# 量产自动驾驶中的决策规划

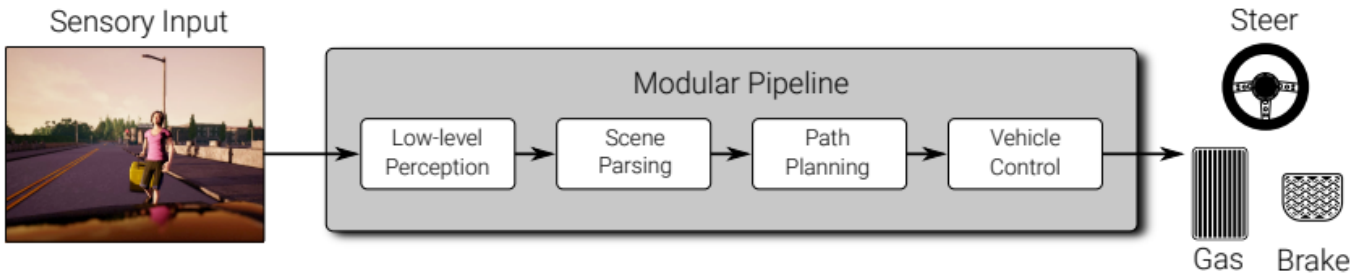
## 从领航高速(NOP)到领航城区(City-NOP)

肖飞宇, 3/10/2023

# 引言

## 自动驾驶中的决策规划

# Planning and Decision Making



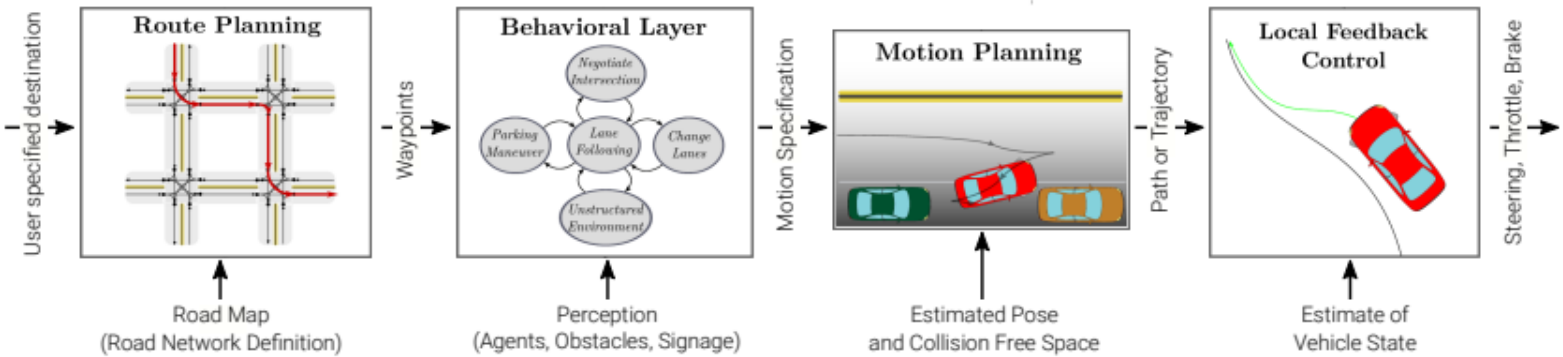

## Problem Definition

- Goal: find and follow path from current location to destination
- Take static infrastructure and dynamic objects into account
- Input: vehicle and environment state (via perception stack)
- Output: Trajectory as input to vehicle controller

## Challenges:

- Driving situations and behaviors are very complex
- Difficult to model as a single optimization problem

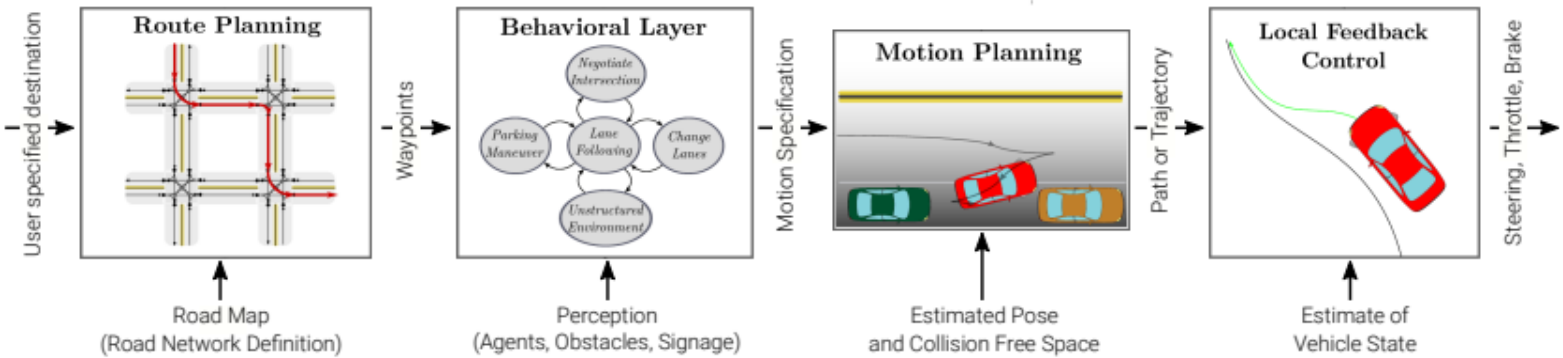
# Planning and Decision Making




- Idea: Break planning problem into a **hierarchy of simpler problems**
- Each problem tailored to its scope and level of abstraction
- Earlier in this hierarchy means higher level of abstraction
- Each optimization problem will have constraints and objective functions

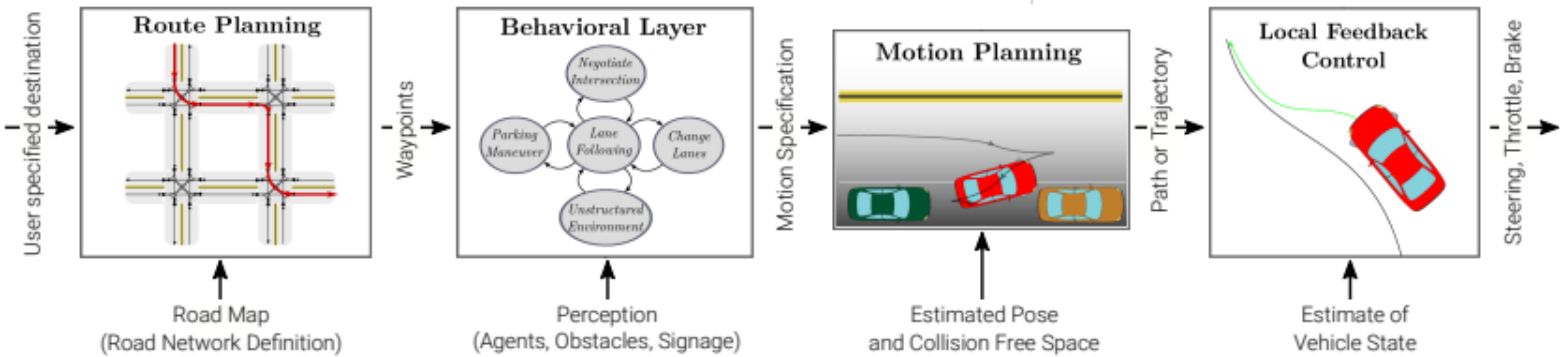
[1]Paden et al.: A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles. TIV, 2016.

# Planning and Decision Making




**Hierarchy**: A destination is passed to a route planner that generates a route through the road network. A behavioral layer reasons about the environment and generates a motion specification to progress along the selected route. A motion planner then solves for a feasible motion accomplishing the specification. A feedback control adjusts actuation variables to correct errors in executing the reference path.

# Step 1: Route Planning

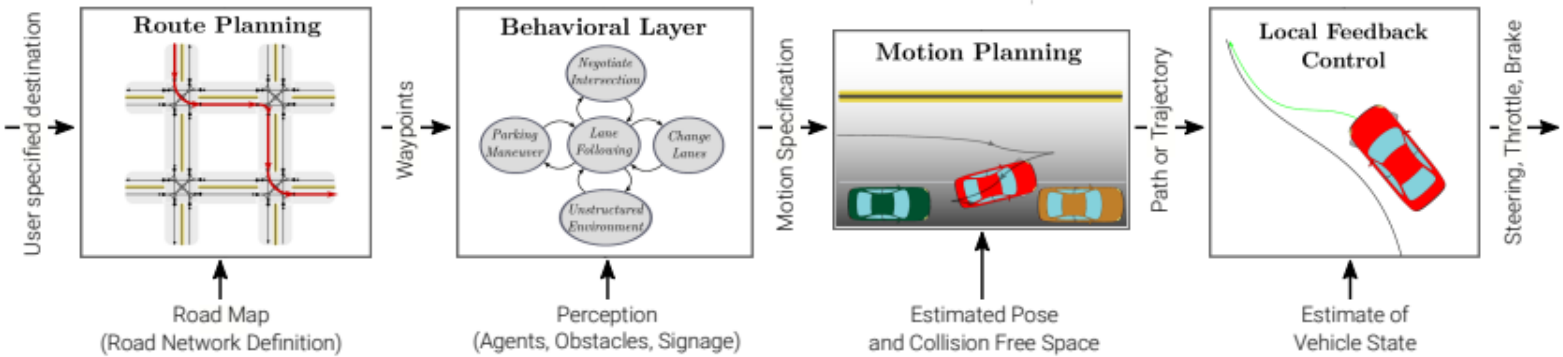



- Represent **road network** as **directed graph**.
- Edge weights correspond to road segment length or travel time
- Problem translates into a minimum-cost graph network problem
- Inference algorithms: Dijkstra, $A^*$

