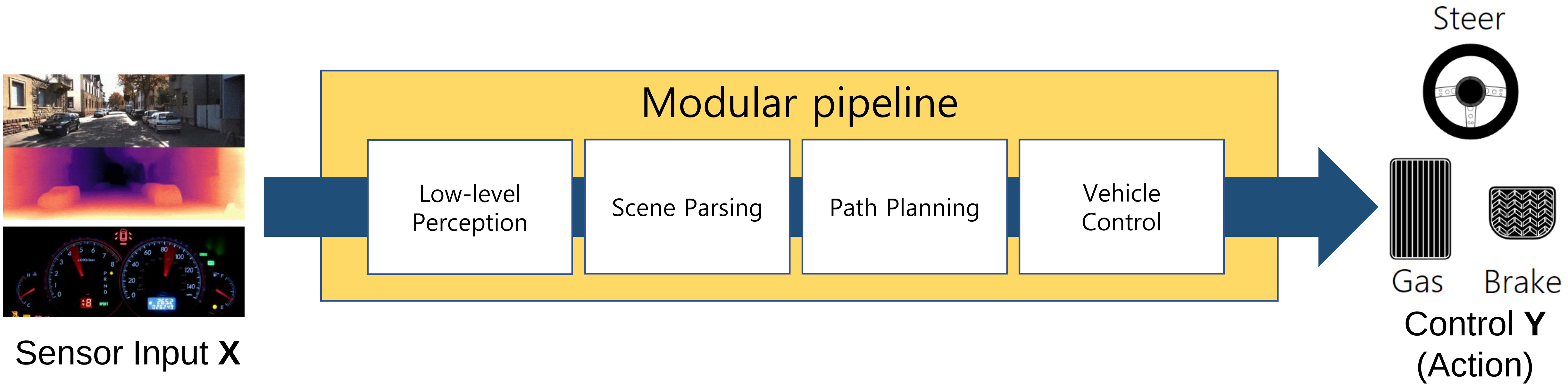
Advanced Programming Practice

Autonomous Driving
-Path Planning-

2022 Fall

Sogang University

Modular Pipeline



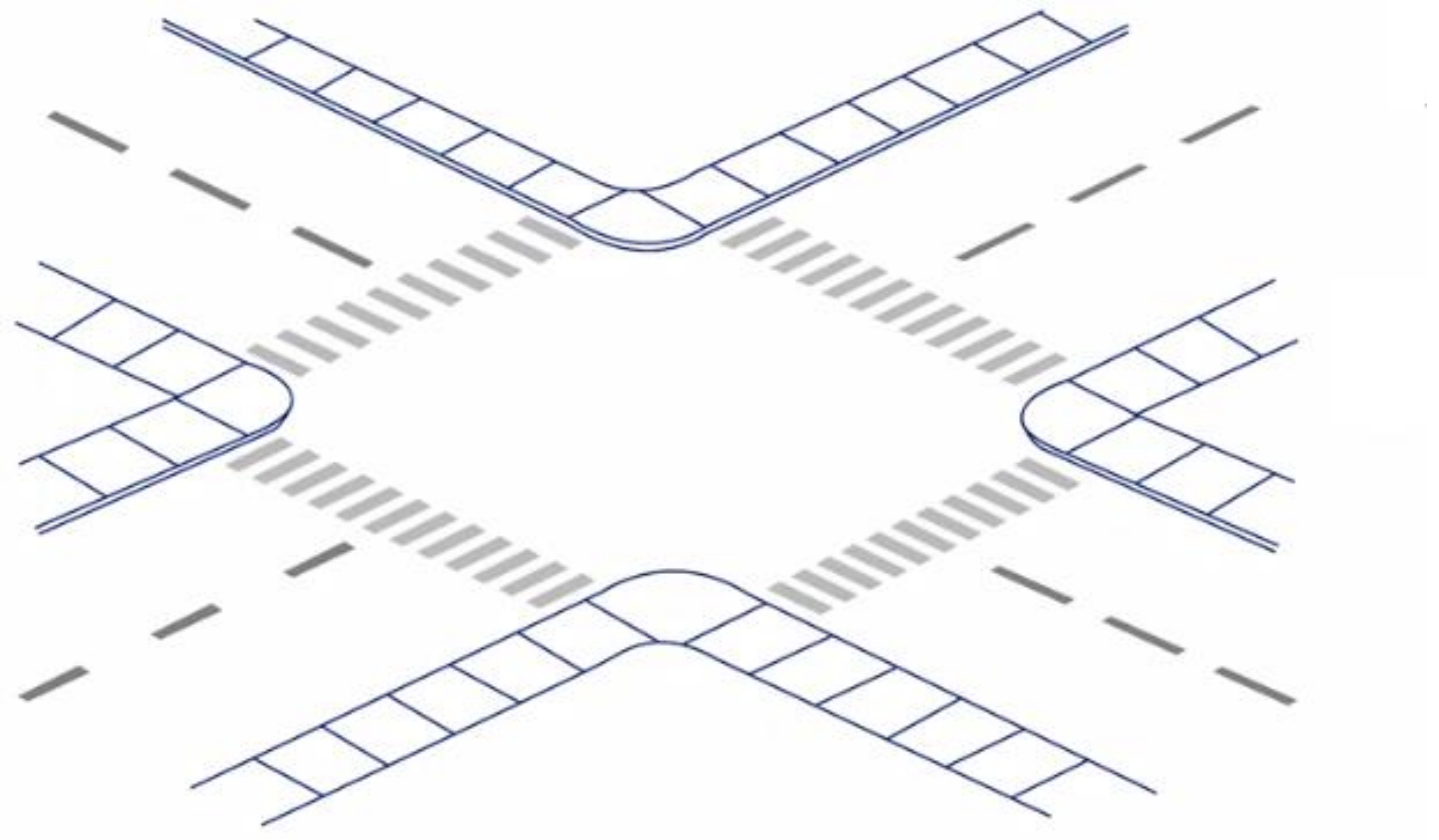
- Low-level Perception & Scene Parsing: Lecture 1
- **Path training: Lecture 2**
- Vehicle Control: Lecture 3

Planning and Decision Making

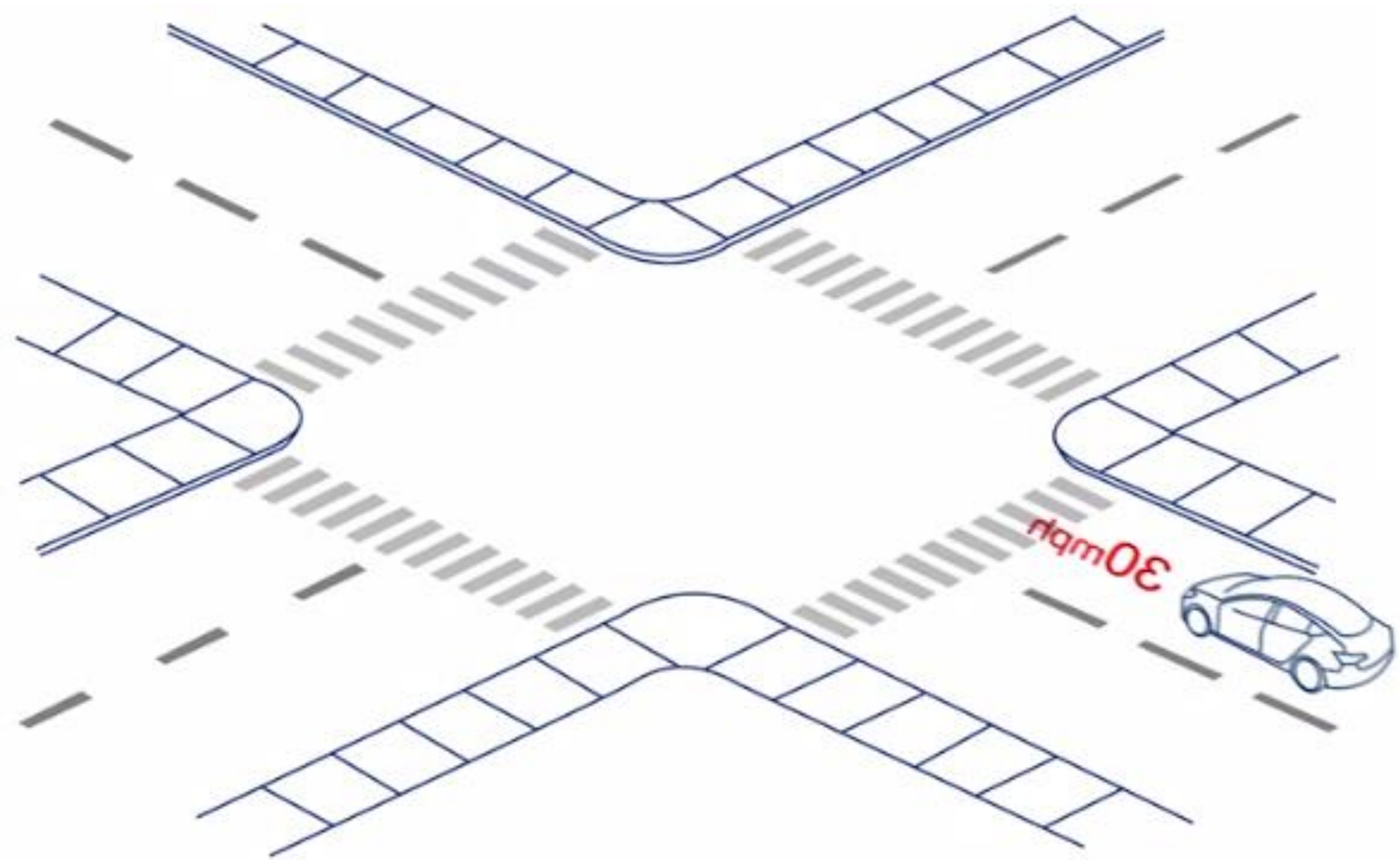
- **Problem definition:**
  - ➢Goal: Find and follow a path for here to destination
    - ➢Need to think static infrastructure and dynamic objects
  - ➢Input: vehicle and sensed surrounding environment state
  - ➢Output: path or trajectory being parsed to a vehicle controller

- **Challenges:**
  - ➢Driving situations and behaviors are very complex
  - ➢Thus difficult to model as a single optimization problem

