量产自动驾驶中的决策规划

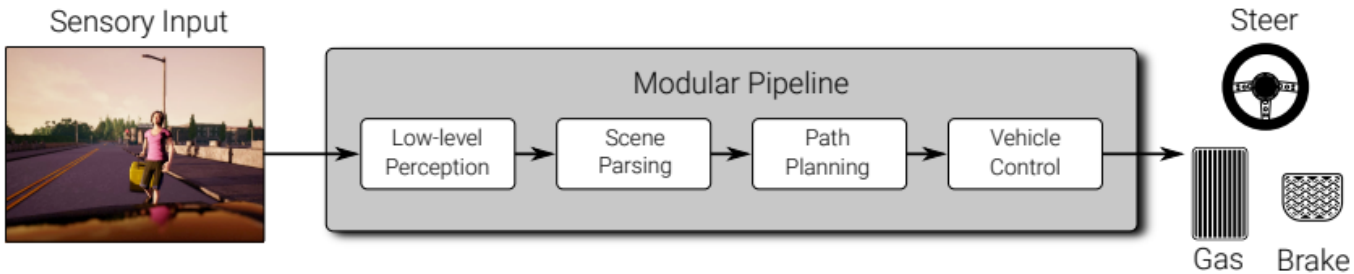
从领航高速(NOP)到领航城区(City-NOP)