**Solved** for industrial applications like Map Apps.

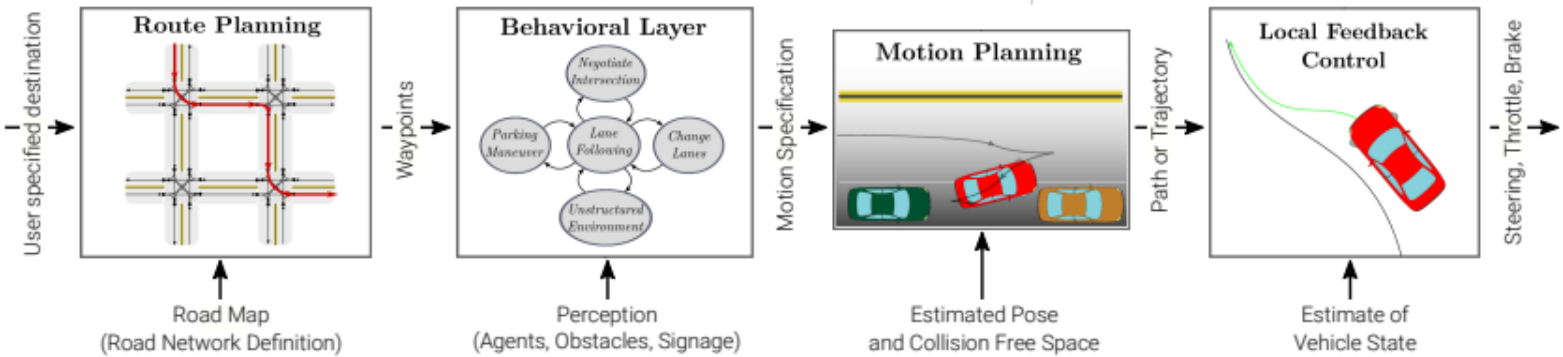
[1]Delling, Daniel et al. “Customizable Route Planning.” The Sea (2011).

# Step 2: Behavior Planning




- Select **driving behavior** based on current vehicle/environment state
- E.g. at stop line: stop, observe other traffic participants, traverse
- Often modeled via **finite state machines** (transitions governed by perception)
- Can be modeled probabilistically, e.g., using Markov Decision Processes (MDPs)

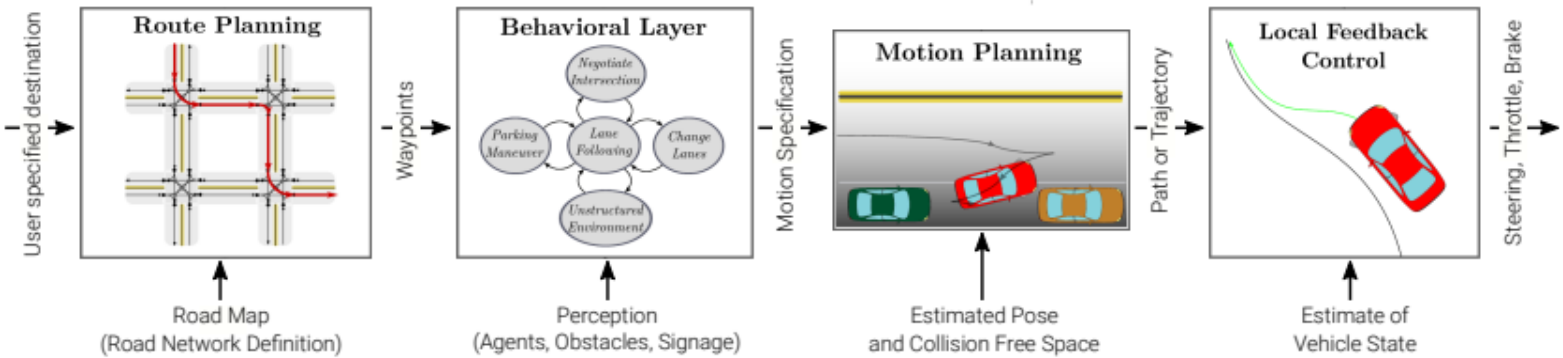
# Step 3: Motion Planning




- Find feasible, comfortable, safe and fast vehicle trajectory
- Exact solutions in most cases computationally intractable
- Often numerical approximations are used
- Approaches: variational methods, graph search, incremental tree-based

[1] Note: find feasible not optimal trajectory.

# Step 4: Local Feedback Control




- **Feedback controller** executes the trajectory from the motion planner
- Corrects errors due to inaccuracies of the vehicle model
- Emphasis on **robustness, stability and comfort**
- *Open-loop planning* plus *feedback control* gets robust excuation.

[1] Note: find feasible not optimal trajectory.

# Behavior Planning

An overall introduction

# Behavior Planning

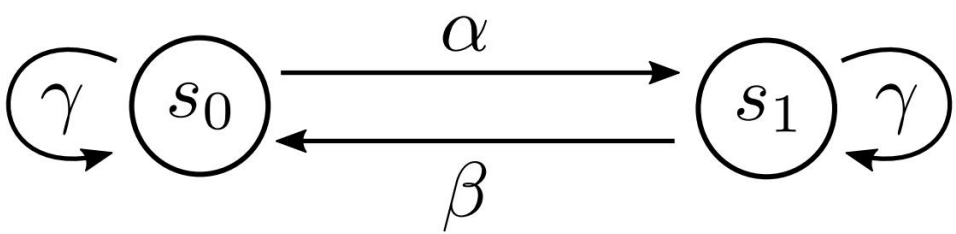
- To follow a planned route, the vehicle must conduct **various maneuvers**
- Examples include: speed tracking, car following, stopping, merging, etc.
- It is difficult to design a motion planner for all maneuvers jointly
- The **behavior planning** stage thus **discretizes the behaviors** into simpler (atomic) maneuvers, each of which can be addressed with a dedicated motion planner
- The behavior layer must take into account traffic rules, static and dynamic objects
- **Input**: High-level route plan and output of perception stack
- **Output**: Motion planner constraints: corridor, objects, speed limits, target, ...
- Frequently used models:
  - Deterministic: Finite State Machines (FSMs) and variants
  - Probabilistic: Markov Decision Processes(RL related)

# Finite State Machine

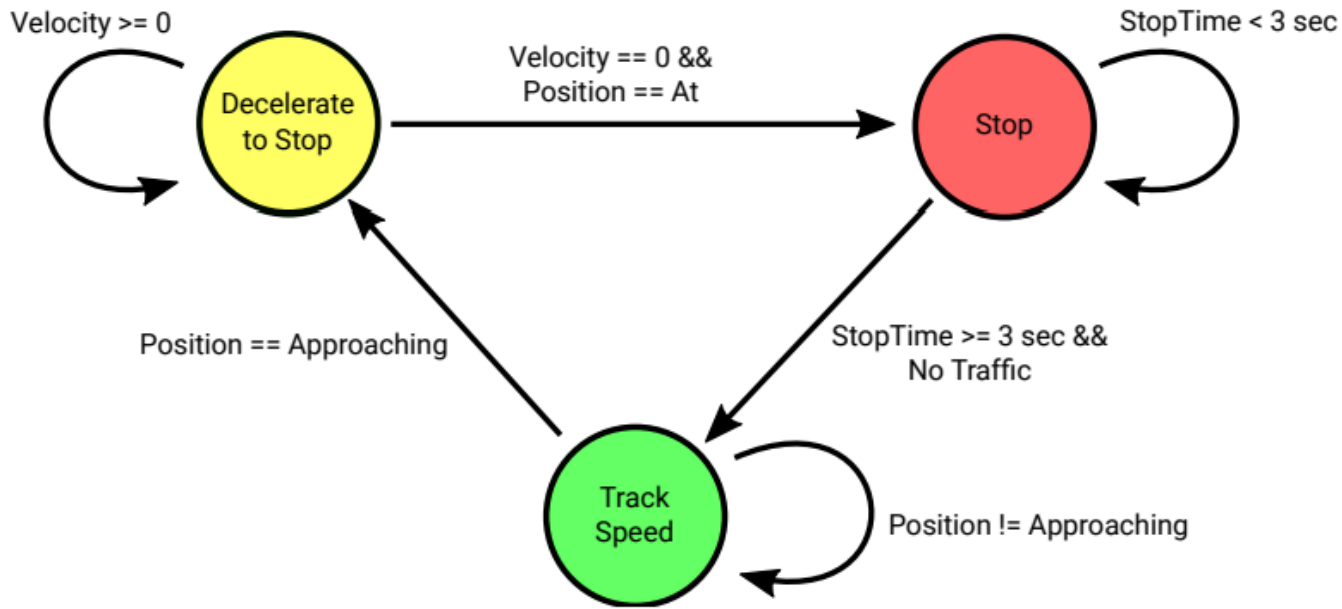
A Finite State Machine (FSM) is defined by quintuple $(\sum, \mathrm{S}, \mathrm{F}, s_0, \delta)$

- $\sum$ is the input alphabet
- $\mathrm{S}$ is a non-empty set of states
- $\mathrm{F} \subset \mathrm{S}$ is the set of final states
- $s_0 \in \mathrm{S}$ is the initial state
- $\sigma : \mathrm{S} \times \sum \to \mathrm{S}$ is the state transition function

