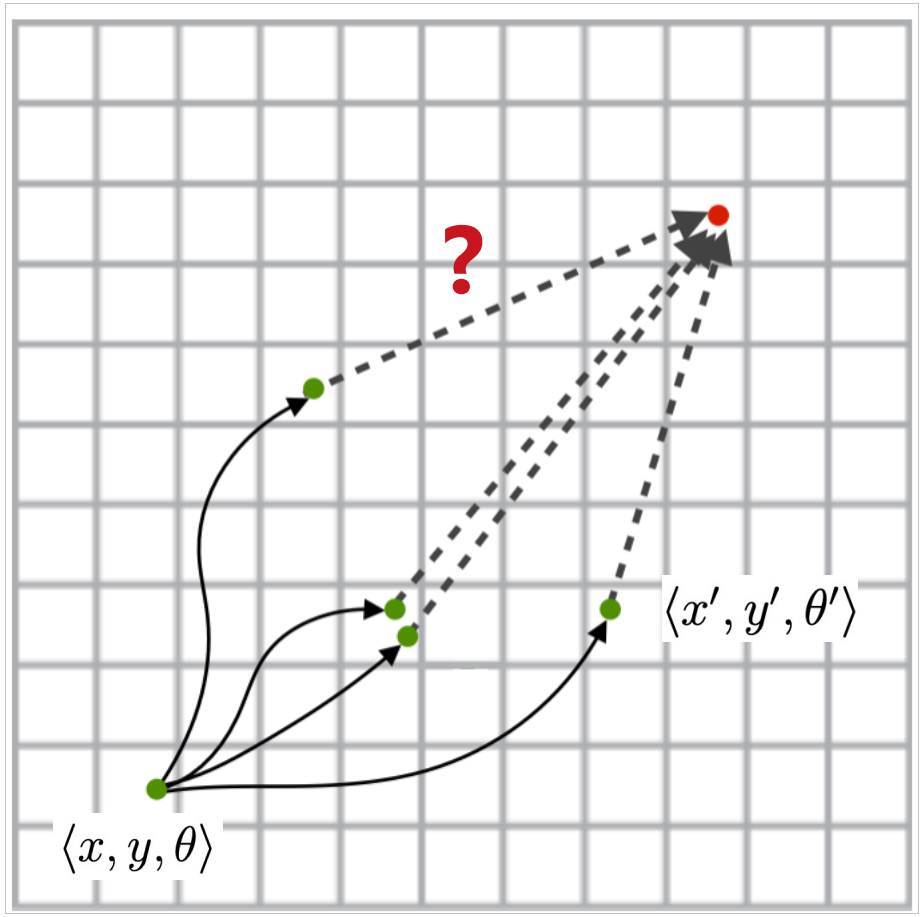
https://lavalle.pl/planning/node821.html
https://lavalle.pl/planning/node822.html

https://github.com/nathanlct/reeds-shepp-curves

Hybrid A* Heuristics

- 启发函数 <- relaxed problem
- **启发2**：Holonomic with obstacles $h_2(q)$
  - 去掉运动学限制 -> A* , D* Lite
  - 保留障碍物