肖飞宇, 3/1/2023

引言

自动驾驶中的决策规划

Planning and Decision Making



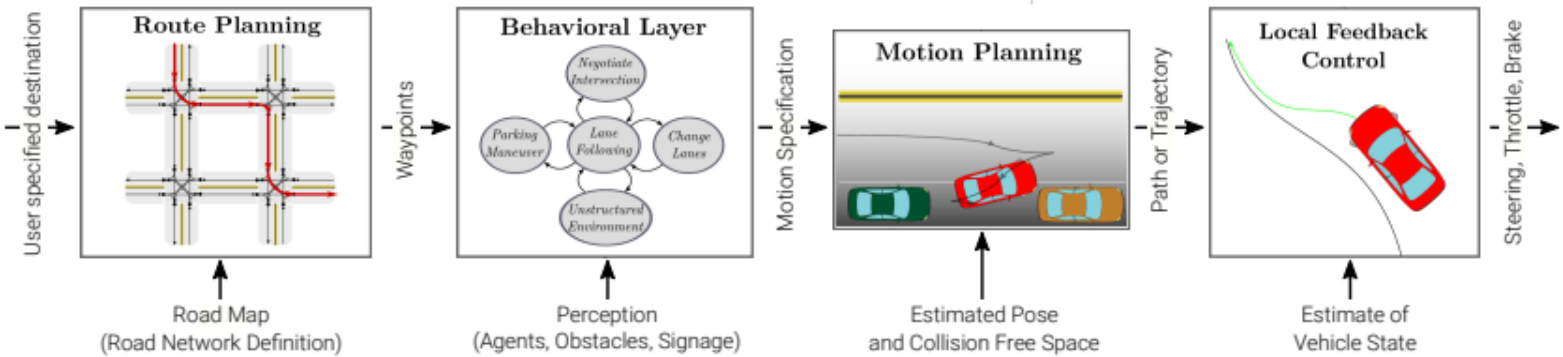
Problem Definition

- 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

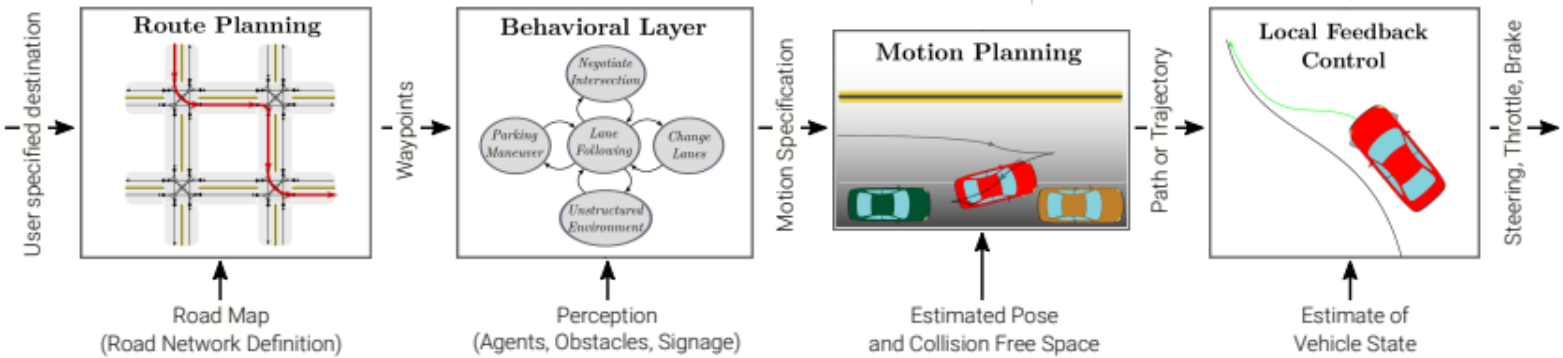
Planning and Decision Making



- 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

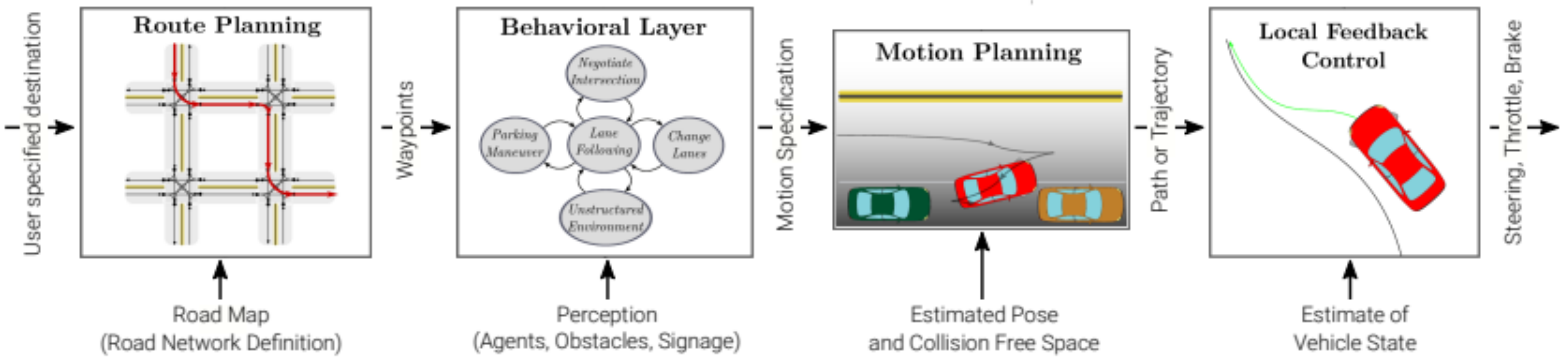
[1]Paden et al.: A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles. TIV, 2016.

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

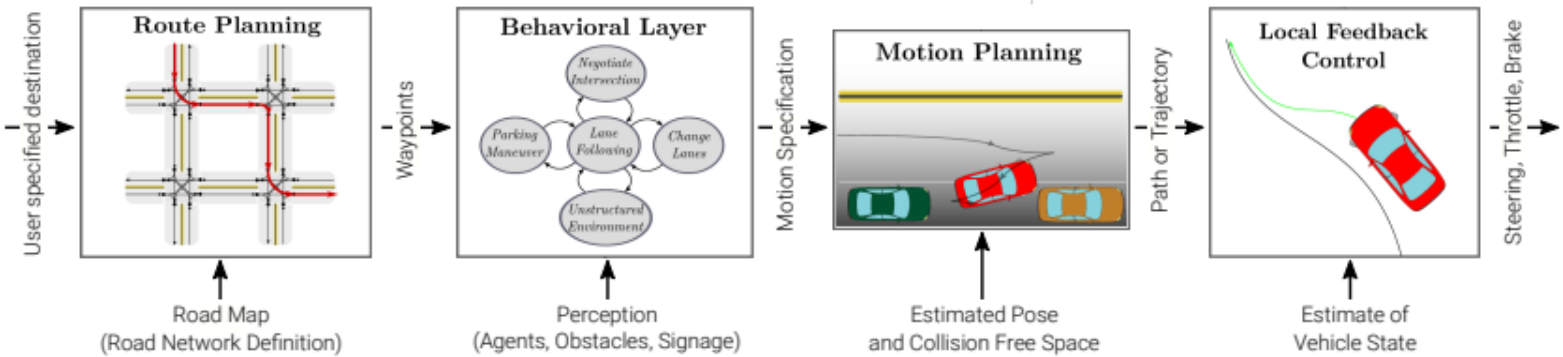


- Represent **road network** as **directed graph**.
- Edge weights correspond to road segment length or travel time
- Problem translates into a minimum-cost graph network problem
- Inference algorithms: Dijkstra, A^*

Solved for industrial applications like Map Apps.

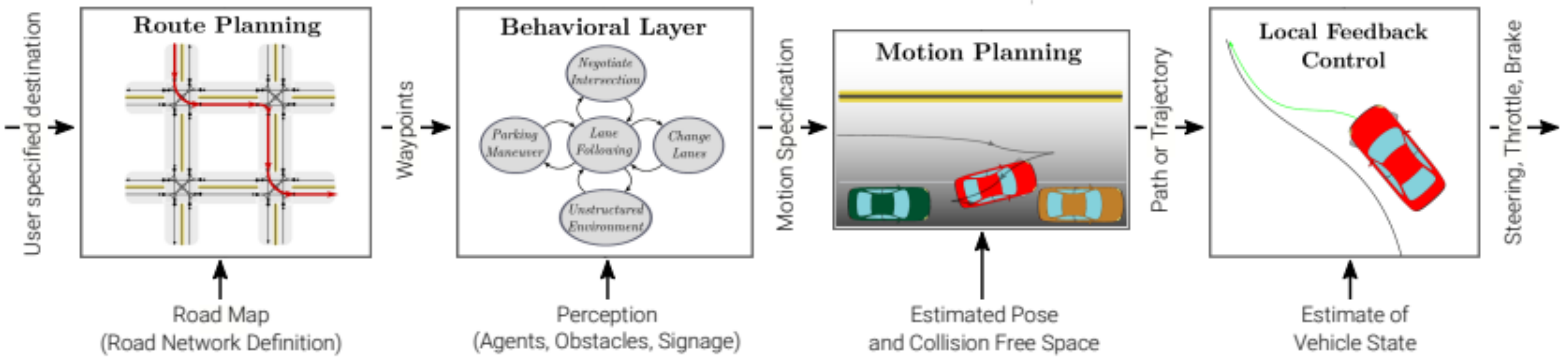
[1]Delling, Daniel et al. “Customizable Route Planning.” The Sea (2011).

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)