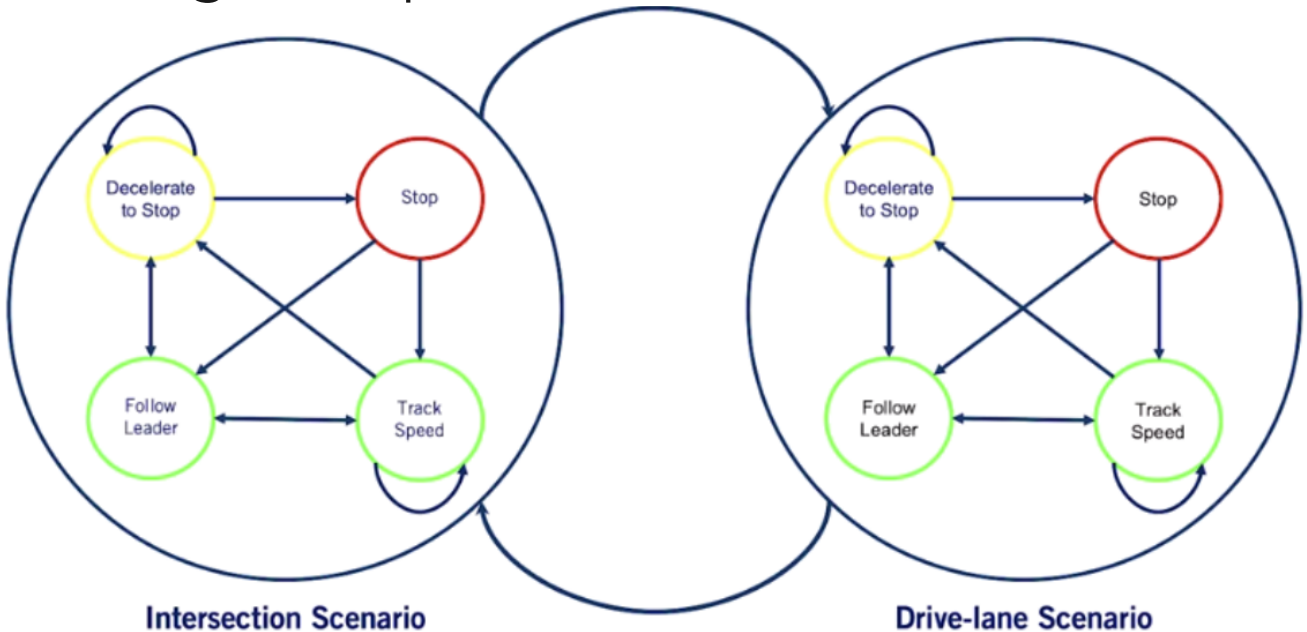
Example:



# FSM for a Simple Vehicle Behavior

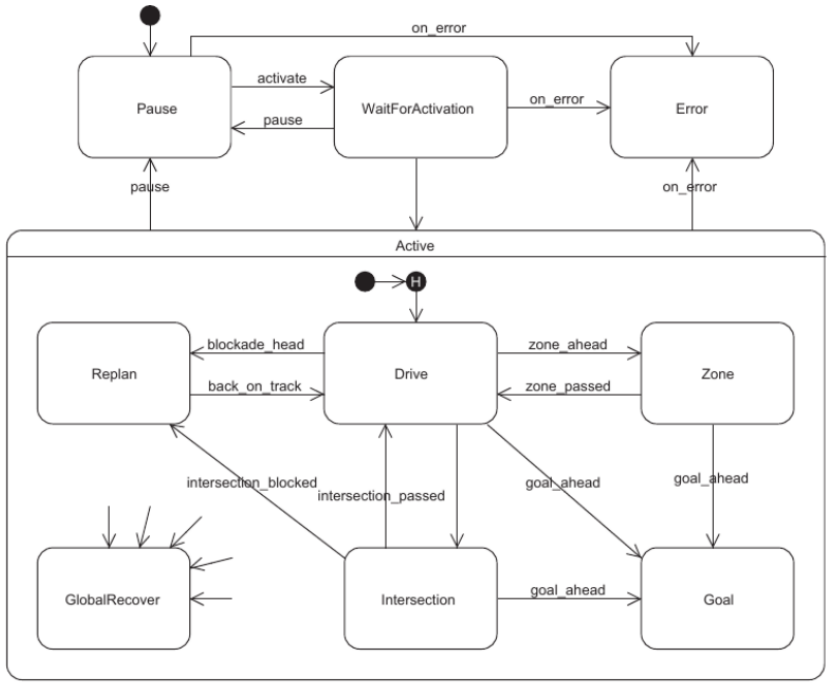


# Handling Multiple Scenarios

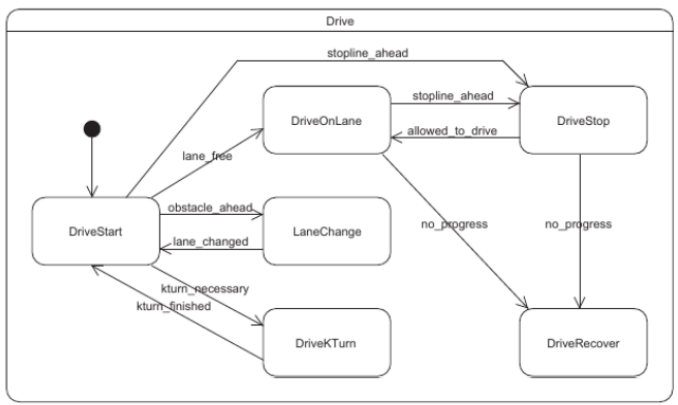



- Hierarchical State Machine (HSM)
- Advantages: Simpler, more efficient – Disadvantages: Rule duplication

# Example from DARPA Challenge




**(a)** State chart of the main level of the FSM.




**(b)** Substate of the state *Drive*.

[1] Gindele et al.: Design of the planner of team AnnieWAY's autonomous vehicle used in the DARPA Urban Challenge 2007. IV, 2008.

# Summary Finate State Machines

- Elegant way to break complex behaviors into simpler maneuvers
- Interpretable and easy to design
- Rule explosion when dealing with complex scenarios
- **Cannot handle noise / uncertainty** $\rightarrow$ MDPs
- **Expert-designed hyperparameters** $\rightarrow$ Reforcement Learning

Widely used in autonomous systems for highway driving(in most cases), not suitable for urban driving.

Note: FSM based decision system is usually denoted as rule-based decision system.

# Motion Planning

An overall introduction

# Motion Planning

**Goal**:

- Compute safe, comfortable and feasible trajectory from the vehicle's current configuration to the goal based on the output of the behavioral layer
- Local goal: center of lane a few meters ahead, stop line, parking spot
- Takes as input static and dynamic obstacles around vehicle and generates collision-free trajectory

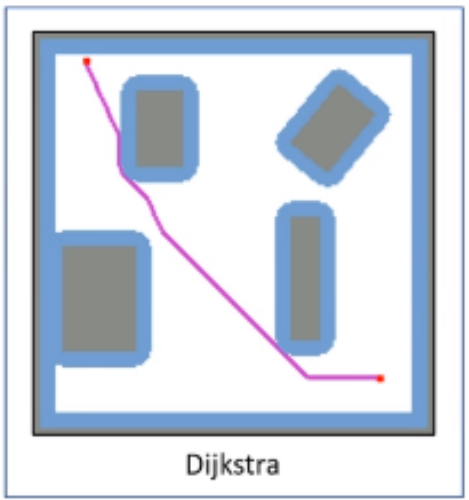
**Focus on Trajectory not only Path**

- Path: $\sigma(l) : [0, 1] \rightarrow \mathcal{X}$ (does not specify velocity)
- Trajectory: $\pi(t) : [0, T] \rightarrow \mathcal{X}$ (explicitly considers time)
- **Completeness of planning**: in the configuration space $\mathcal{X}$ of the vehicle and $T$ the planning horizon

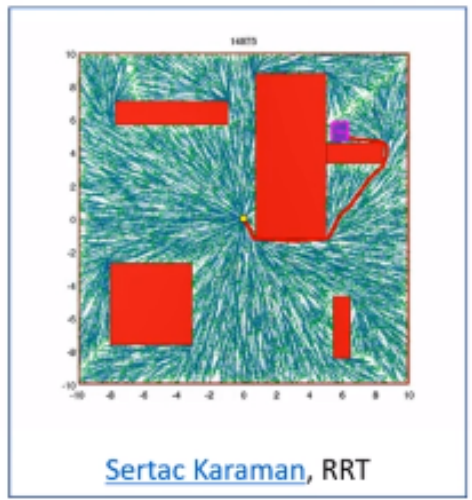
# Motion Planning

**Main Formulations**:

- **Path planning**
  - Deterministic: Dijkstra, A*, D*, D* lite/LPA*, and etc.
  - Randomized: PRM,RRT,RRT*,FMT,BIT*,and etc.