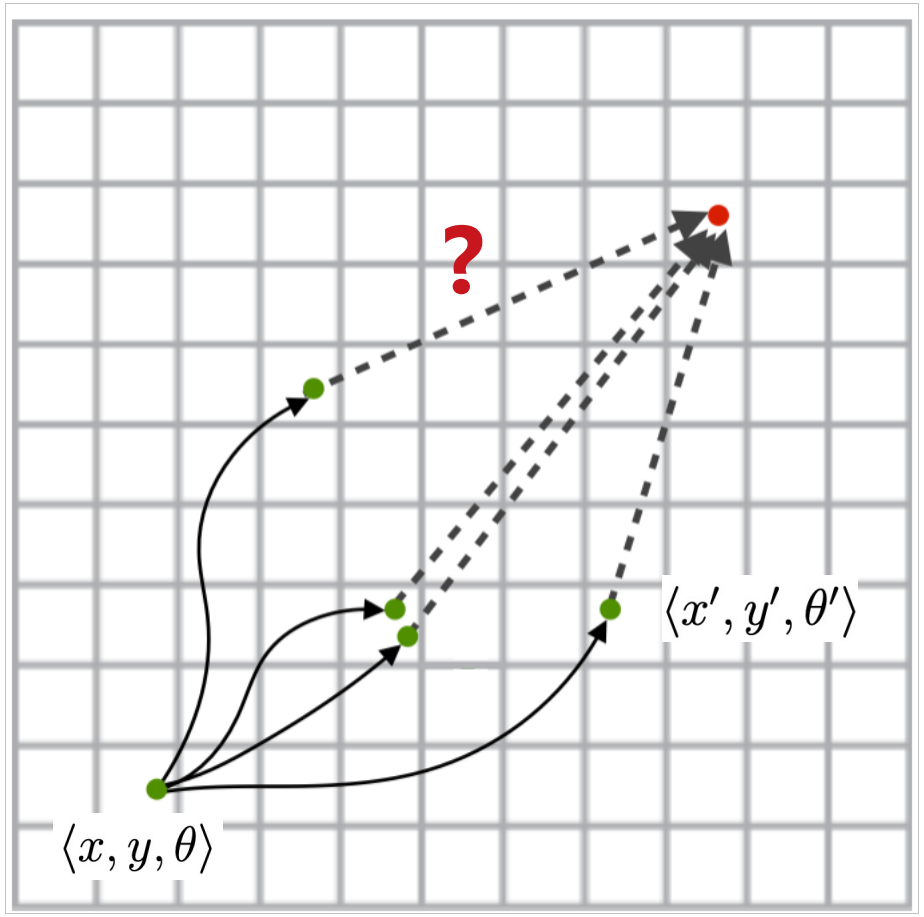




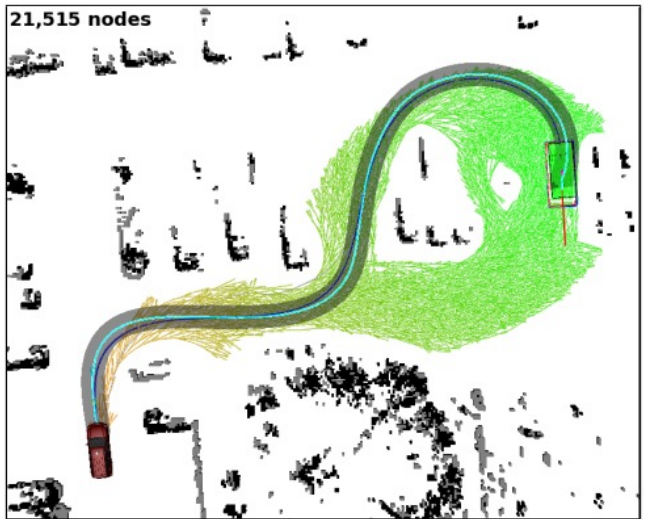
图片来自 Karl's master thesis

Hybrid A* Heuristics

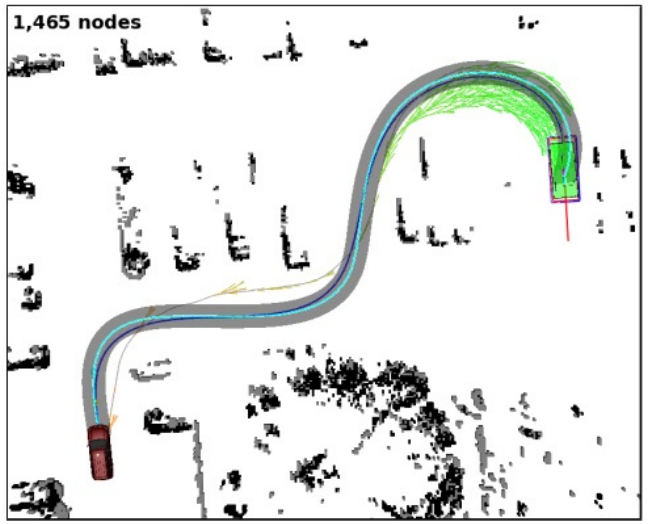
- 启发函数 <- relaxed problem:
- **启发1**：Nonholonomic without obstacles $h_1(q)$
  - 保持非完整约束
  - 去掉所有障碍物
- **启发2**：Holonomic with obstacles $h_2(q)$
  - 去掉运动学限制 -> A*，D* Lite
  - 保留障碍物
- **合并启发**：
  - $\boldsymbol{h(q) = max\{h_1(q), h_2(q)\}}$
  - $h(q)$ 一定比 $h_1(q)$ 和 $h_2(q)$ 更好