Planning and Decision Making

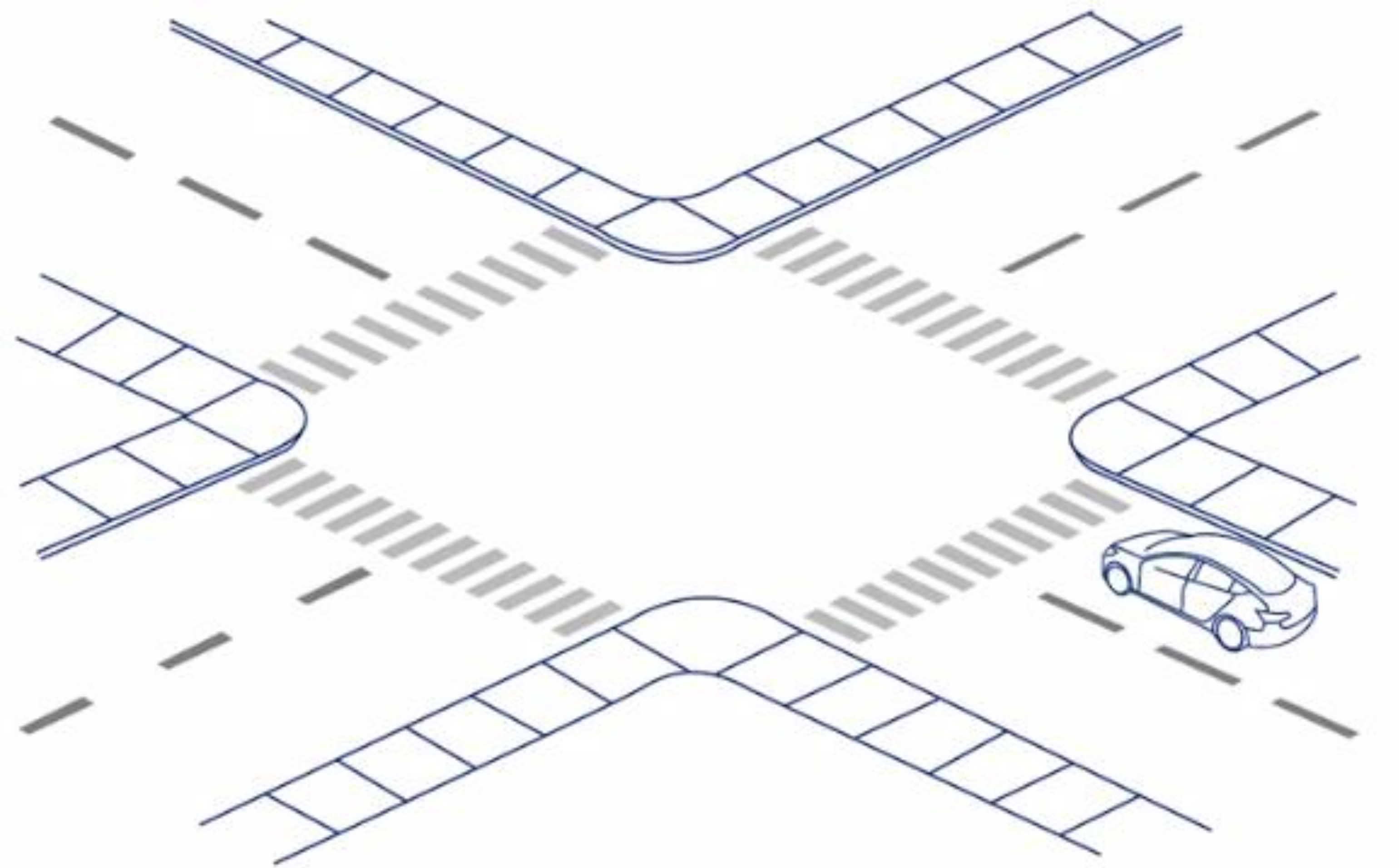


Planning and Decision Making



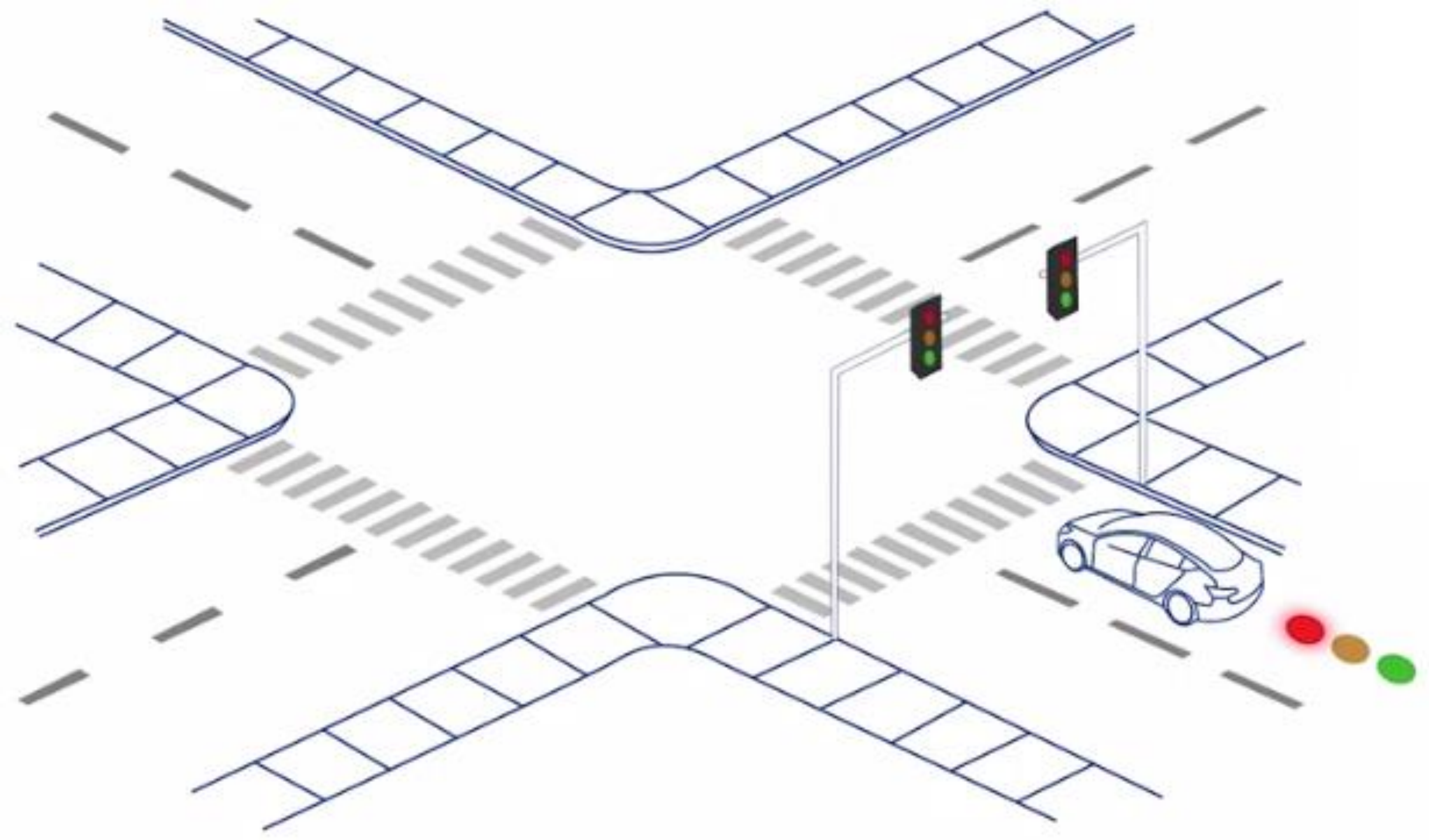
Speed tracking (30 mph)