Dijkstra



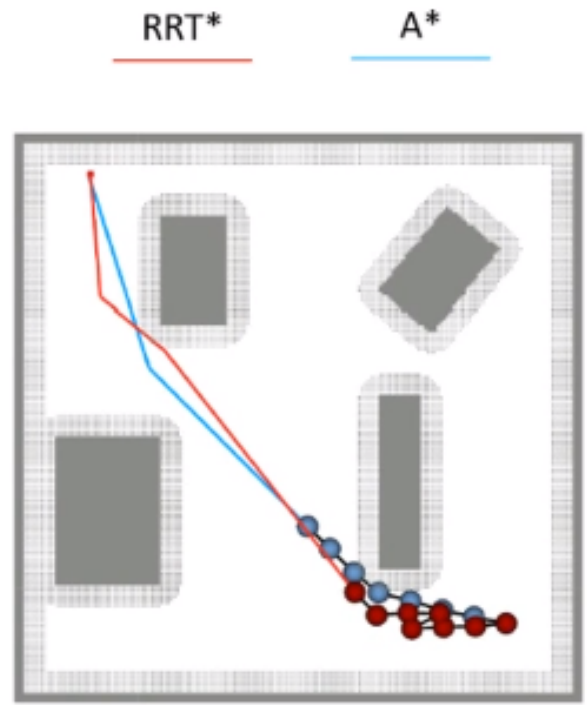
Sertac Karaman, RRT

Not dynamically feasible

# Deterministic vs Randomized

Dijkstra, A*, D*,
D* Lite/LPA*, ARA*

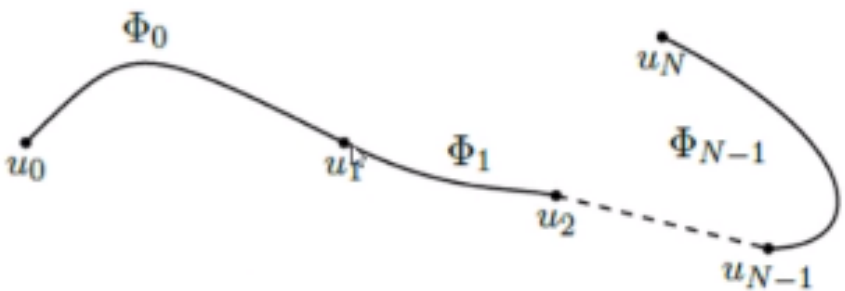
PRM, RRT, RRT*, FMT,
Informed-RRT*, BIT*

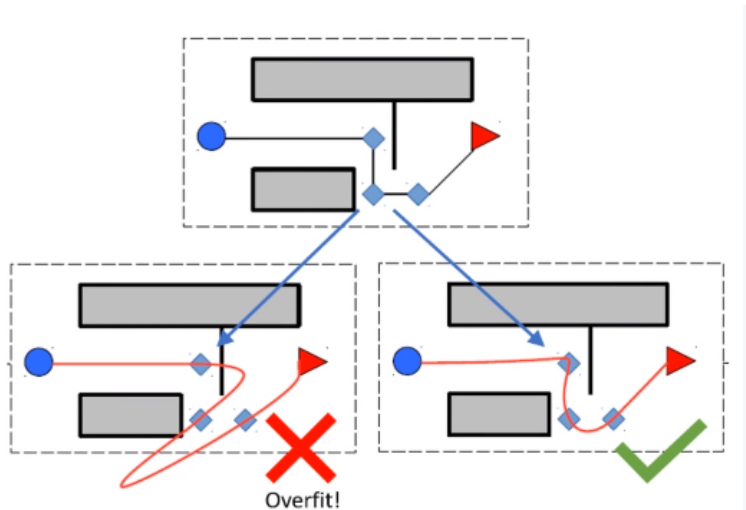


# Optimization-based Trajectory Planning

- **Piece-wise Polynomial**

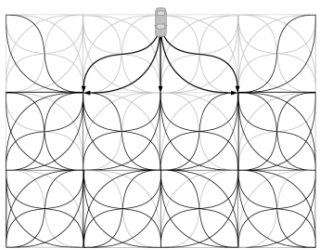
$$f(t) = \begin{cases} f_1(t) \doteq \sum_{i=0}^{N} p_{1,i} t^i & T_0 \le t \le T_1 \\ f_2(t) \doteq \sum_{i=0}^{N} p_{2,i} t^i & T_1 \le t \le T_2 \\ \vdots & \vdots \\ f_M(t) \doteq \sum_{i=0}^{N} p_{M,i} t^i & T_{M-1} \le t \le T_M \end{cases}$$

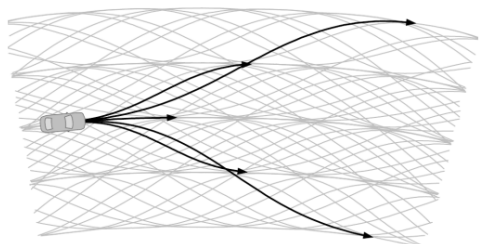


# Search-based Trajectory Planning

- Idea:
  - Discretize configuration space into graph
- Global Optimality





$$J(D) := \int_0^T \|u(t)\|^2 \, dt = \int_0^T \left\| p_D^{(n)}(t) \right\|^2 dt$$

$$\arg\min_{D,T} \; J(D) + \rho T$$

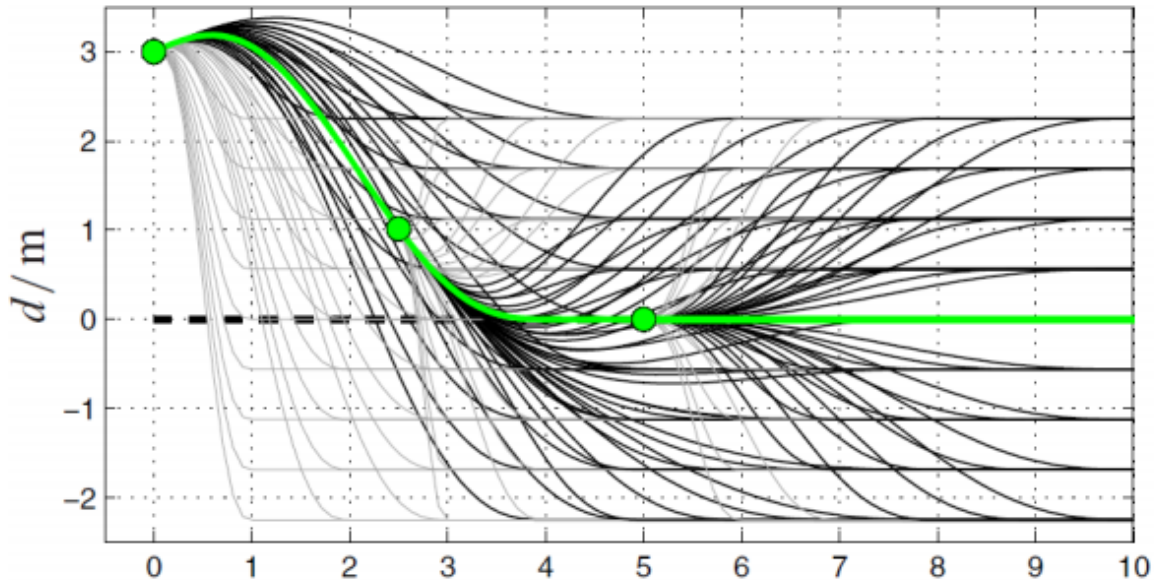
$$s.t. \quad \dot{x}(t) = Ax(t) + Bu(t), \quad \forall t \in [0,\ T]$$

$$x(0) = x_0, \quad x(T) \in \mathcal{X}^{goal}$$

$$x(t) \in \mathcal{X}^{free}, \quad u(t) \in \mathcal{U}, \quad \forall t \in [0,\ T]$$

# Search-based Trajectory Planning

- Motionprimitives:
  - Existing work: sampling in state space

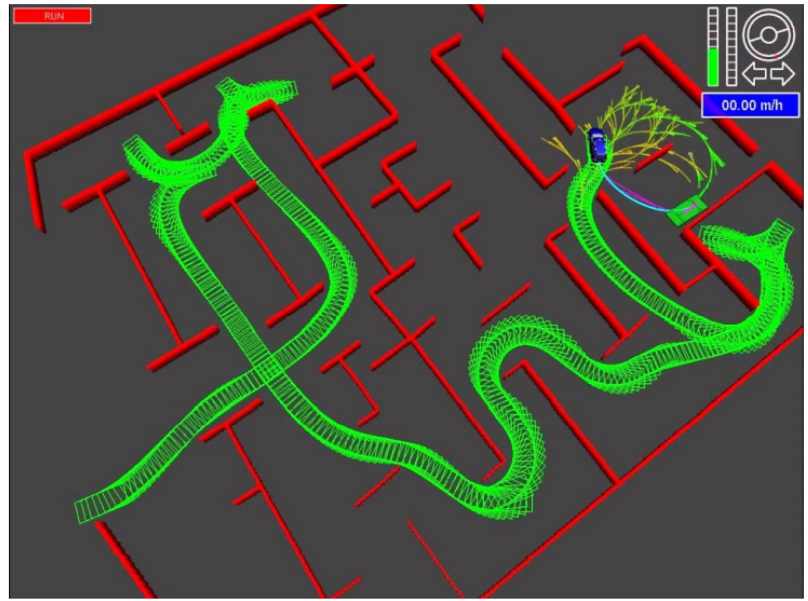


# Hybrid $A^*$

- Hybrid A* is an A* variant that guarantees kinematic feasibility of the path
- Planning is re-applied continuously as the car explores the environment
- A practical method for **kinematic motion planning**

**Kinodynamic : Kinematic + Dynamic**

- Coarse-to-fine proces
- Trajectory only optimizes locally
- Infeasible path means nothing to nonholonomic system