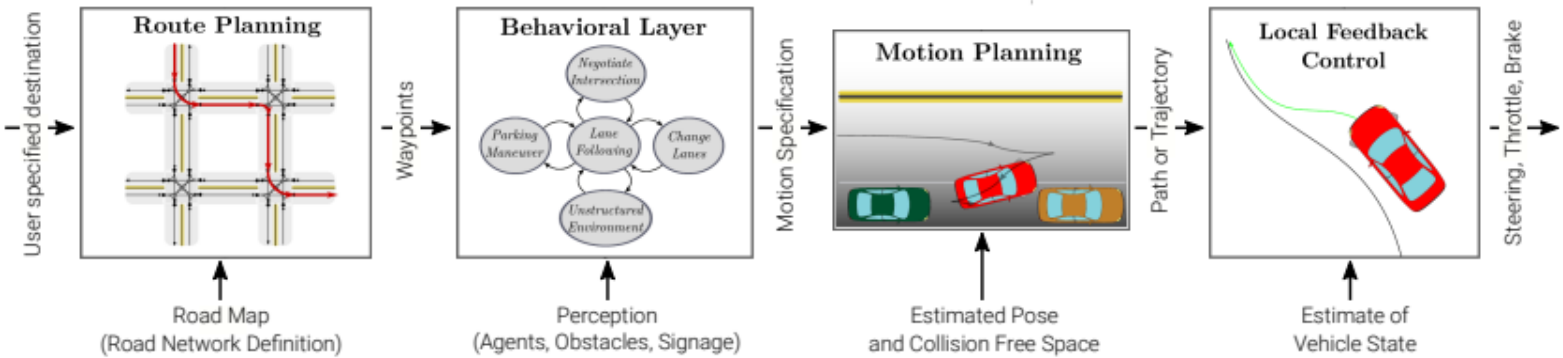
Step 3: Motion Planning



- 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

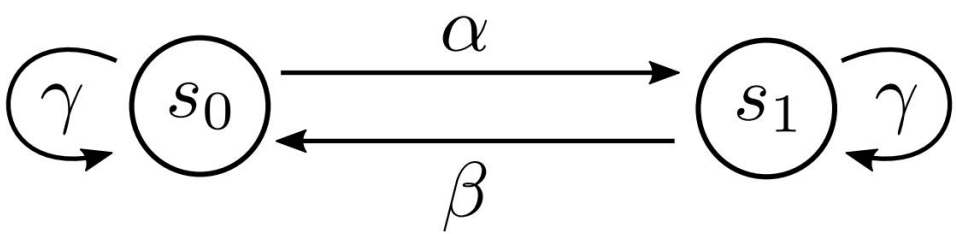
- To follow a planned route, the vehicle must conduct **various maneuvers**
- Examples include: speed tracking, car following, stopping, merging, etc.
- It is difficult to design a motion planner for all maneuvers jointly
- The **behavior planning** stage thus **discretizes the behaviors** into simpler (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

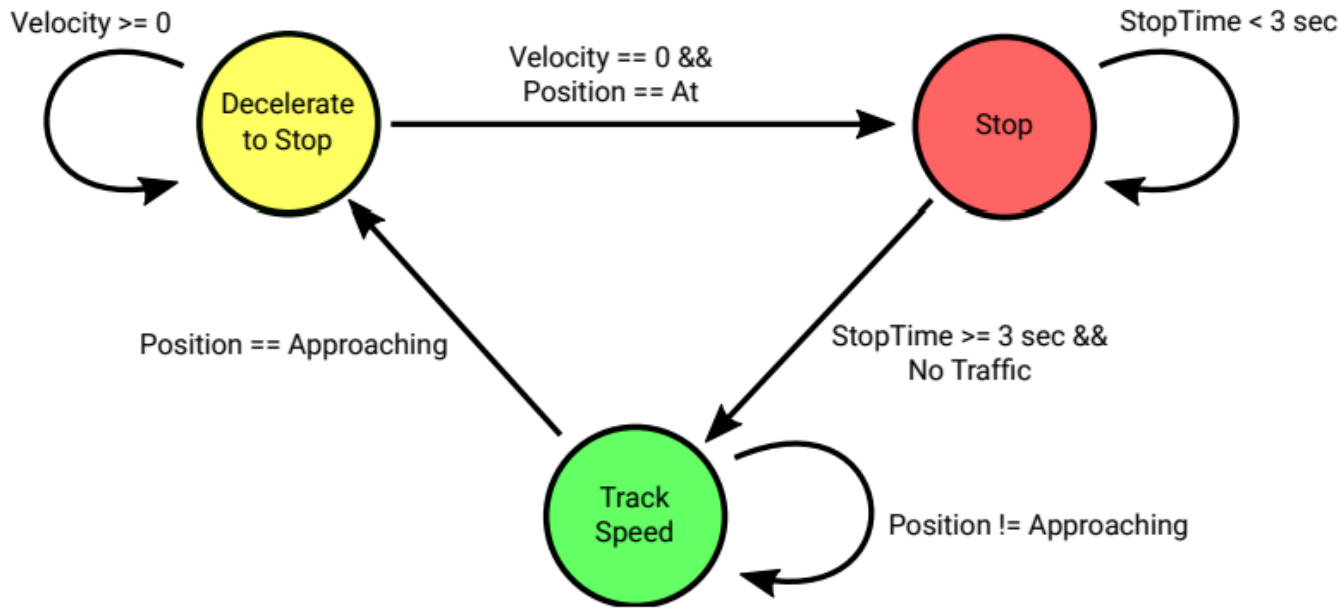
A Finite State Machine (FSM) is defined by quintuple $(\sum, \mathrm{S}, \mathrm{F}, s_0, \delta)$

- $\sum$ is the input alphabet
- 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