Planning and Decision Making



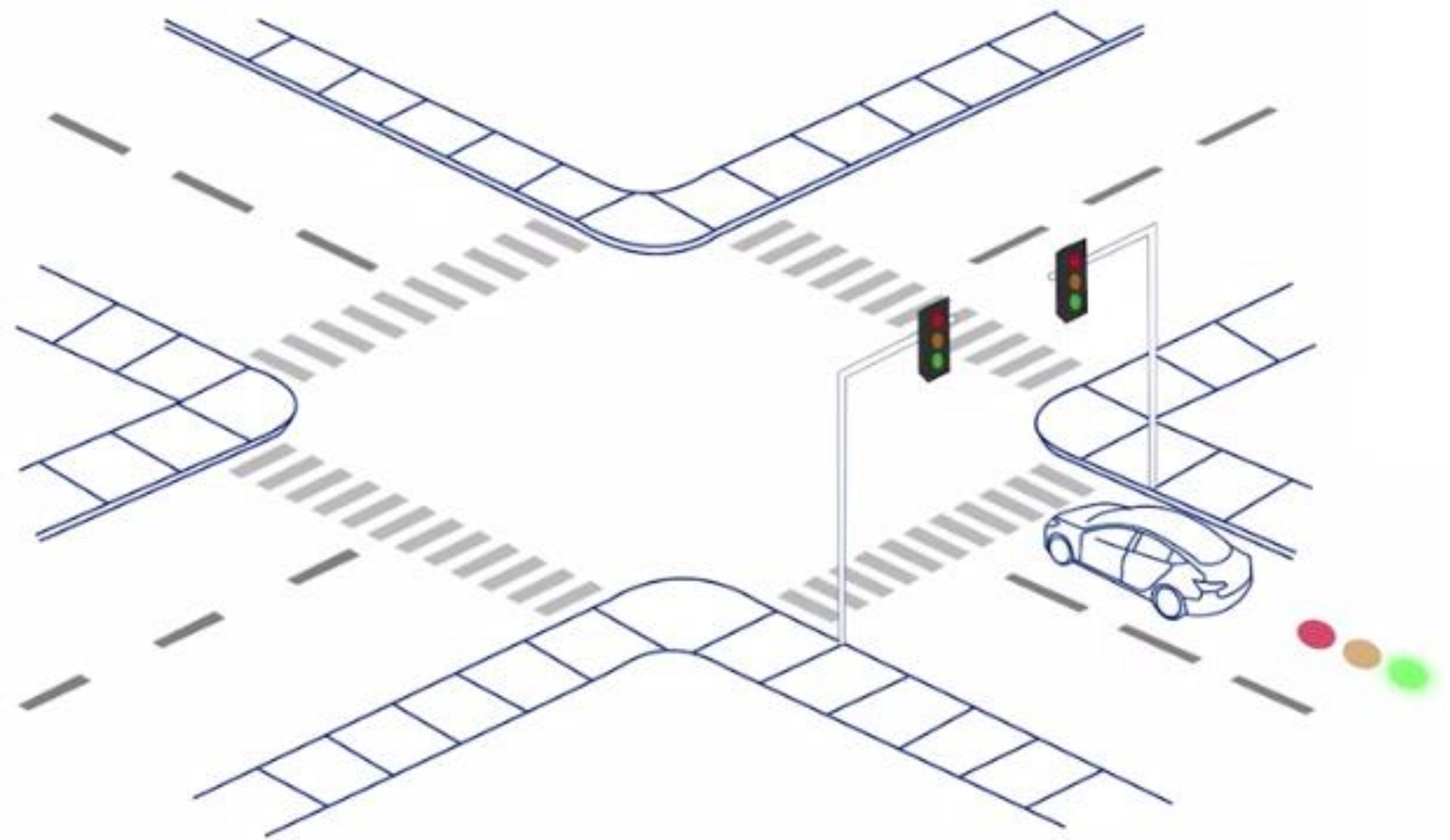
Decelerate to stop

Planning and Decision Making



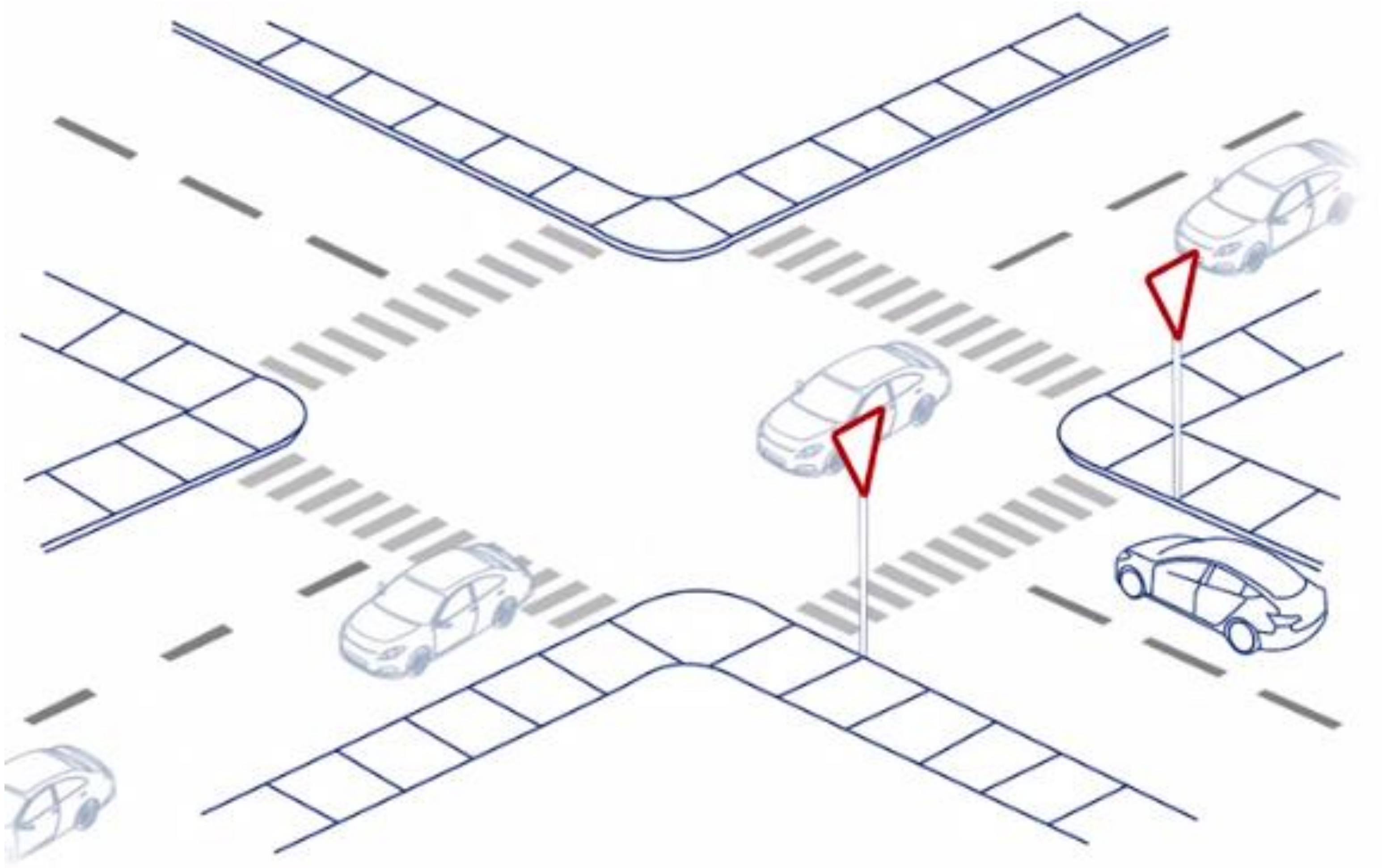
Stay stopped

Planning and Decision Making



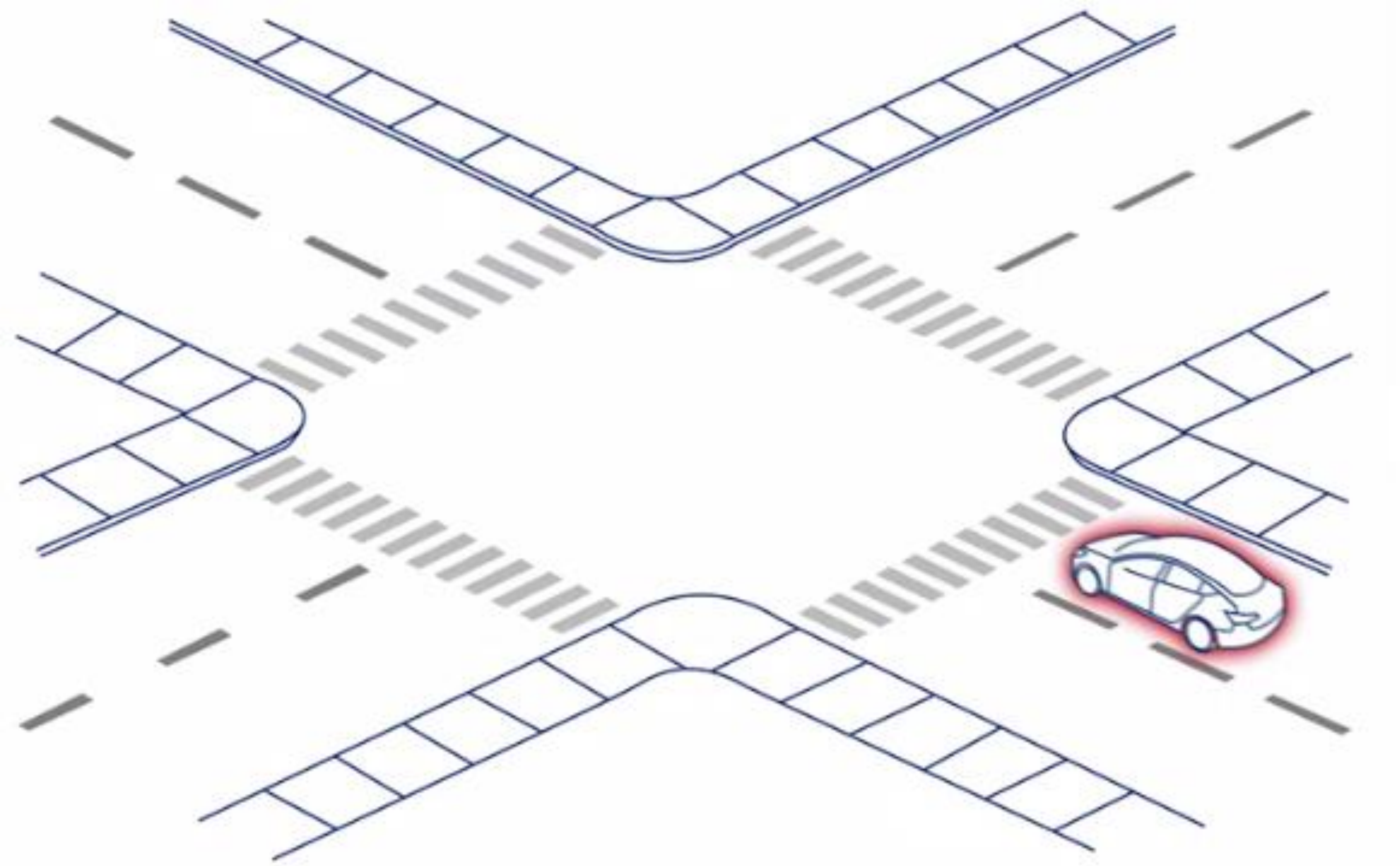
Start driving

Planning and Decision Making



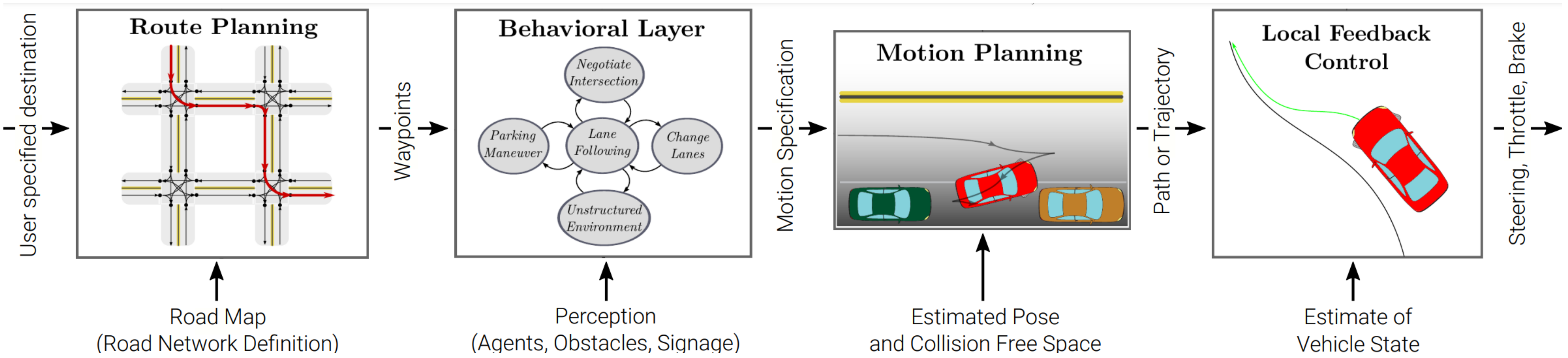
Yield to traffic

Planning and Decision Making



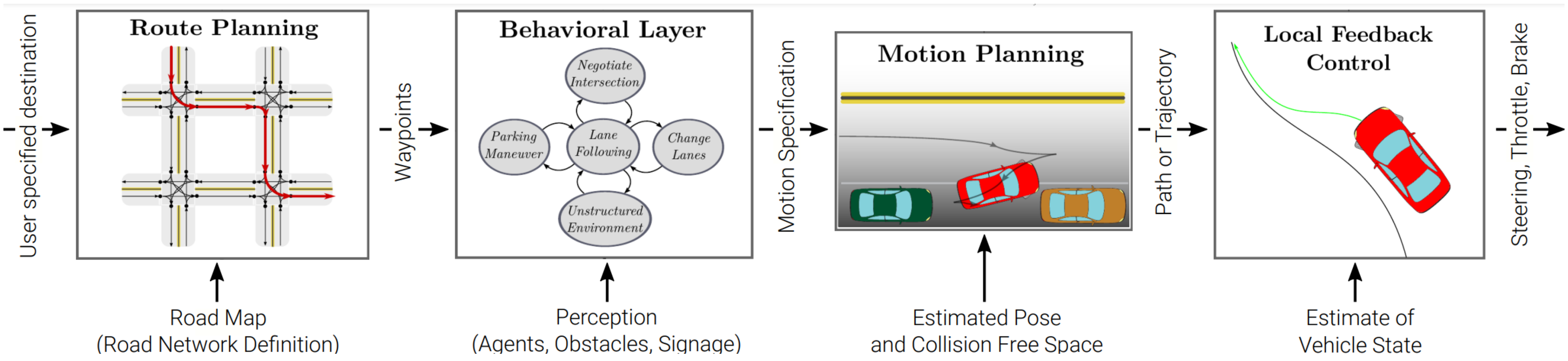
Emergency stop (and many more ...)