Practical Search Techniques in Path Planning for Autonomous Driving. STAIR, 2008.[1]

# Summary

- Driving situations and behaviors are very complex
- Thus, we break the problem into a hierarchy of simpler problems:
  - Route planning, behavior planning and motion planning
  - Each problem is tailored to its scope and level of abstraction
- A* exploits planning heuristics to improve efficiency
- Behavior planning can be implemented using finite state machines
- For motion planning, variational and graph search methods are often used

# 量产 L2+ 中的决策规划算法实践

以高速领航辅助驾驶(NOP)为例

# Motion planning in L2+

**How to evaluate motion planner**

- Feasibilty: whether the planning results are executable or not for AVs
- Safety: whether the planning results are collision-free or not
- Optimality: whether the planning result is optimal or sub-optimal
- Completeness: if there exists a solution, whether the planner is able to find it
- Run time: how much time it takes to find the planning result

# L2+ 决策规划场景：量产自动驾驶功能

- 智能行车辅助：
  - 近距离加塞应对（Close Cut-in Handling）
  - 智能避障（Intelligent Collision Avoidance）
  - 拨杆变道（Driver Triggered Lane Changing）
- 领航高速驾驶：
  - 高级超车辅助（Advanced Overtaking Assistant）
  - 自动上下匝道（Ramp to Ramp）
  - 智能调速（Intelligent Speed Limit）
- 领航城区驾驶：
  - 路口处理（Intersection Handing）
  - 绕行避障（By Pass Driving）

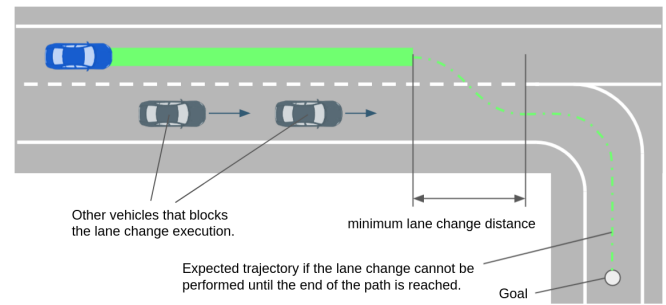
# 高速 NOP 中的决策

## 决策模块的输出

- 路径的长度以及左右边界限制
- 路径上的速度限制
- 时间上的位置限制（Recall:轨迹和路径的区别）

## 高速 NOP 中需要决策的场景

- 抢行还是让行
- 是否要主动变道
- 在什么位置进行变道、并入那两辆车之间
- 如何绕行前方障碍物/是否借道
- 何时从匝道汇入主路
- more ...

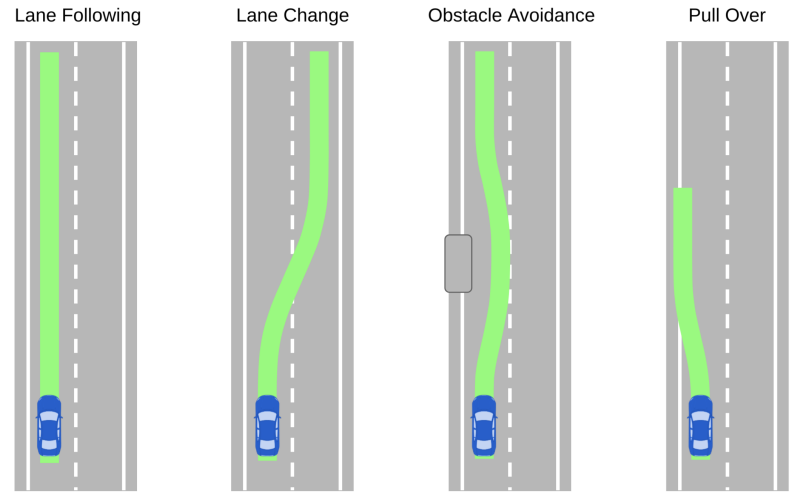



决策模块相当于无人驾驶系统的大脑，保障无人车的行车安全，同时也要理解和遵守交通规则。 为了实现这样的功能，决策模块为下游的规划模块提供了各种的限制信息

# 决策：车道决策

## 常用算法

- 基于规则的有限状态机
  - 状态之间的转移通常用规则进行判断
  - 适用于低级别自动驾驶与简单高速场景
- 基于轨迹评价的方法
  - 核心思想： 基于（粗）规划的结果进行决策
  - 保证决策可以为规划提供一个合适的解空间
  - 常用评价指标：
    - 轨迹越光滑越好
    - 距离周围车辆越远越好
    - 距离目标越近越好
  - 适用于高级别自动驾驶和复杂城区道路
  - 评价函数可以通过数据驱动的方式持续改进




决定是否应该保持当前车道、借道或者变道[1]

# 决策：车道决策

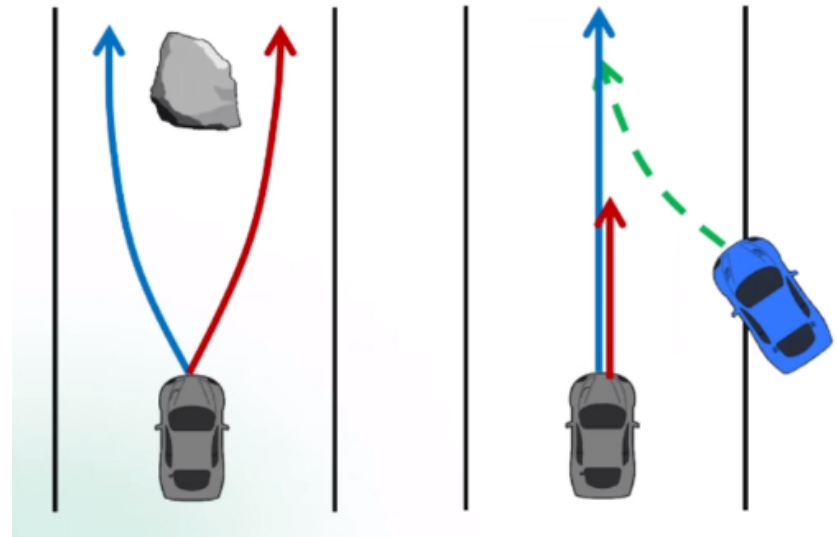
**常用算法**

- 基于规则（适用于单一障碍物的简单场景）
- 基于搜索的方法如 A*，DP 等（适用于多障碍物复杂场景）

**难点与挑战**

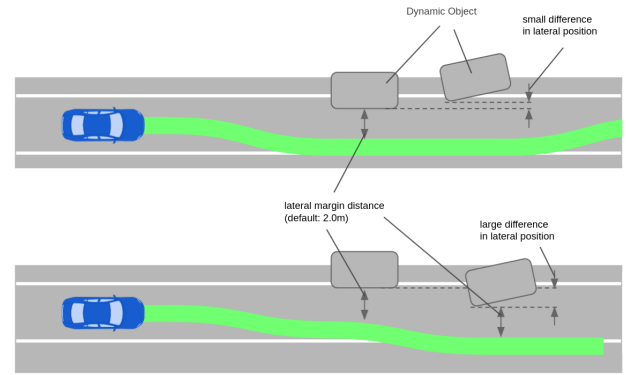
- 如何与障碍物进行复杂的交互与决策

车道决策与障碍物决策共同为下游的轨迹规划提供可行空间。



给定车道决策下，定性地决定如何处理一个障碍物[1]

# 高速 NOP 中的决策



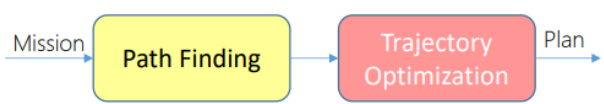

给定导航路线、车道决策和障碍物决策，定量地规划一条从车辆当前位置指向目的地的轨迹

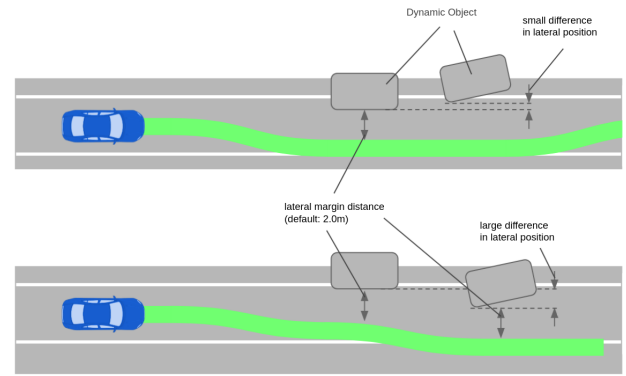
# 高速 NOP 中的规划

- 轨迹定义： 车辆状态的时间序列

$$t \rightarrow (x, y, \psi, \kappa, v, a)$$

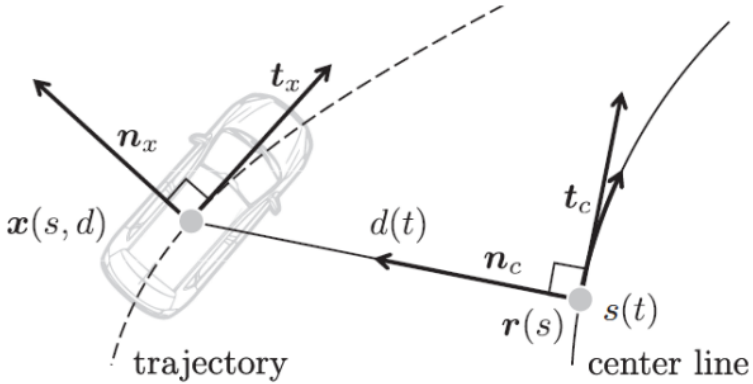
- 轨迹需要满足的性质：
  - 安全：不能与障碍物、路沿、护栏等发生碰撞
  - 舒适：加减速平滑
  - 遵守交规：红绿灯、道路限速
  - 效率：尽可能快的到达目的地
- PipeLine