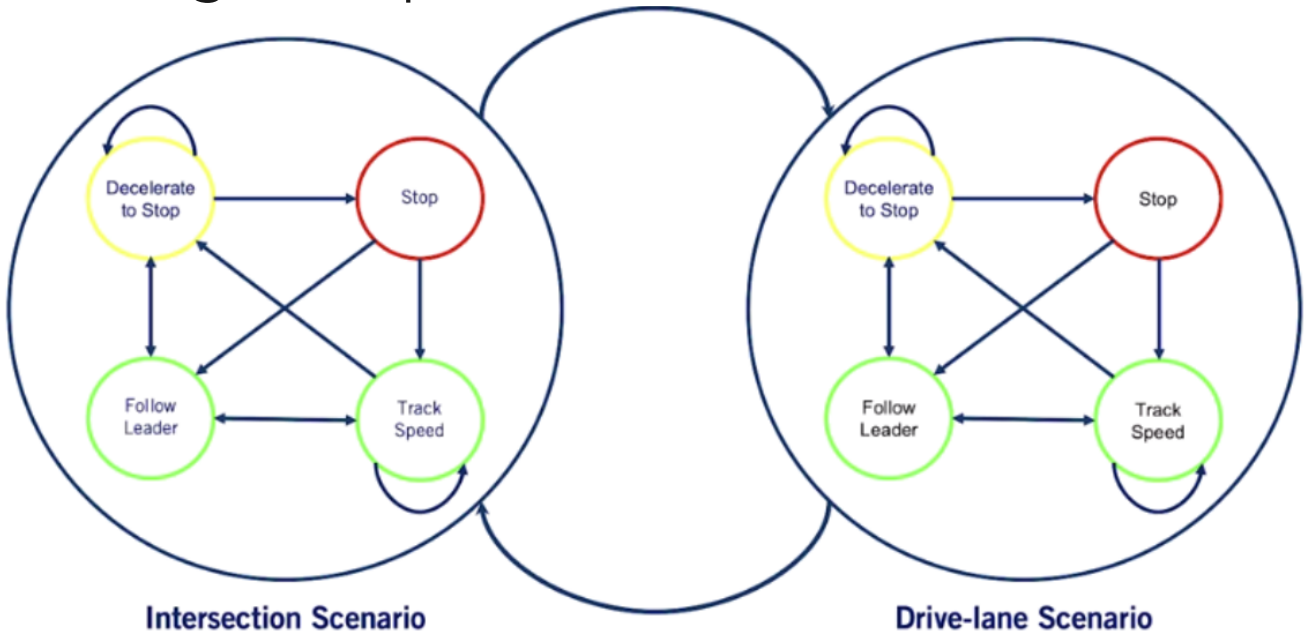
Example:



FSM for a Simple Vehicle Behavior

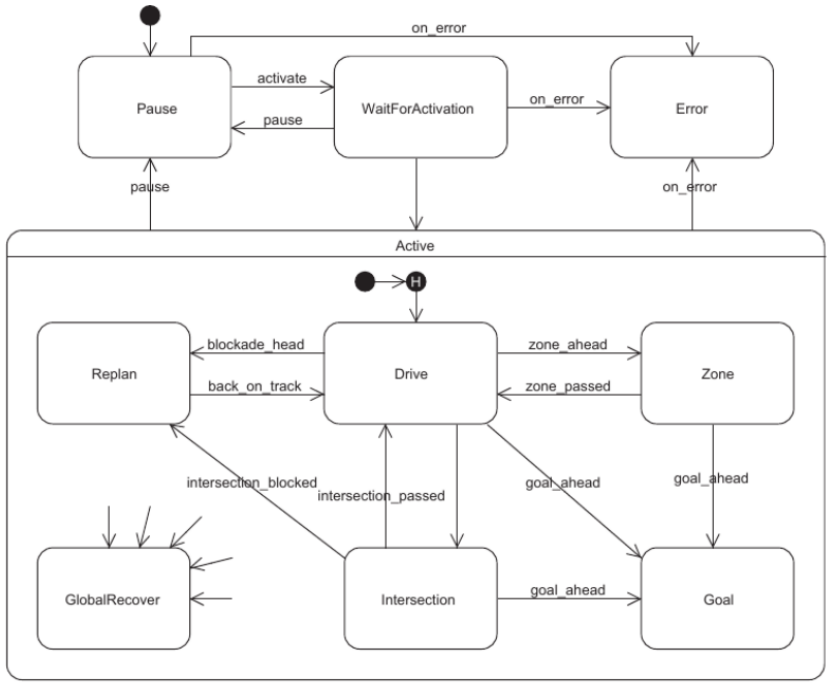


Handling Multiple Scenarios

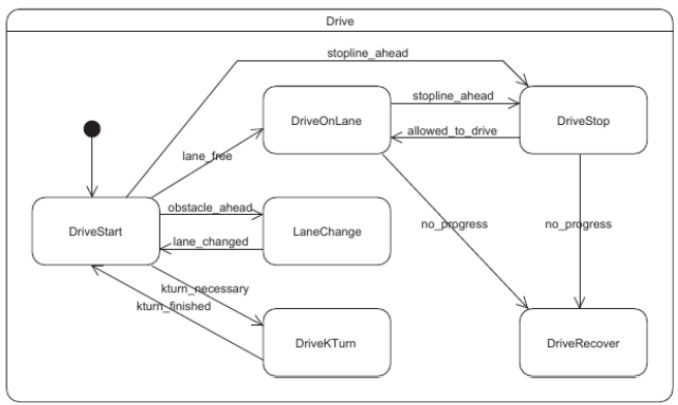


- Hierarchical State Machine (HSM)
- Advantages: Simpler, more efficient – Disadvantages: Rule duplication

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:

- Variational Methods
- Graph Search Methods
- Incremental Search Techniques

Variational Methods

Variational methods minimize a functional:

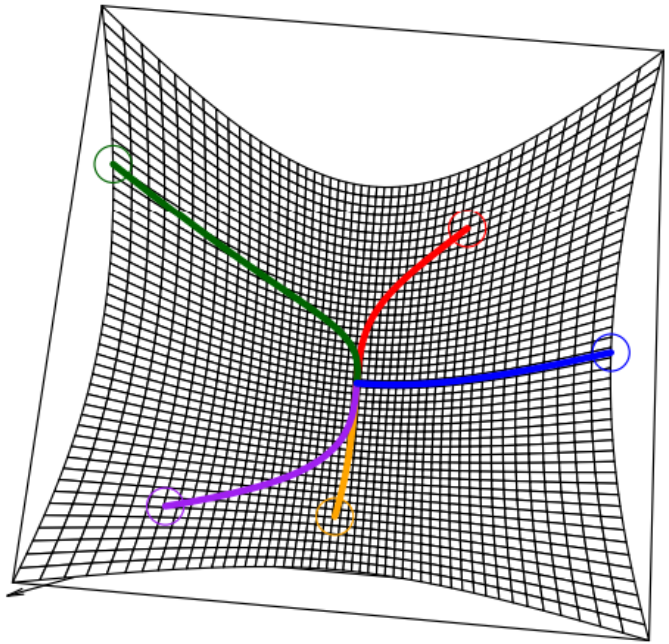
$$\underset{\pi}{\operatorname{argmin}} J(\pi) = \int_0^T f(\pi) dt$$
$$\text{s.t. } \pi(0) = \mathbf{x}_{\text{init}} \ \wedge \pi(T) \in \mathbf{X}_{\text{goal}}$$

- The functional integrates soft constraints (spatial, velocity, jerk, etc.)
- Additional hard constraints can be formulated (minimum turn radius, etc.)
- Solved using numerical optimization
- Often nonconvex problem $\rightarrow$ **converges slowly to local optimal or even not converge**

Note: that's the reason why we need good hierarchy, namely, if we have better decision inputs to formulate the state space into sub-convex space, the problem can be numerically solved.