Planning and Decision Making



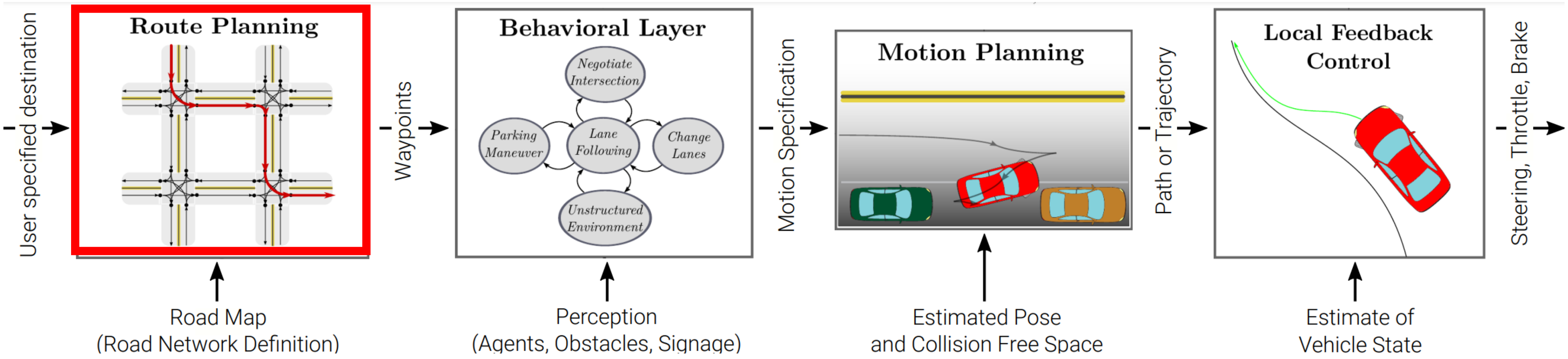
- Idea: Break planning problem into a **hierarchy of simpler problems**
- Each problem tailored to its scope and level of abstraction
- Earlier in this hierarchy means higher level of abstraction
- Each optimization problem will have constraints and objective functions

Planning and Decision Making



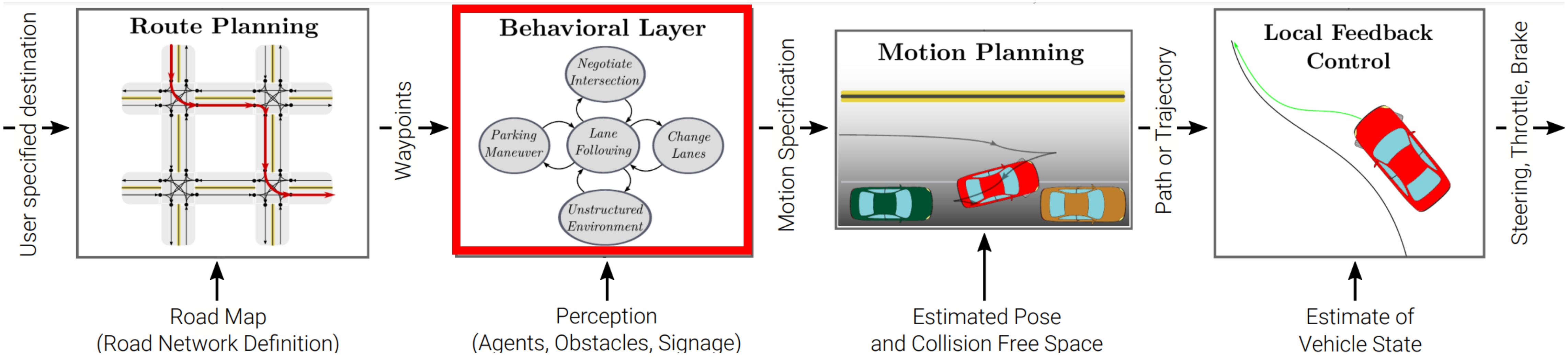
- Route planning: a route through the road network
- Behavior layer: motion specification responding to the environment
- Motion Planning: solving a feasible path accomplishing the specification.
- Feedback Control: adjusting actuation variables to correct errors in executing the path.

Route Planning



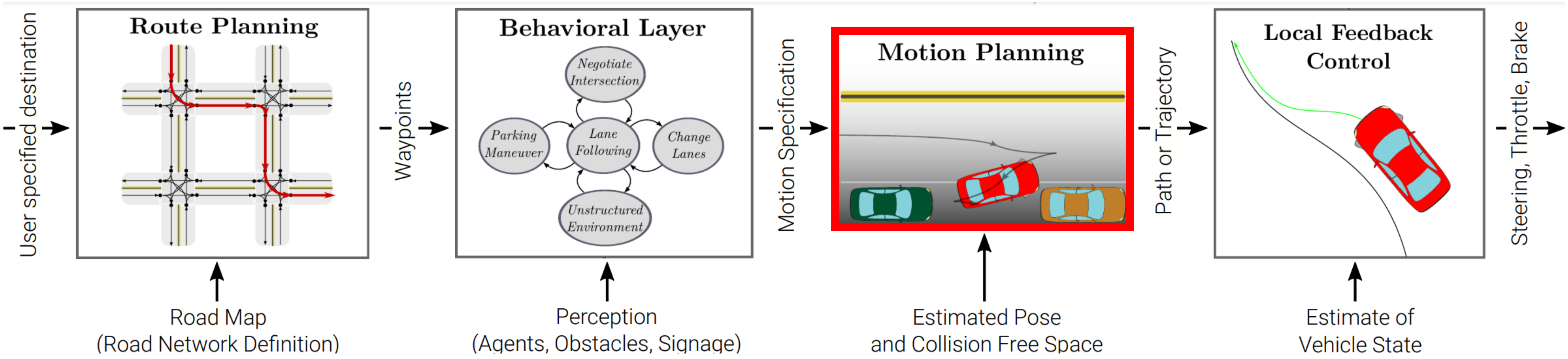
- Represent road network as directed graph
- Edge weights correspond to road segment length or travel time
- Problem translates into a minimum-cost graph network problem
- Inference algorithms: Dijkstra, A*, . . .

Behavioral Layer



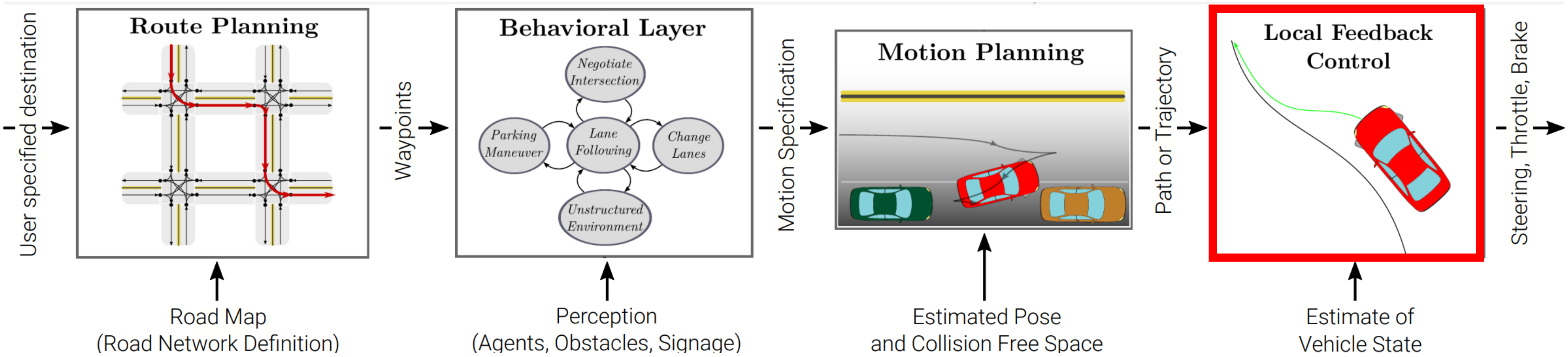
- Select driving behavior based on current vehicle/environment state
- E.g. at stop line: stop, observe other traffic participants, traverse
- Often modeled via finite state machines (transitions governed by perception)
- Can be modeled probabilistically, e.g., using Markov Decision Processes (MDPs)

Motion Planning



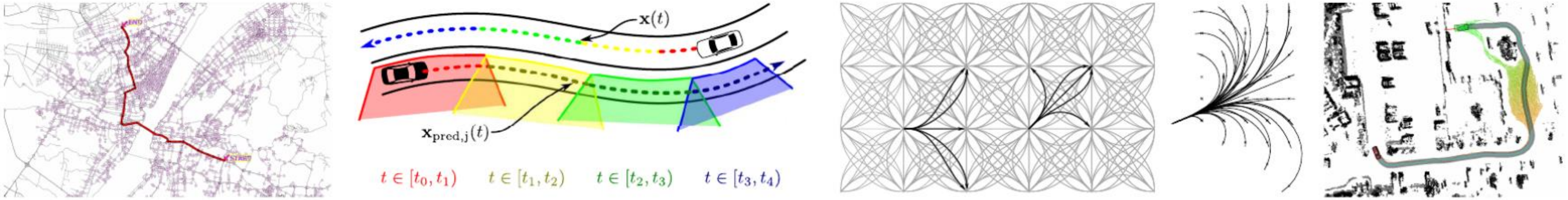
- Find feasible, comfortable, safe and fast vehicle path/trajectory
- Exact solutions in most cases computationally intractable
- Thus often numerical approximations are used
- Approaches: variational methods, graph search, incremental tree-based