给定导航路线、车道决策和障碍物决策，定量地规划一条从车辆当前位置指向目的地的轨迹

# 规划：Frenet 坐标

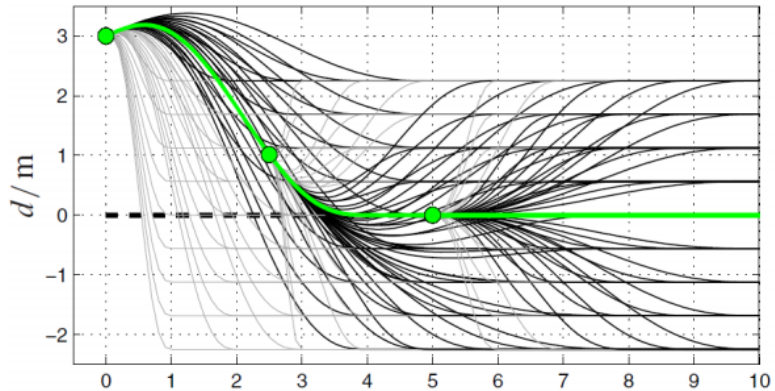



生成构型空间中满足目标和约束的连续可微曲线

$$[x, y, \theta, \kappa, v, a] \Leftrightarrow [s, \dot{s}, \ddot{s}, d, d', d'']$$

- dynamic reference frame.
- lateral and longitudinal independently.

# 轨迹生成




$$d(t) = a_{d0} + a_{d1}t + a_{d2}t^2 + a_{d3}t^3 + a_{d4}t^4 + a_{d5}t^5$$

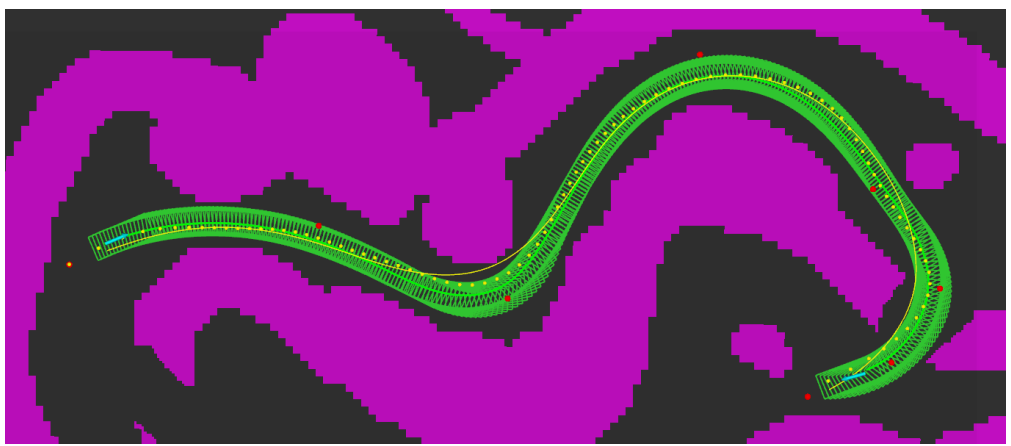
$$s(t) = a_{s0} + a_{s1}t + a_{s2}t^2 + a_{s3}t^3 + a_{s4}t^4 + a_{s5}t^5$$

- 采样
- 碰撞检测
- 轨迹评估

# 最小化 jerk 轨迹生成

**Why trajectory generation/optimization**

- Good for autonomous moving.
- Velocity/higher order dynamics can’t change immediately.
- The robot should not stop at turns.
- Save energy.



# 光滑轨迹跟踪：微分平坦

**常用算法**

- Boundary condition: start, goal positions (orientations)
- Intermediate condition: waypoint positions (orientations)
  - Waypoints can be found by path planning (A*, RRT*, etc)
- Smoothness criteria
- Generally translates into minimizing rate of change of “input”

The front-end is kinodynamic feasible, why a back-end necessary? Path quality and time efficiency.[1]

# 光滑轨迹跟踪：微分平坦

**微分平坦性的定义**

对于非线性系统系统： $\dot{x} = f(x, u)$

如果能存在下列形式的输出量：

$$z(t) = h(x, u, \dot{u}, ..., u^i)$$

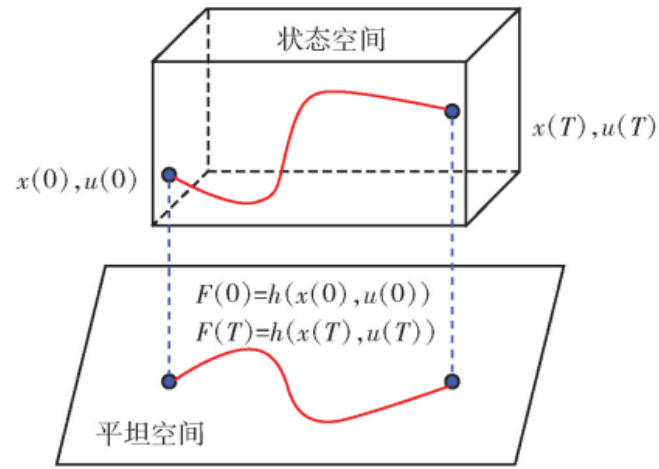
使得 $x, u$ 均可以表示为输出 $z$ 和其导数的函数，即

$$x(t) = x(z, \dot{z}, ..., z^i)$$

$$u(t) = u(z, \dot{z}, ..., z^i)$$

则称系统对于输出 $z$ 是微分平坦的

**如果一个非线性系统具有微分平坦性，则该非线性系统可控**




假设完成了轨迹优化，那么如何保证控制能准确实现优化的轨迹而不至于由于控制误差引发碰撞？[1]

# 微分平坦的车辆模型

自行车模型的方程如下：

$$
\begin{aligned}
\dot{x} &= v\cos\theta \\
\dot{y} &= v\sin\theta \\
\dot{\theta} &= v\tan\delta/L \\
\dot{\delta} &= \omega
\end{aligned}
$$

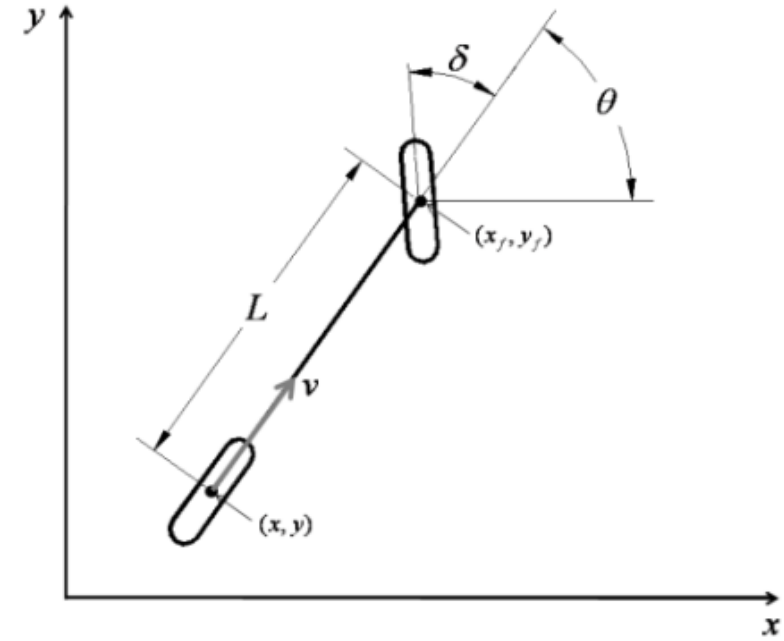
选取 $(x, y)$ 为微分平坦变量可以得到

$$\theta = \operatorname{atan}2(\dot{y}, \dot{x})$$

$$\delta = \arctan\frac{(\ddot{y}\dot{x}-\ddot{x}\dot{y})L}{(\dot{x}^2+\dot{y}^2)^{\frac{3}{2}}}$$

$$\omega = \frac{(\dddot{y}\dot{x}-\dddot{x}\dot{y})\left(\dot{x}^2+\dot{y}^2\right)-3(\ddot{y}\dot{x}-\ddot{x}\dot{y})(\dot{x}\ddot{x}+\dot{y}\ddot{y})}{(\dot{x}^2+\dot{y}^2)^3+L^2(\ddot{y}\dot{x}-\ddot{x}\dot{y})^2}\sqrt{\dot{x}^2+\dot{y}^2}L$$