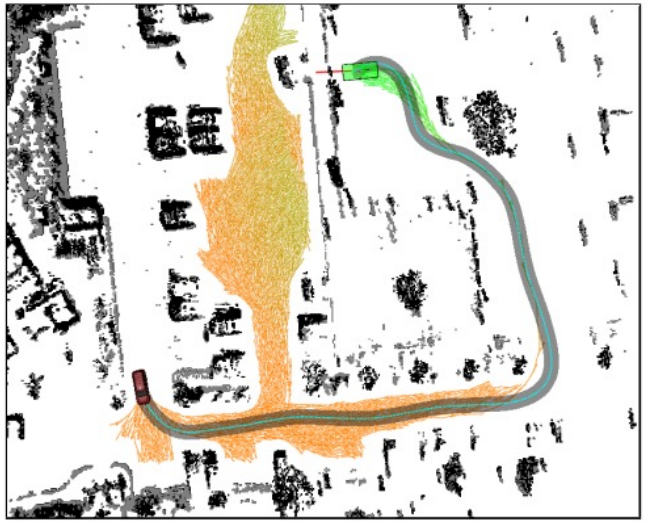
Hybrid A* Heuristics



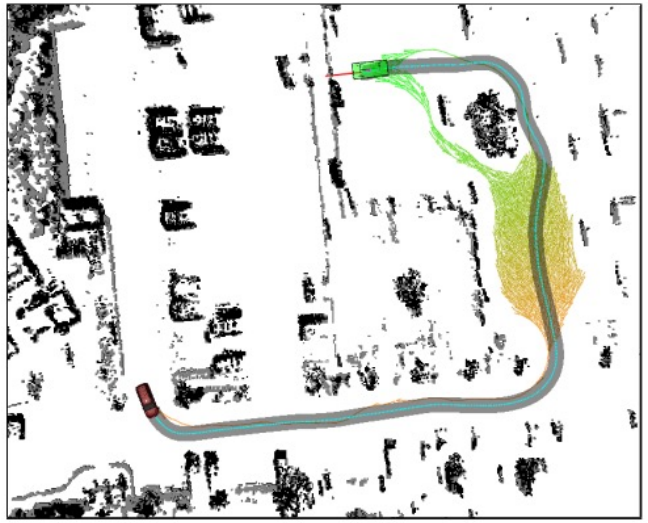
Euclidean Distance



Nonholonomic without obstacles



Nonholonomic without obstacles



+ Holonomic with obstacles

Hybrid A* 效果（DARPA Urban Challenge 2007）

Developed by Dmitri Dolgov, who currently serves as the co-chief executive officer of Waymo

Free-form path planner used by Junior from Standford: https://youtu.be/qXZt-B7iUyw

第八讲总结

- 层级式自动驾驶规划
  - 任务规划：“城市地图中怎么从A到B”
  - 行为规划：“车流中，何时进行跟车、变道、超车、转向…”
  - 运动规划：“以什么轨迹进行变道、超车，…，停车、绕行，…”
    - 路径规划
    - 速度曲线生成
  - 运动控制：“如何控制方向盘、油门、刹车以精准地跟踪参考轨迹”
- 图搜索算法（任务规划、路径规划）
  - 静态图：A*
  - 动态图：LPA*，D* Lite
  - 自动驾驶专用：Hybrid A*

第八讲扩展阅读

- **A*** ：Hart, Peter E., Nils J. Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths." IEEE transactions on Systems Science and Cybernetics 4, no. 2 (1968): 100-107.
- **Lifelong Planning A*** ：Koenig, Sven, Maxim Likhachev, and David Furcy. "Lifelong planning A∗." Artificial Intelligence 155, no. 1-2 (2004): 93-146.
- **D* Lite** ：Koenig, Sven, and Maxim Likhachev. "Fast replanning for navigation in unknown terrain." IEEE transactions on robotics 21, no. 3 (2005): 354-363.
- **Hybrid A*** ：Montemerlo, Michael, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel et al. "Junior: The stanford entry in the urban challenge." Journal of field Robotics 25, no. 9 (2008): 569-597.
- **Hybrid A*** ：Dolgov, Dmitri, Sebastian Thrun, Michael Montemerlo, and James Diebel. "Path planning for autonomous vehicles in unknown semi-structured environments." The international journal of robotics research 29, no. 5 (2010): 485-501.
- **Parking Planner based on Hybrid A***: Li, Bai, Tankut Acarman, Youmin Zhang, Yakun Ouyang, Cagdas Yaman, Qi Kong, Xiang Zhong, and Xiaoyan Peng. "Optimization-based trajectory planning for autonomous parking with irregularly placed obstacles: A lightweight iterative framework." IEEE Transactions on Intelligent Transportation Systems 23, no. 8 (2021): 11970-11981.