Local Feedback Control

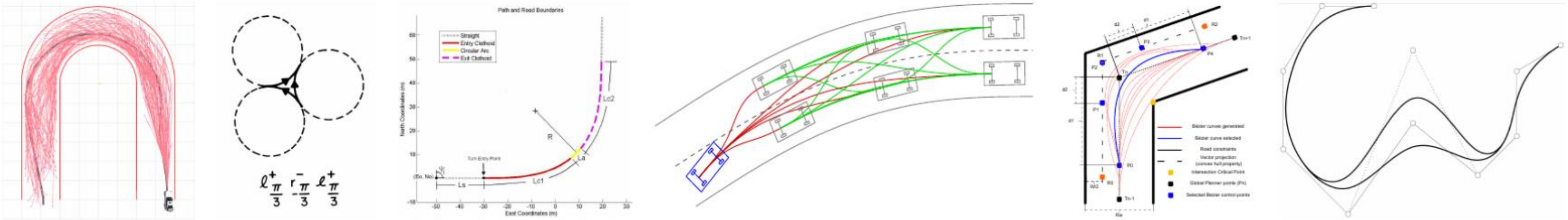


- Feedback controller executes the path/trajectory from the motion planner
- Corrects errors due to inaccuracies of the vehicle model
- Emphasis on robustness, stability and comfort
- Vehicle dynamics and control in Lecture 3

Path Algorithms



(a) Dijkstra [29] (b) FunctionOptimization [38] (c) Lattices [39] (d) A* [36]

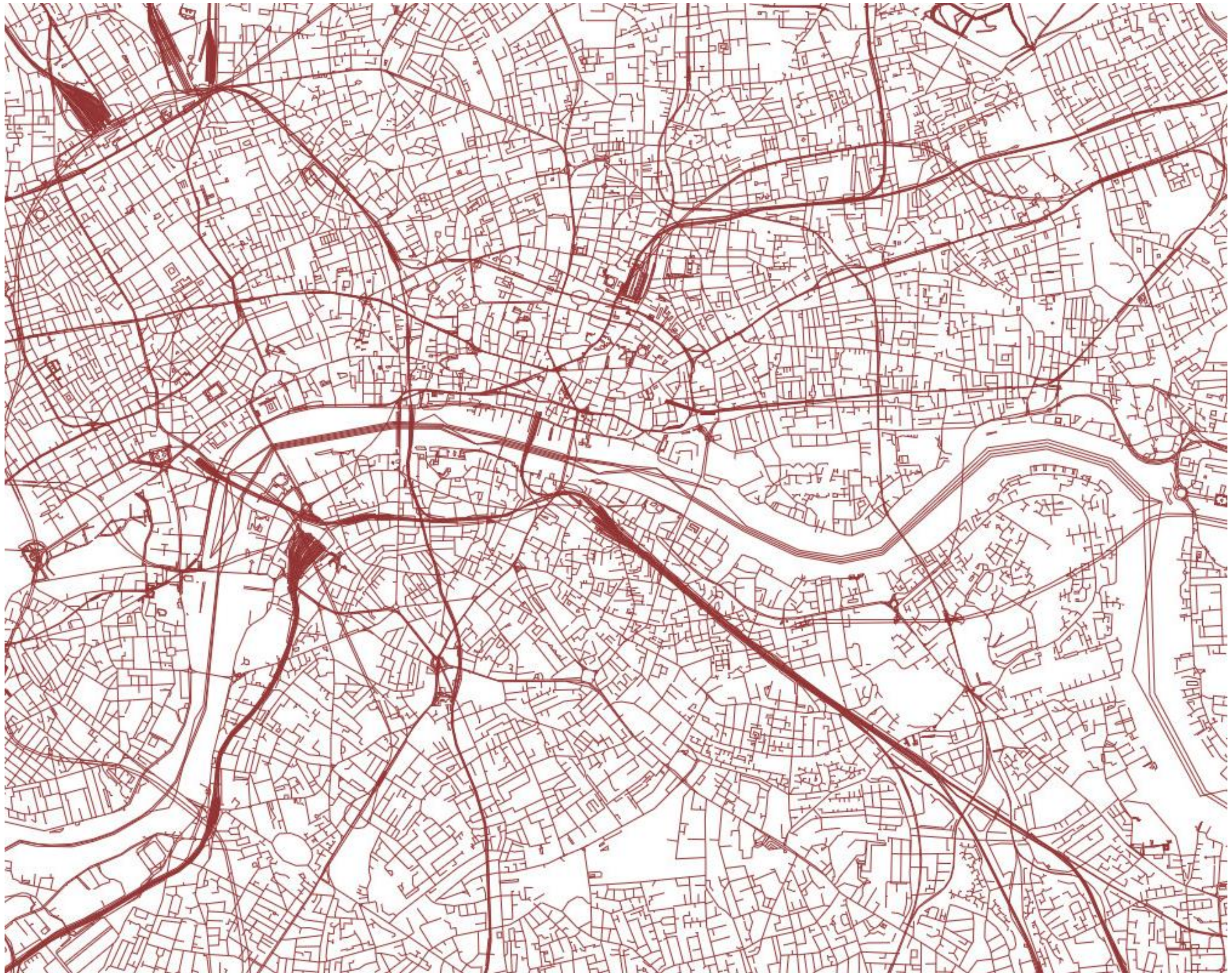


(e) RRT [40] (f) Line&Circle [41] (g) Clothoid [42] (h) Polynomial [43] (i) Bezier [44] (j) Spline [45]

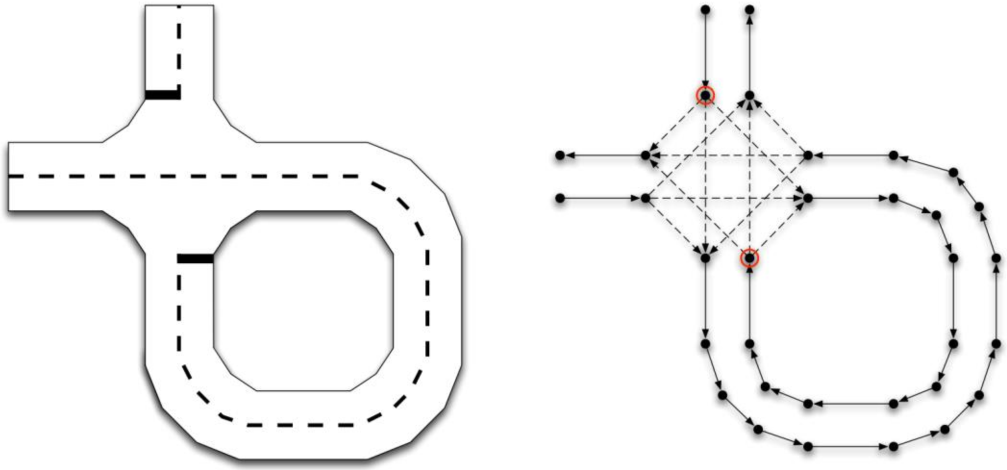
- Planning algorithms used in the autonomous driving literature
- There are many of them – we will focus only on a few today

Route Planning

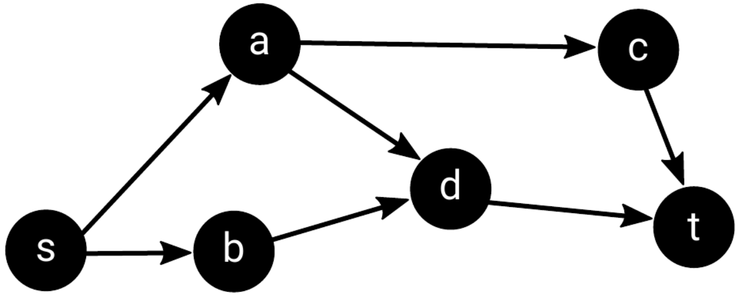
Road Networks as Graphs



Road Networks



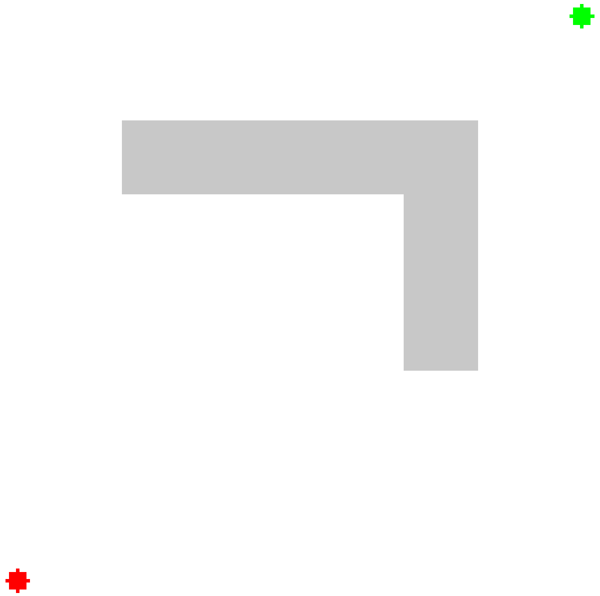
How to interpret roads in graphs



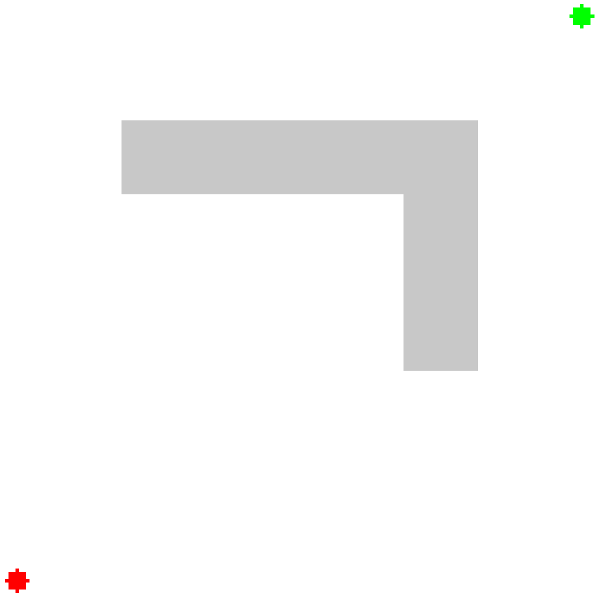
A route network is a directional graph!