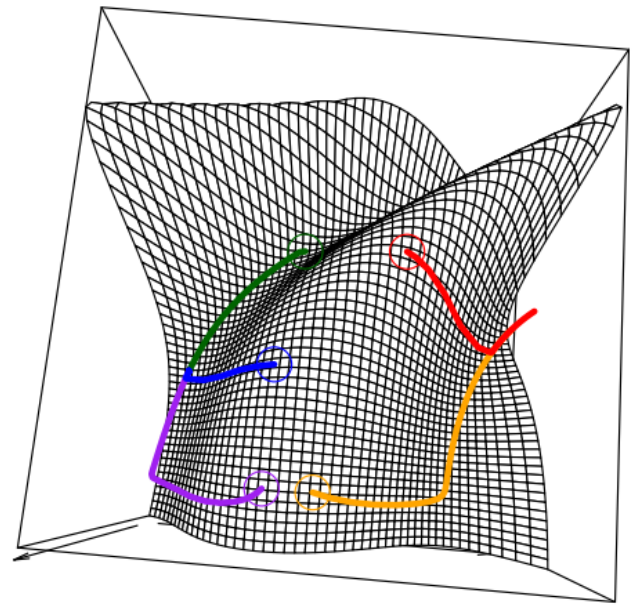
Convex

Convex cases can reach global minimum.

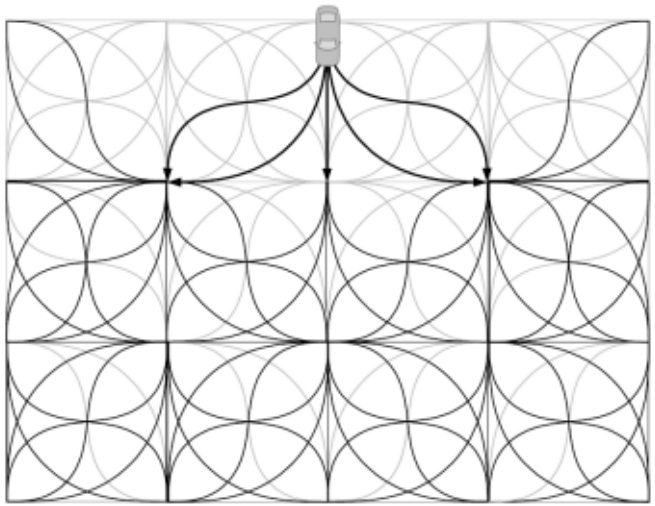


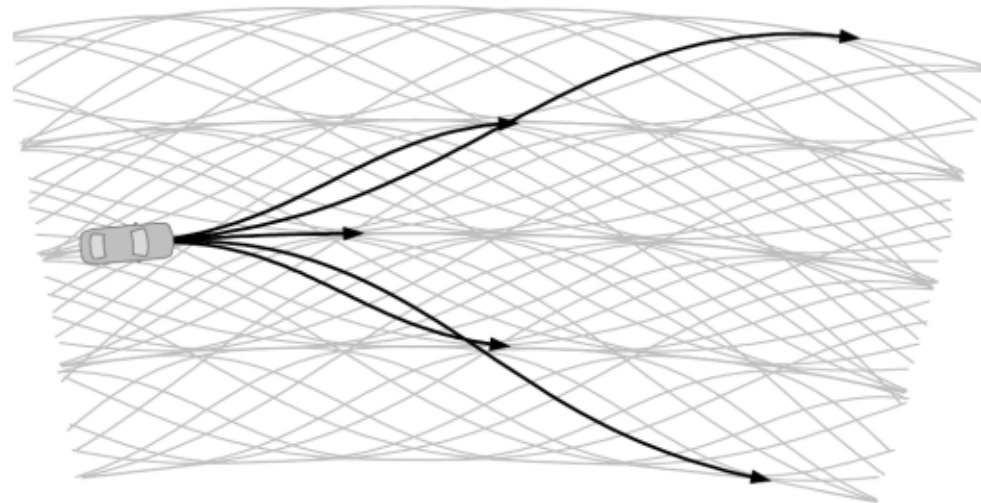
Non Convex

Nonconvex cases only fall into local minimum.