**只要轨迹 $(x, y)$ 至少两阶微分存在**，则可以得到系统其他状态 $(v, w)$ 与控制输入 $(v, w)$，即系统可控

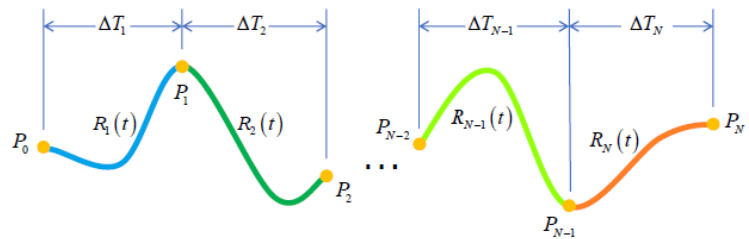


# 光滑轨迹生成

## 多项式轨迹

- Easy determination of smoothness criterion with polynomial orders
- Easy and closed form calculation of derivatives
- Decoupled trajectory generation in three dimensions

$$f(t) = \begin{cases} f_1(t) \doteq \sum_{i=0}^{N} p_{1,i} t^i & T_0 \le t \le T_1 \\ f_2(t) \doteq \sum_{i=0}^{N} p_{2,i} t^i & T_1 \le t \le T_2 \\ \vdots & \vdots \\ f_M(t) \doteq \sum_{i=0}^{N} p_{M,i} t^i & T_{M-1} \le t \le T_M \end{cases}$$

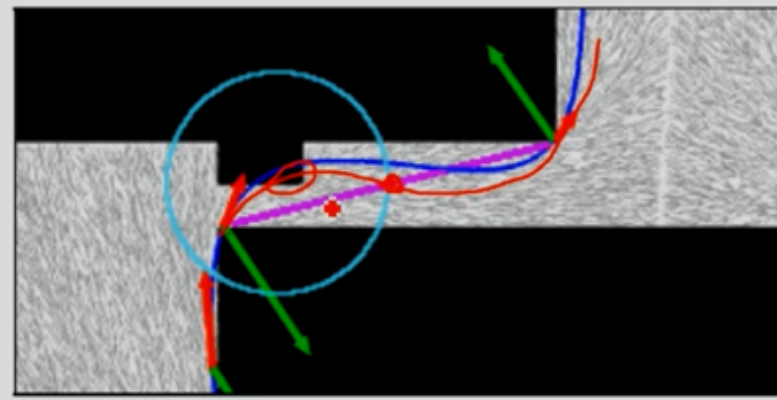



Minimum jerk -minimize angular velocity, good for visual tracking

# 解决碰撞：迭代方法

**Smooth is enough?**

- 初始轨迹无碰撞
- 轨迹优化后产生碰撞
- 通过（迭代地）添加中间碰撞点处的路径点来实现



**Better solution?**

- 迭代的点可能需要加很多次
- 无法保证有限次的加点可以解决这个问题
- 导致局部轨迹质量下降很快

Recall: 决策模块的输出：时间上的位置限制

**以决策的输出作为约束条件，最小化 jerk 为目标函数，构建一个优化问题**

# 基于优化的轨迹生成

**考虑曲率和避障的轨迹优化**

轨迹可以分为横纵向两个部分，分别表示为：

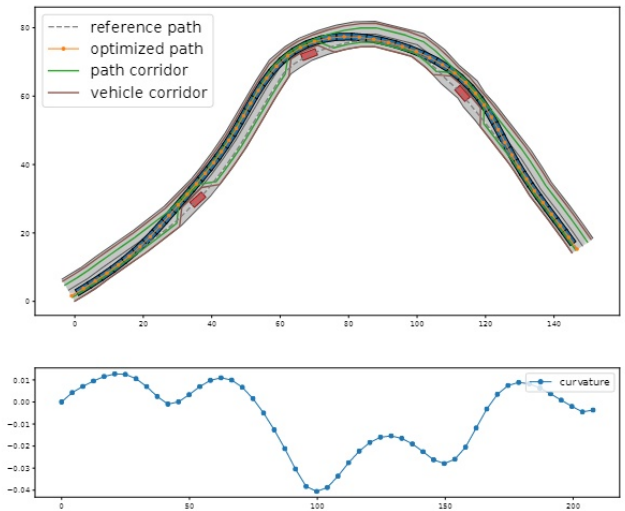
$$f_i(s) = a_{i0} + a_{i1}\left(s - s_i\right) + a_{i2}\left(s - s_i\right)^2 + a_{i3}\left(s - s_i\right)^3$$
$$g_i(s) = b_{i0} + b_{i1}\left(s - s_i\right) + b_{i2}\left(s - s_i\right)^2 + b_{i3}\left(s - s_i\right)^3$$

一个可取的目标函数为：

$$\min_{a_{ij}, b_{ij}} \sum_{i=0}^{N} w_1 \left( \ddot{f}_i\left(s_i\right)^2 + \ddot{g}_i\left(s_i\right)^2 \right)$$

约束有：

- 不能与障碍物发生碰撞
- 曲率连续(控制需求)
- 连续性条件(各段轨迹需要导数连续)




可以较好的解决多种复杂约束情况下的轨迹生成问题

# 量产自动驾驶中的决策规划挑战

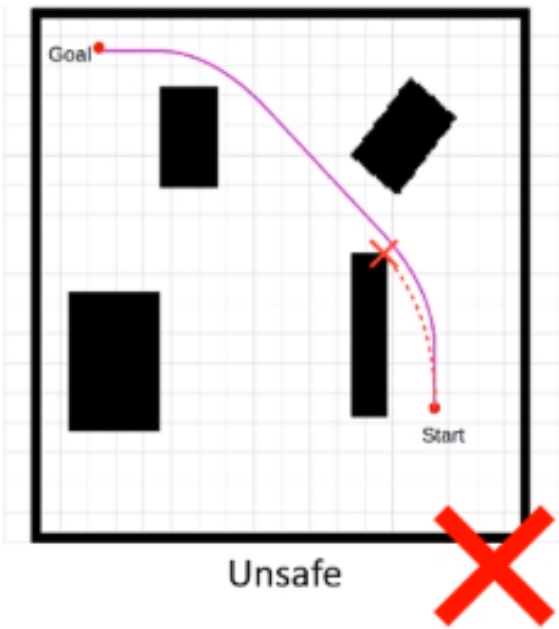
## 基于城区领航辅助驾驶(City-NOP)分析

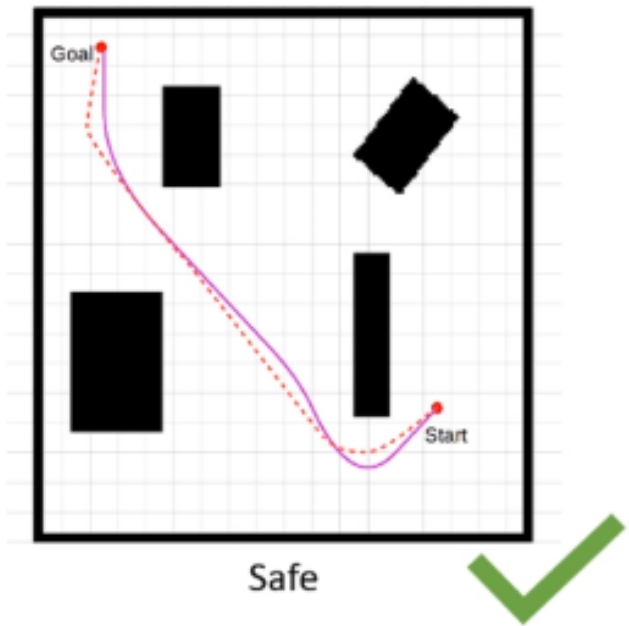
# City-NOP：Difficulties

- Interaction
- Uncertainty
  - with motion uncertatinty
  - with limited field-of-view(occulsion...)
  - with percetion uncertatinty

# Uncertatinty: motion uncertatinty

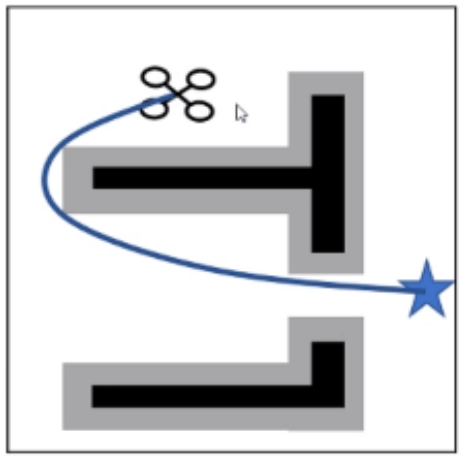
- Motivations:
  - High control authority is impractical
  - Real-world env factors: rain, snow, terrain and etc.