Route Planning Algorithms

- Breadth First Search
- Dijkstra algorithm
- A* algorithm
- Other heuristics



Dijkstra Algorithm

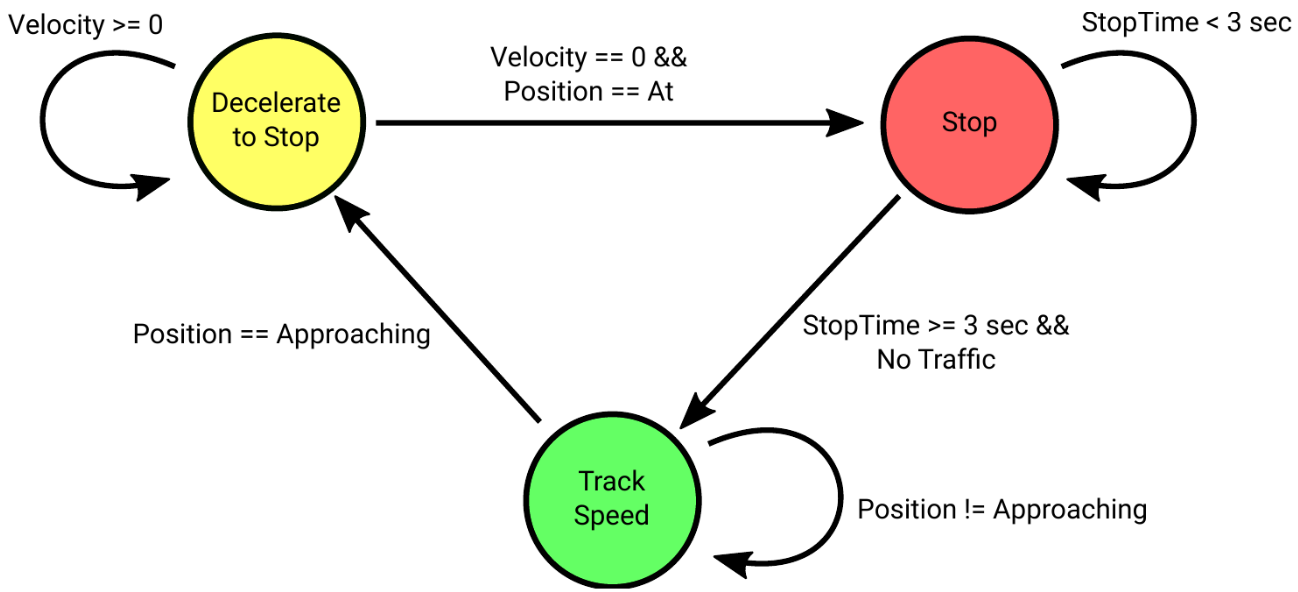


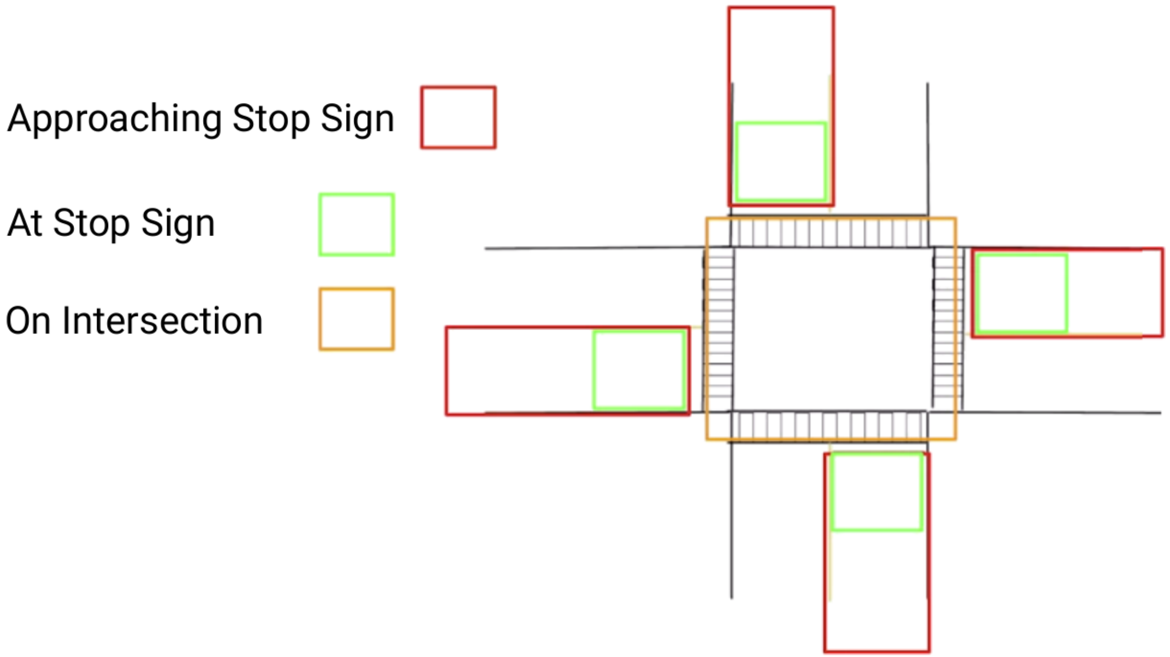
A* Algorithm

Behavior Planning

Finite State Machine for Simple Vehicle Behavior

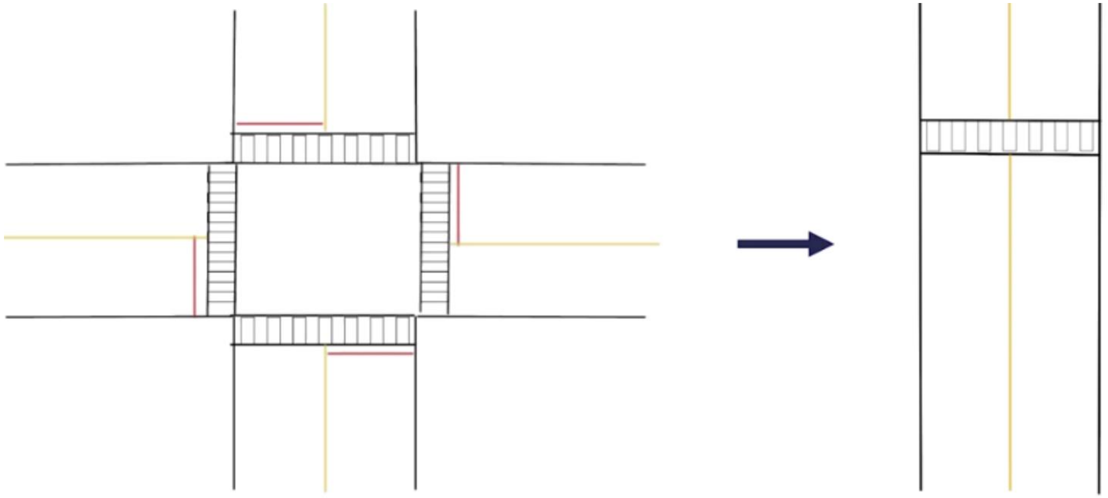
- While driving, a car needs various maneuvers (decelerating, stop, follow the lane).
- Discretizing car behaviors into atomic maneuvers and the developer design a motion planner dedicated for each maneuver.

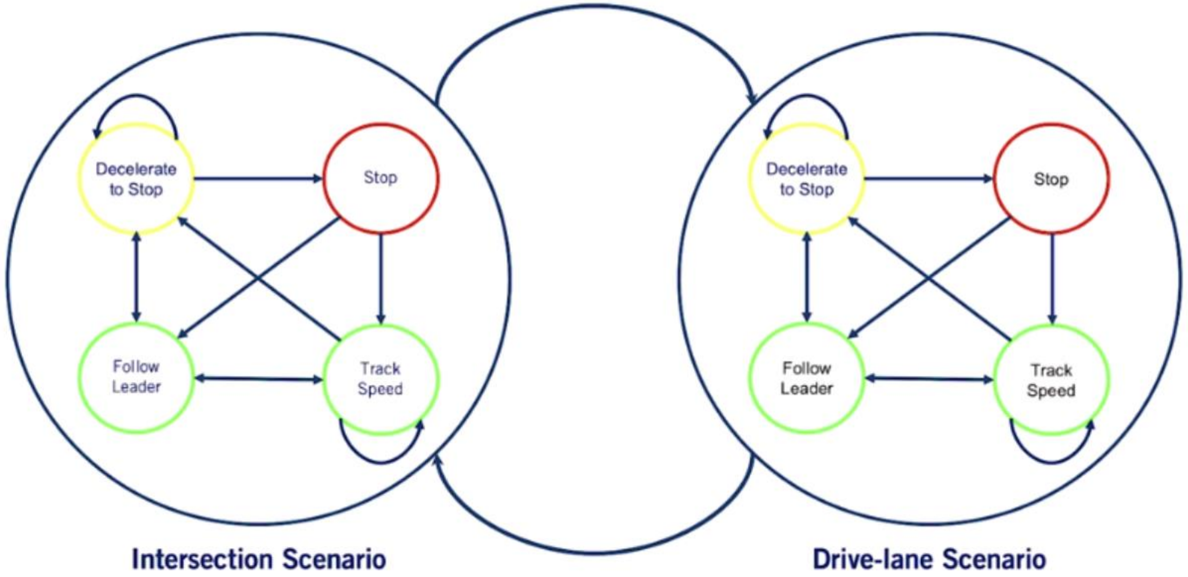


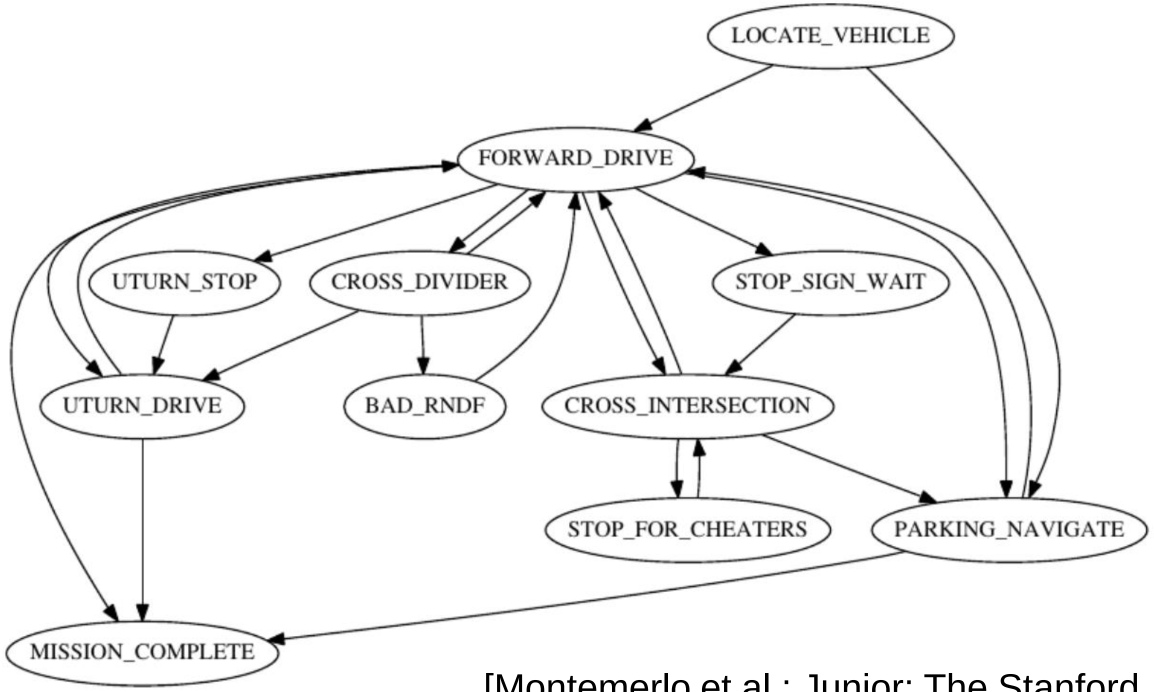


[Slide Credit: Steven Waslander]