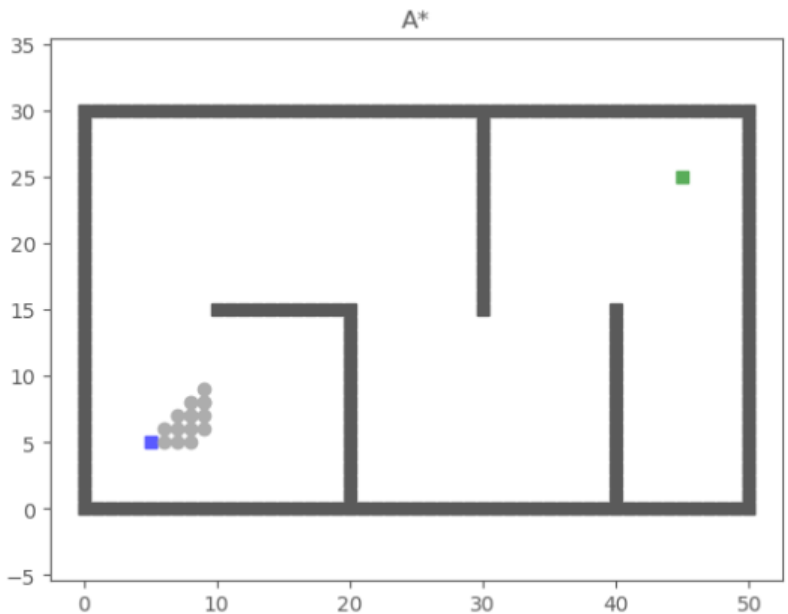
Graph Search Methods

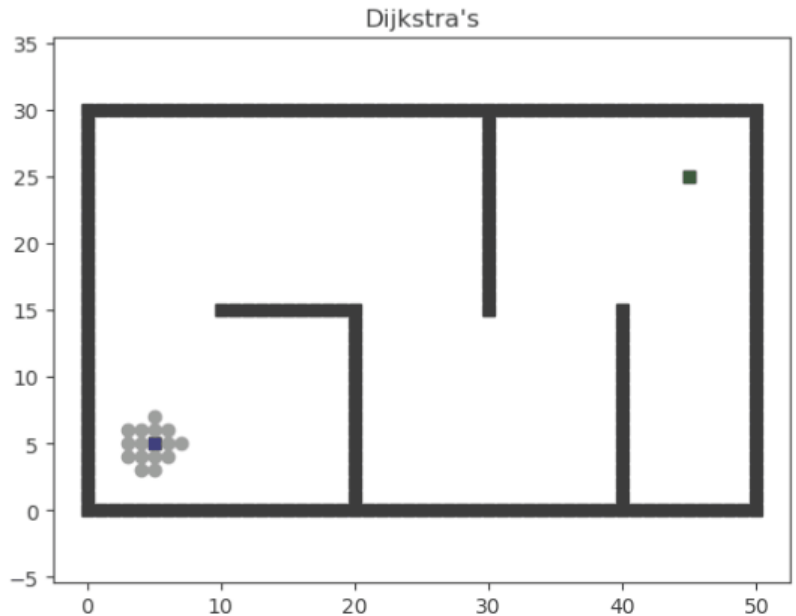




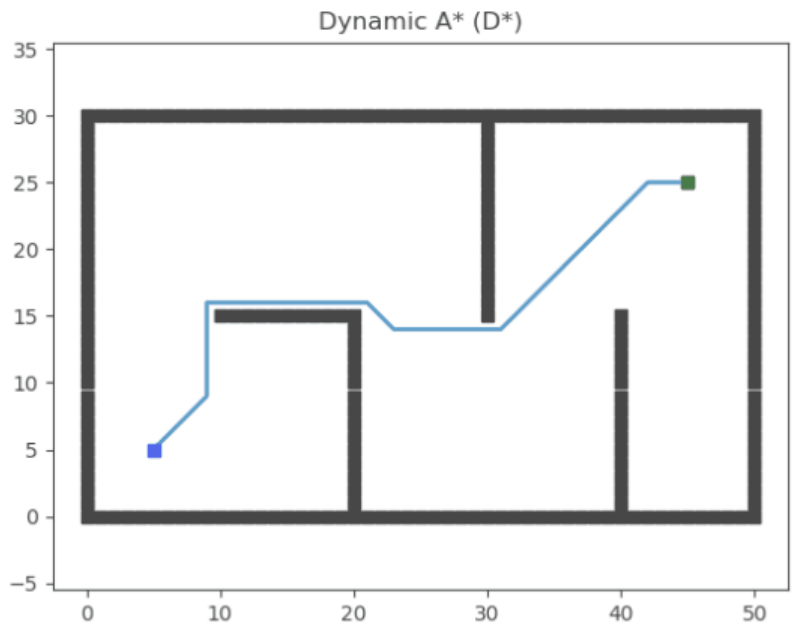
- Idea: Discretize configuration space into graph
- Various algorithms for constructing graphs
- Search strategies: Dijkstra, A^*, ... (like *route planning*)