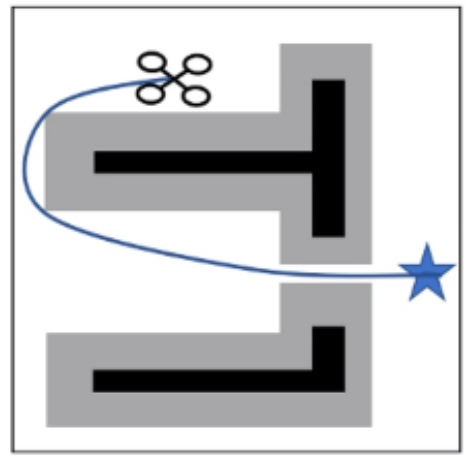



# Uncertatinty: motion uncertatinty

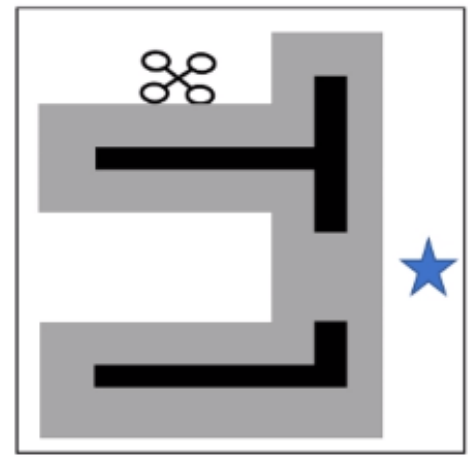
- Motivations:
  - Over-inflating obstacles is not a complete solution



Proper inflation



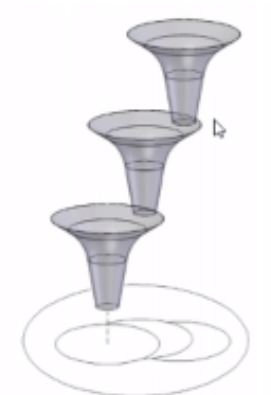
Okay over inflation



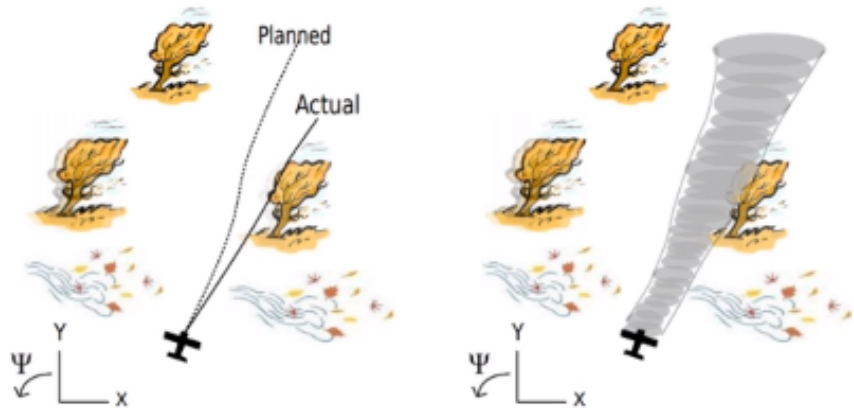
Bad over inflation

# Uncertatinty: motion uncertatinty

- Related work:
  - Reachability analysis



R. Tedrake, MIT, 2009



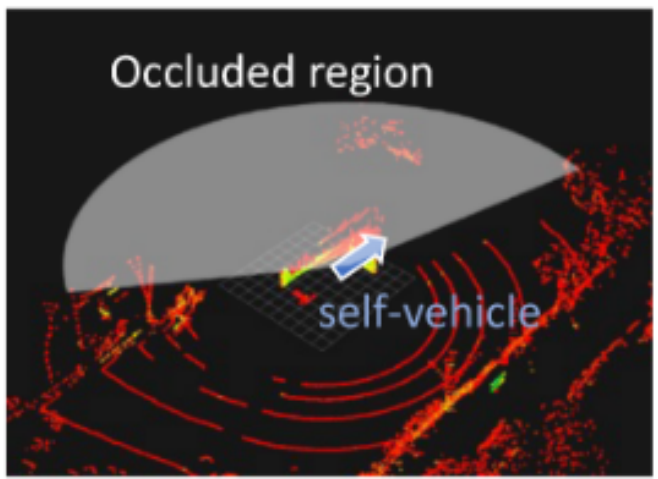

A. Majumdar and R. Tedrake, MIT, 2017

Drawbacks:

1. Computationally expensive
2. Too conservative

# Uncertatinty: limited FOV

- Challenges in real world navigation:
  - Map is unknown/not-fully reliable
  - Sensor has limited FOV
  - Occulsion usually exists in urban

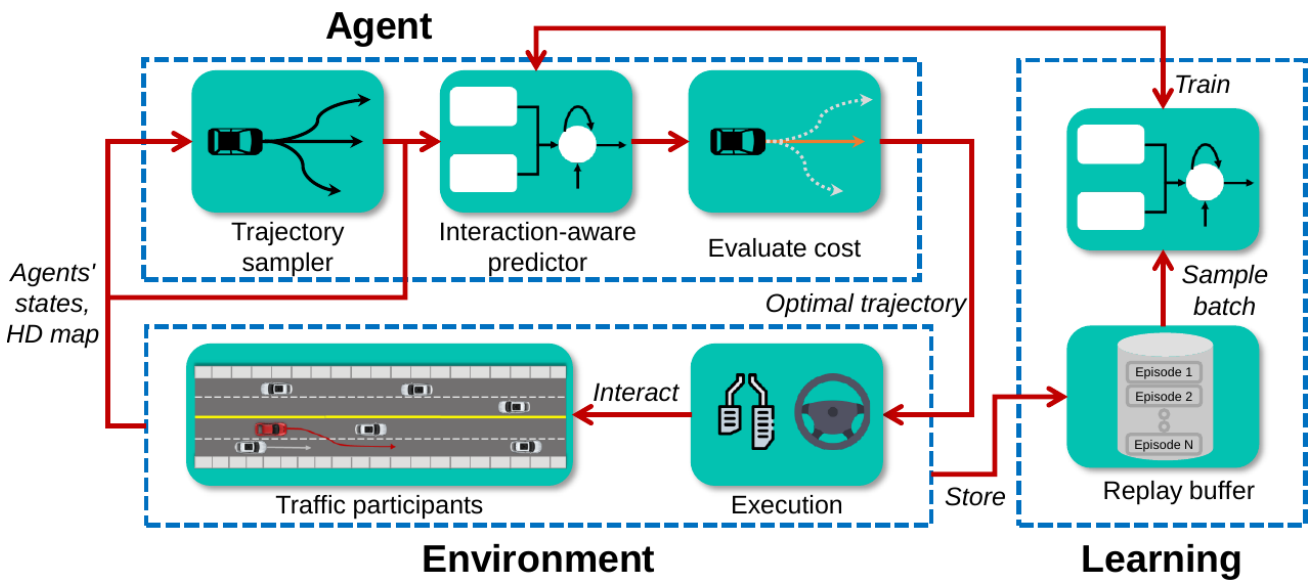
# Uncertatinty: perception

## Decision density

# Method: Learning-based Decision

**Prediction-coupled behavior planning**

- Embed the history states of agents and scene context into high-dimensional spaces, encode the interactions between agents and the scene context using Transformer modules.
- Employ a learned optimizer as a motion planner to explicitly plan a future behavior for the AV according to the most-likely prediction result and initial motion plan.



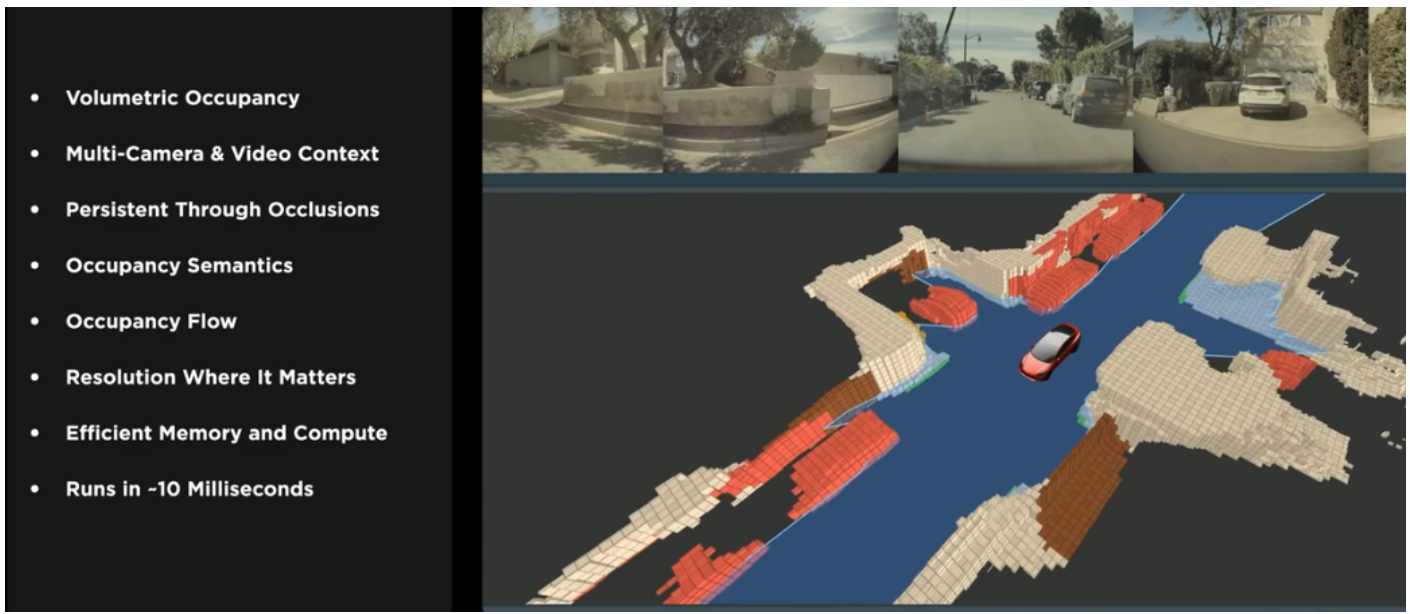
# Method: Learning-based Decision

add more

# Method: Occupancy-based Planning

**A low-cost urban solution**

- From **Free Space** to **Occupancy map**
- Occupancy Grid/Flow/Occusion
- Not object-based, suitable for motion planning



# Occupancy-based Planning

**Features**

- 3D representation
- General object detection
- long-tail case(not in trainning set)
- easy to use in prediction/planning pipeline
- uncertatinty prediction(probality)
- dynamic features

**Occupancy map based Planning method**

- A*
- State Lattice
- RRT
- nearly all planning methods, actually ...