Handling Multiple Scenarios







[Montemerlo et al.: Junior: The Stanford Entry in the Urban Challenge. JFR, 2008]

[Slide Credit: Steven Waslander]

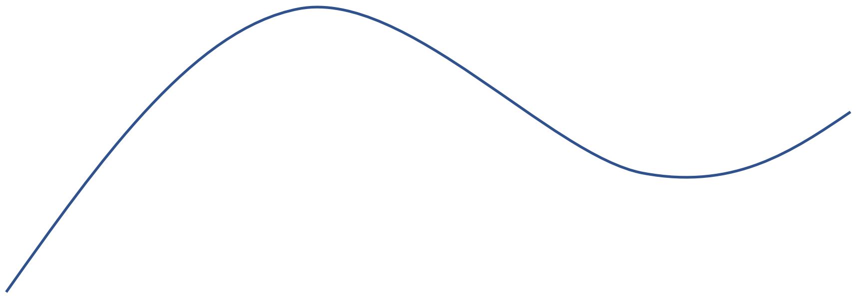
Motion Planning

Variational Optimization (함수 최적화)

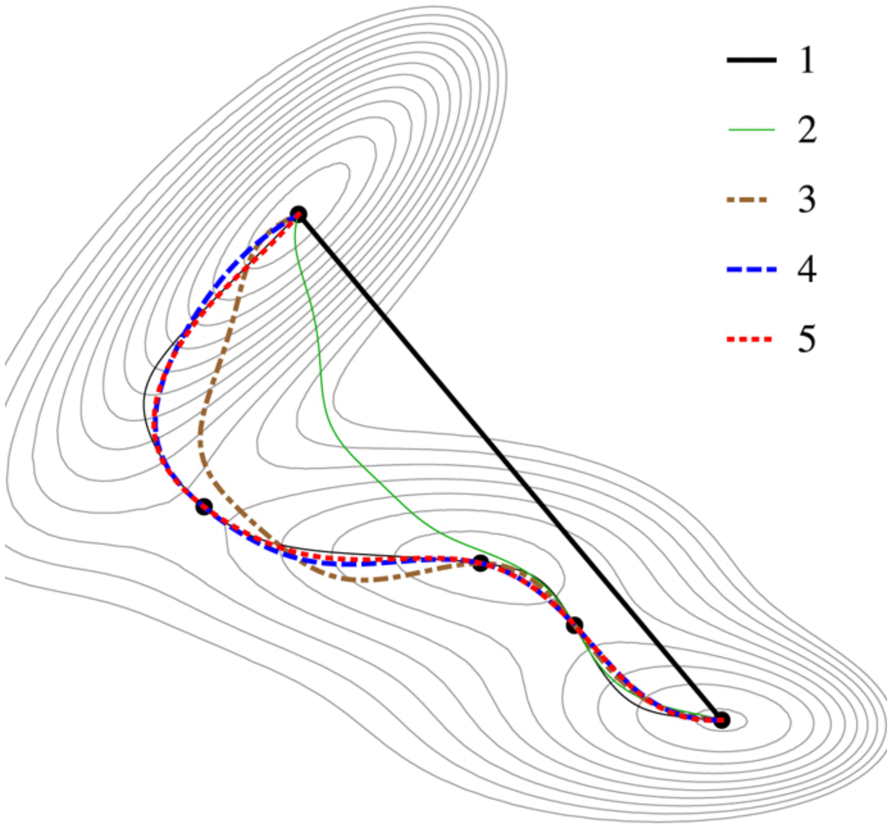
- Variational methods minimize a functional (a function that takes a function as input):

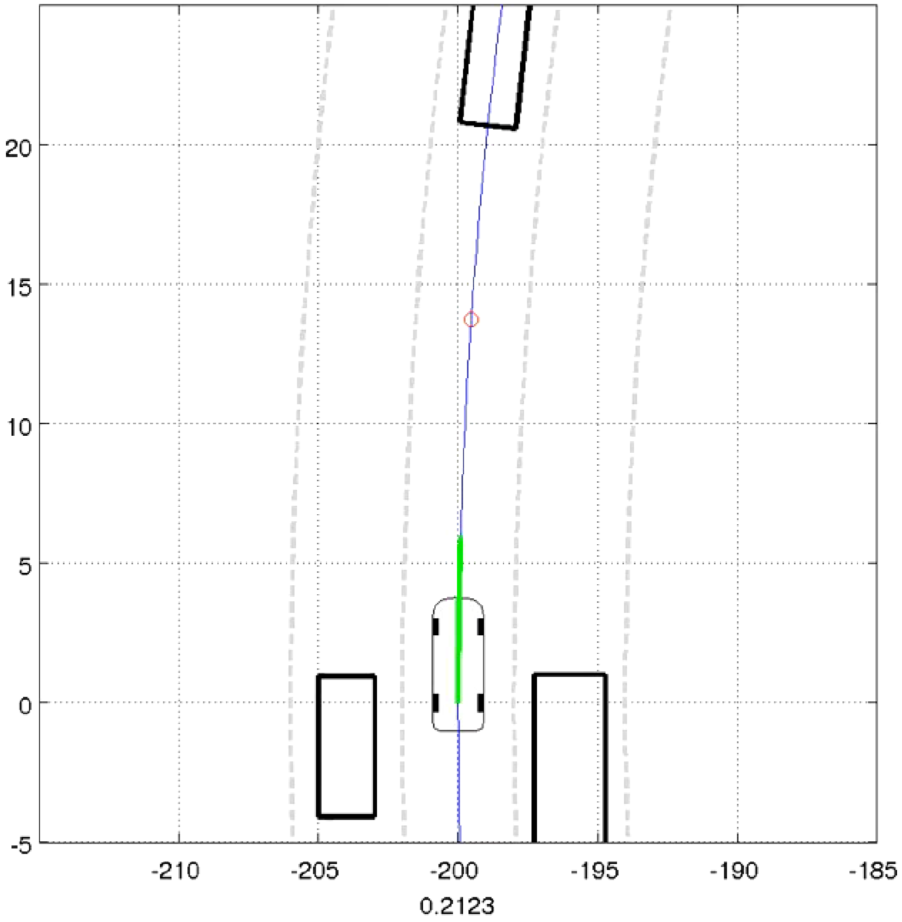
$$\underset{\pi}{\operatorname{argmin}}\, J(\pi) = \int_0^T f(\pi)\, dt$$