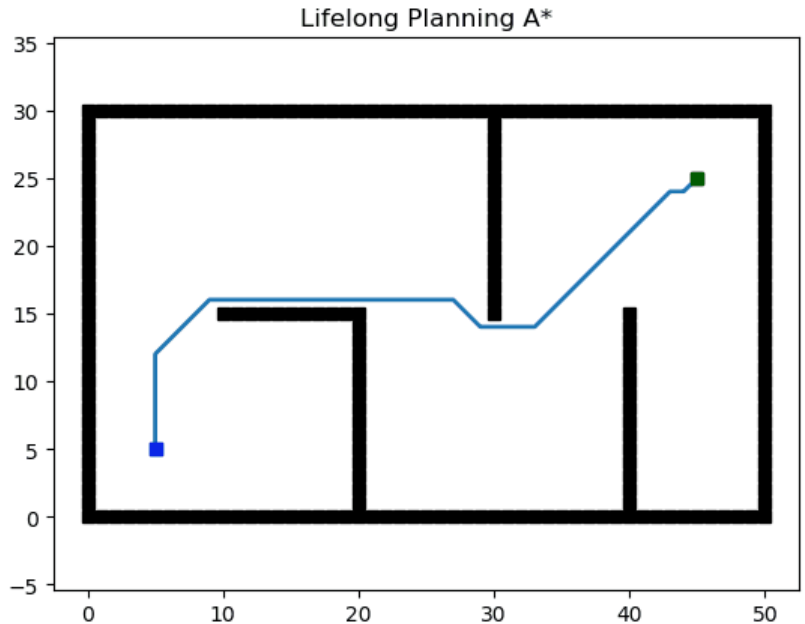
Graph Search Methods： Examples





Graph Search Methods： Examples

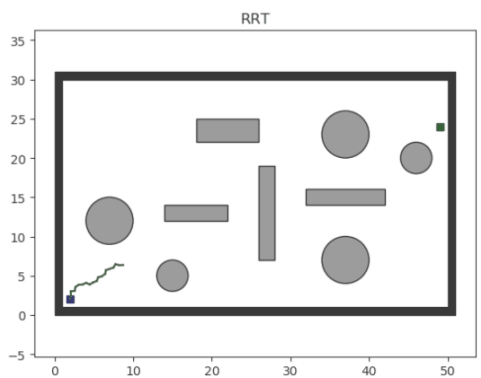


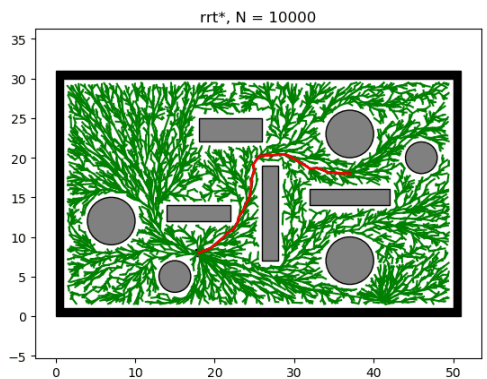


Incremental Search Techniques

- Idea: Incrementally build increasingly finer discretization of configuration space
- Guaranteed to provide feasible path given enough computation time
- But: computation time can be unbounded
- Prominent example: Rapidly exploring random trees (RRTs)

Note: Not complete and usually suitable for vehicle motion planning.



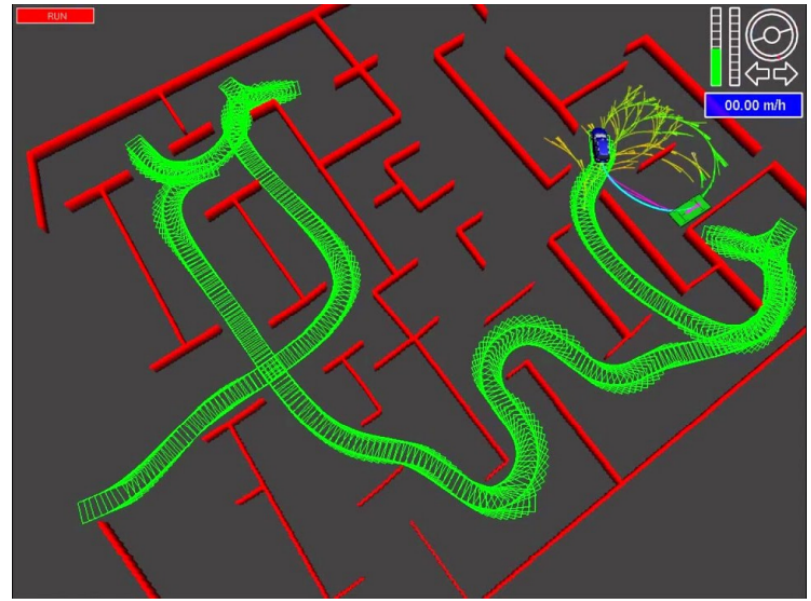


Hybrid A^*

- 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

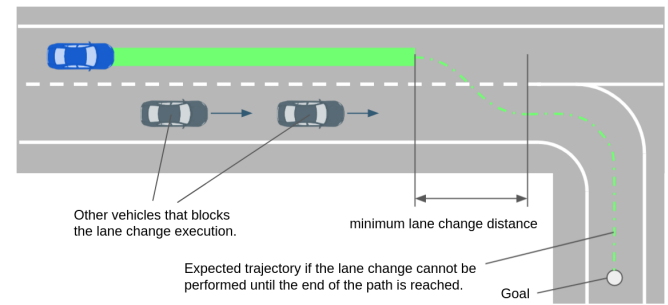
高速 NOP 中的决策

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 ...