$$\text{s.t. } \pi(0) = \mathbf{x}_{init} \wedge \pi(T) \in \mathbf{x}_{goal}$$



Variational Optimization examples

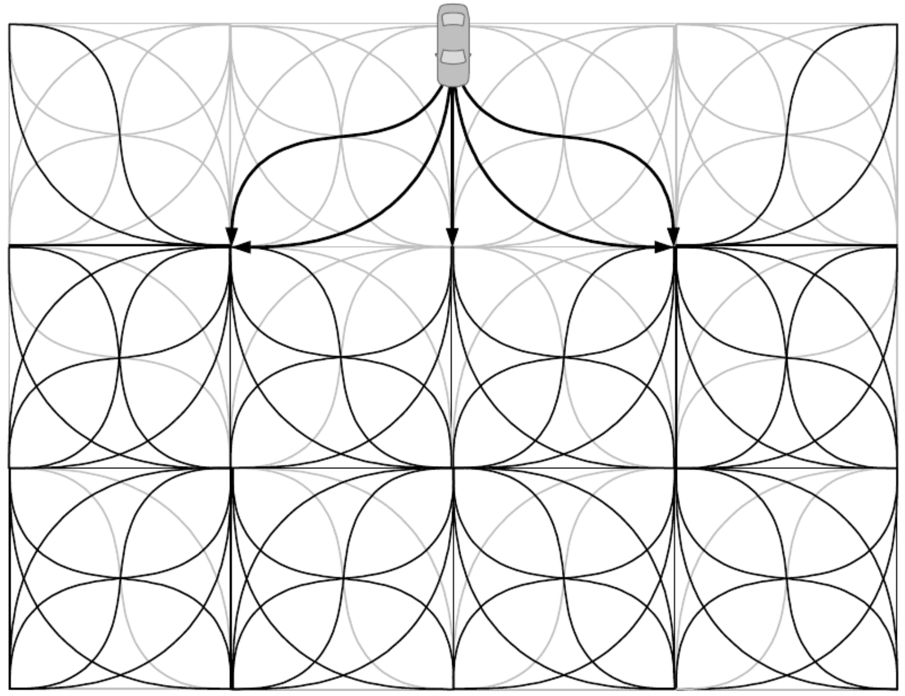


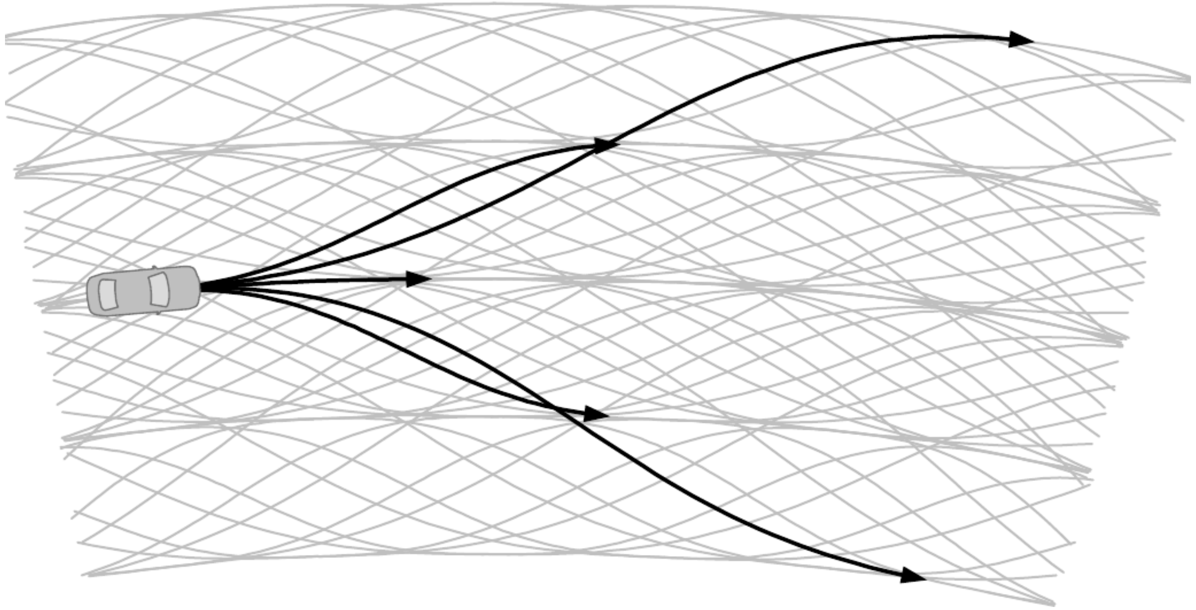


Minimizing the 1st derivative of a track

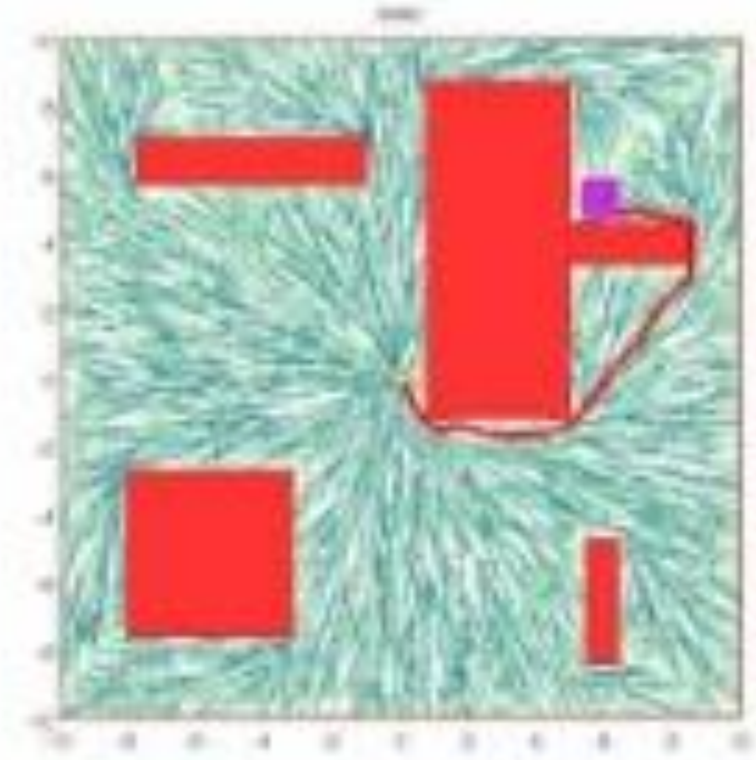
Graph Search Methods

- Discretize the action space to detour variational optimization.





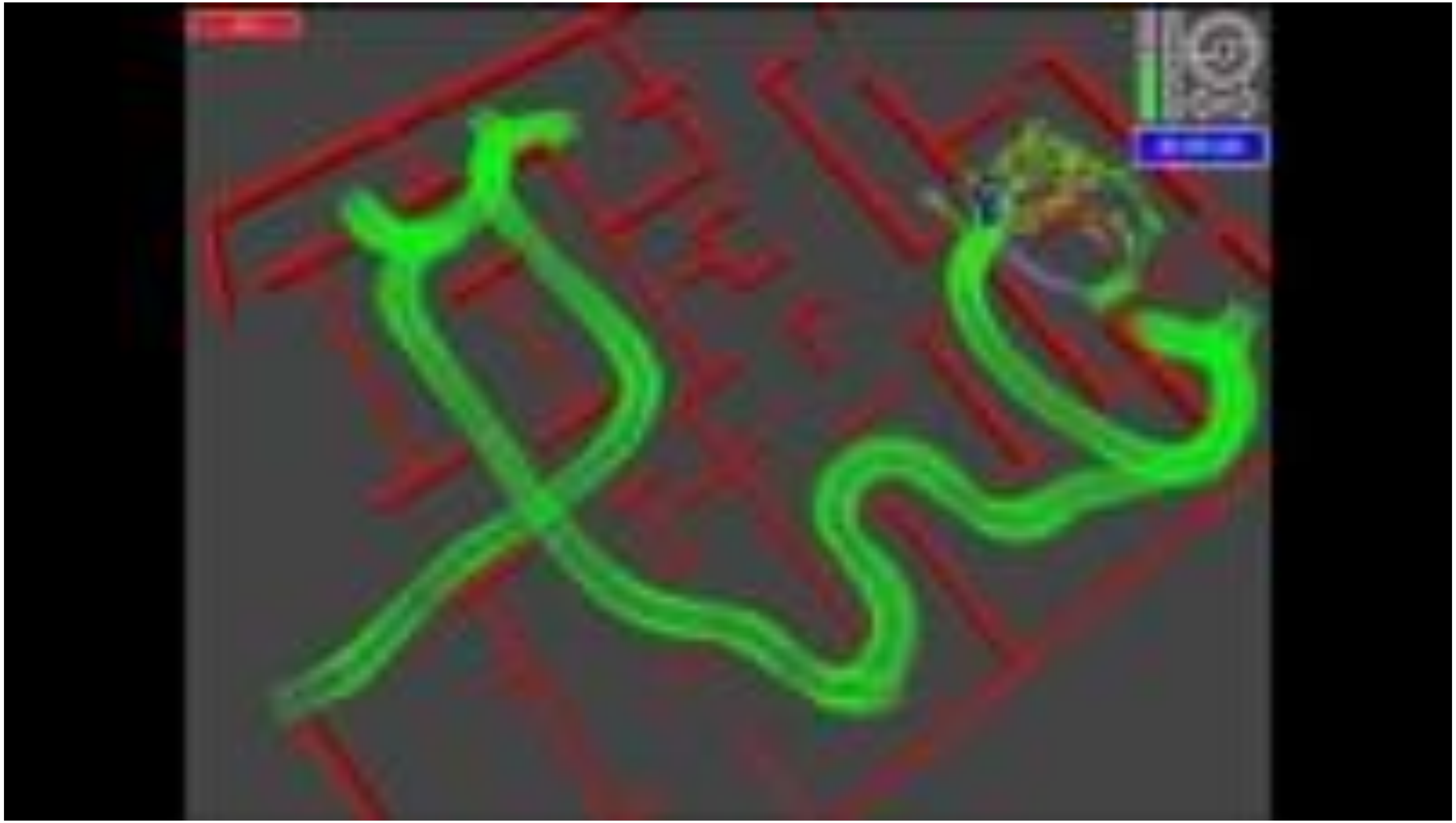
Incremental Search Techniques



- Incrementally build increasing finer discretization of configuration space.
- Rapidly exploring random trees (RRT) and RRT*

LaValle: Rapidly-exploring random trees: A new tool for path planning. Techical Report, 1998.

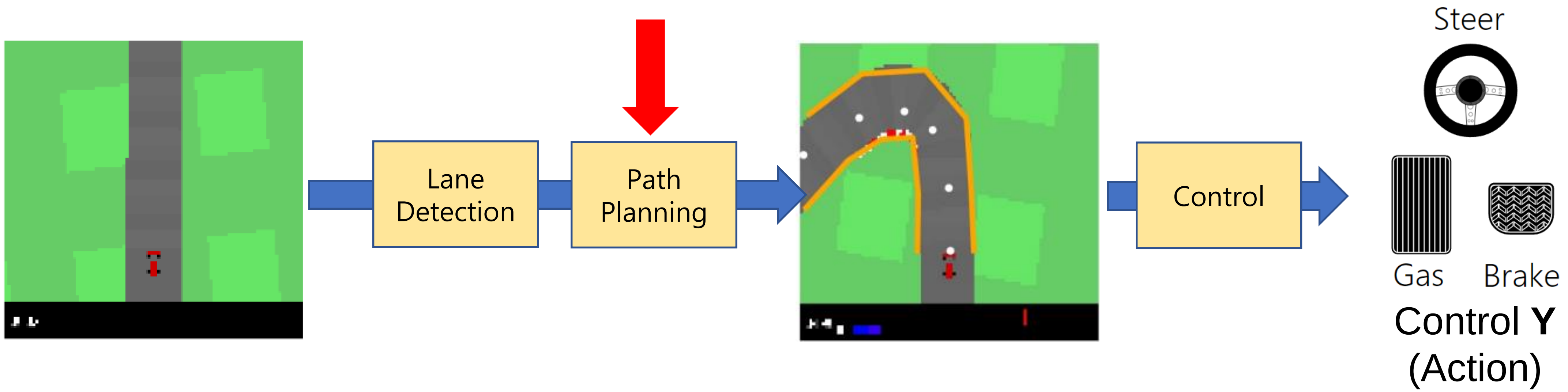
RRT meets A* algorithm



Dolgov et al.: Practical Search Techniques in Path Planning for Autonomous Driving. STAIR, 2008.

Experiment

Modular Pipeline Overview



- Implement simplified version of modular pipeline.
- You will understand basic concepts and get experiences of developing a simple self-driving application.

Path Planning

- Template
  - waypoint_prediction.py
  - Test_waypoint_prediction.py for testing

a) **Road Center:**

- Use the lane boundary splines and derive lane boundary points for 6 equidistant spline parameter values

  → waypoint_prediction()
- Determine the center between lane boundary points with the same spline parameter

  → waypoint_prediction()

Path Planning

b) Path Smoothing:

- Improve the path by minimizing the following objective regarding the waypoints *x* given the center waypoints *y*

$$\underset{x_1,\ldots,x_N}{\operatorname{argmin}} \sum_i \left|y_i - x_i\right|^2 - \beta \sum_n \frac{\left(x_{n+1} - x_n\right)\cdot\left(x_n - x_{n-1}\right)}{\left|x_{n+1} - x_n\right|\left|x_n - x_{n-1}\right|}$$

- Explain the effect of the second term
- Implement second term
  → curvature()

Path Planning

c) Target Speed Prediction:

- Implement a function that outputs the target speed for the predicted path in the state image, using

$$v_{\text{target}}\left(x_1,\ldots,x_N\right)=\left(v_{\max}-v_{\min}\right)\exp\left[-K_v\cdot\left|N-2-\sum_n\frac{\left(x_{n+1}-x_n\right)\cdot\left(x_n-x_{n-1}\right)}{\left|x_{n+1}-x_n\right|\left|x_n-x_{n-1}\right|}\right|\right]+v_{\min}$$

As initial parameters use: $v_{\max}=60, v_{\min}=30,\ \text{and}\ K_v=4.5$

→ target_speed_prediction()