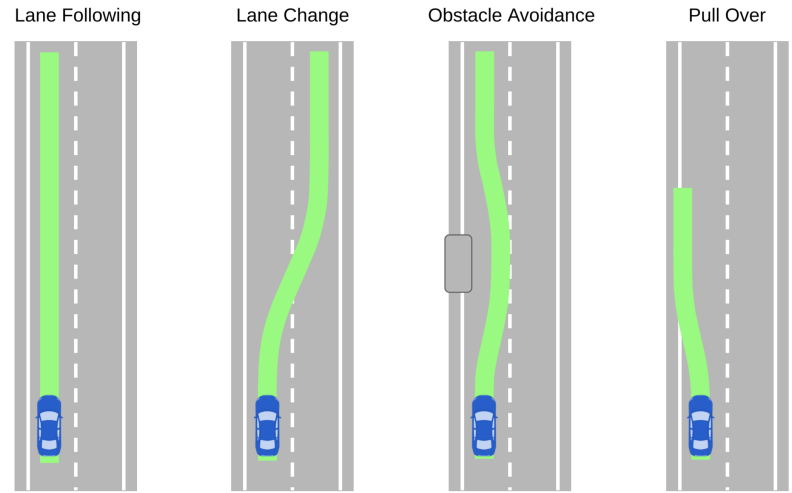


决策模块相当于无人驾驶系统的大脑，保障无人车的行车安全，同时也要理解和遵守交通规则。 为了实现这样的功能，决策模块为下游的规划模块提供了各种的限制信息

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

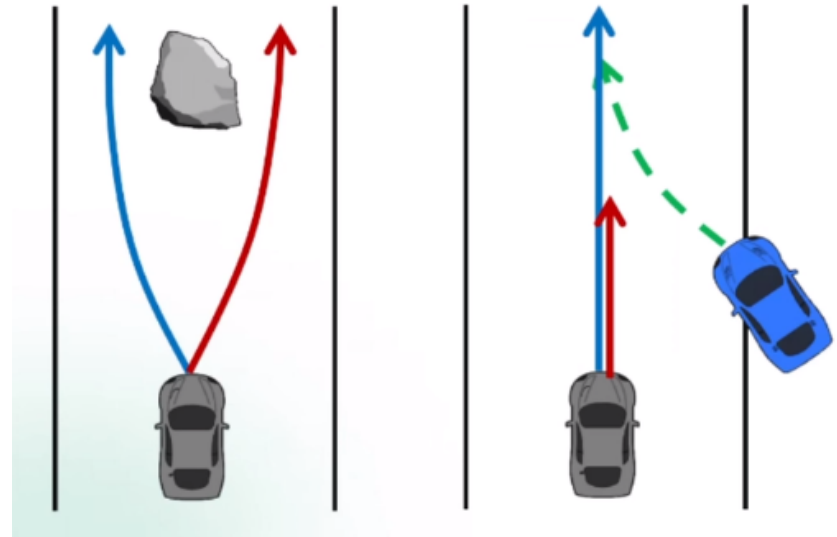
常用算法

- 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。



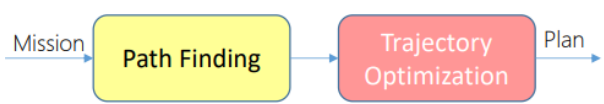
给定车道决策下，定性地决定如何处理一个障碍物[1]

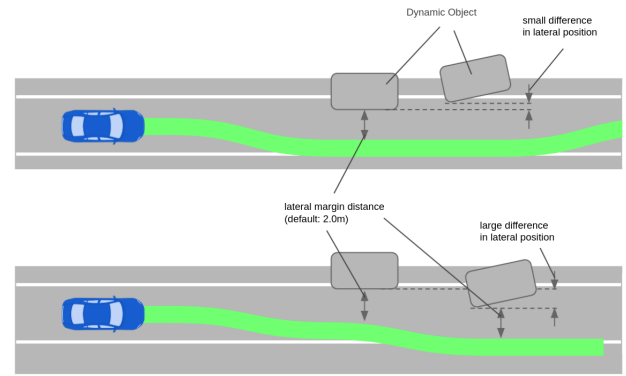
高速 NOP 中的规划

- 轨迹定义： 车辆状态的时间序列

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

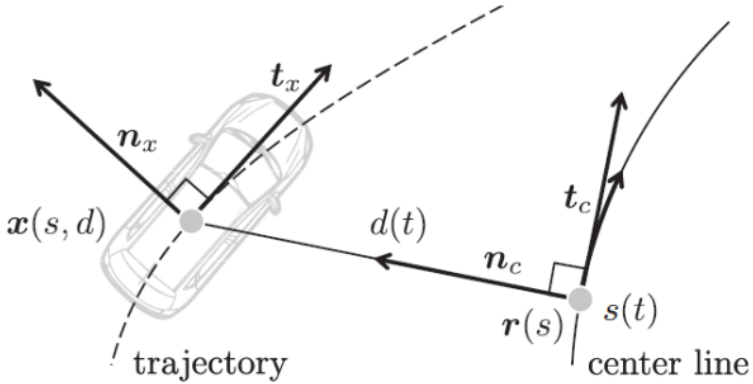
- 轨迹需要满足的性质：
  - 安全：不能与障碍物、路沿、护栏等发生碰撞
  - 舒适：加减速平滑
  - 遵守交规：红绿灯、道路限速
  - 效率：尽可能快的到达目的地
- PipeLine





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

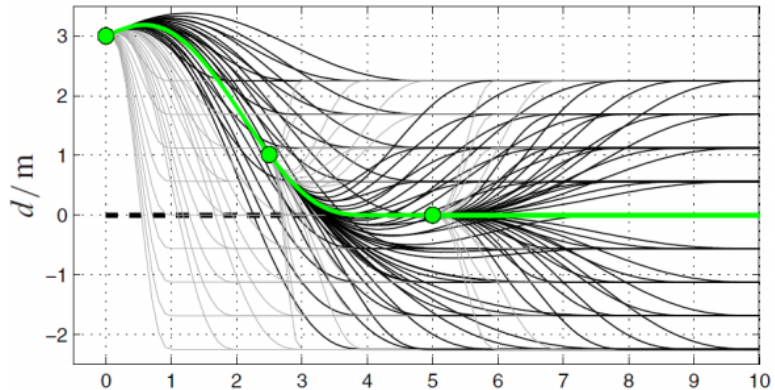


生成构型空间中满足目标和约束的连续可微曲线

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

- dynamic reference frame.
- lateral and longitudinal independently.

轨迹生成



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

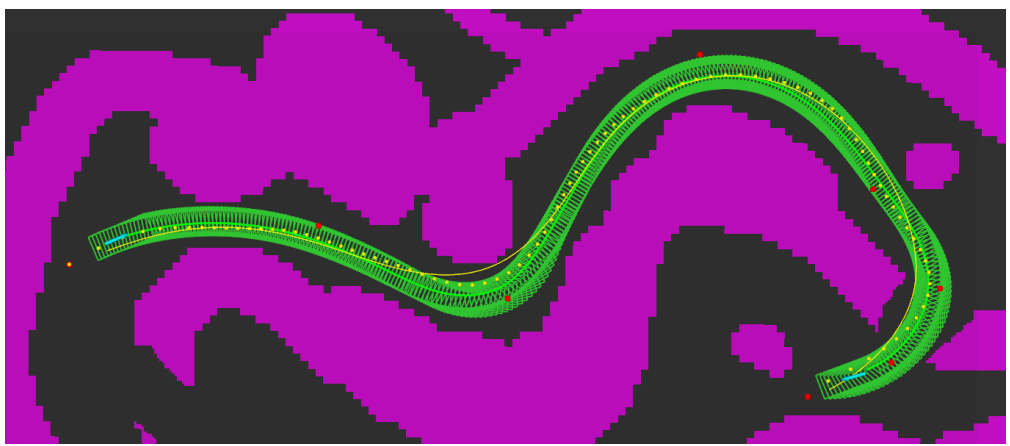
$$s(t) = a_{s0} + a_{s1}t + a_{s2}t^2 + a_{s3}t^3 + a_{s4}t^4 + a_{s5}t^5$$

- 采样
- 碰撞检测
- 轨迹评估

Minimum jerk trajectory generation

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.



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