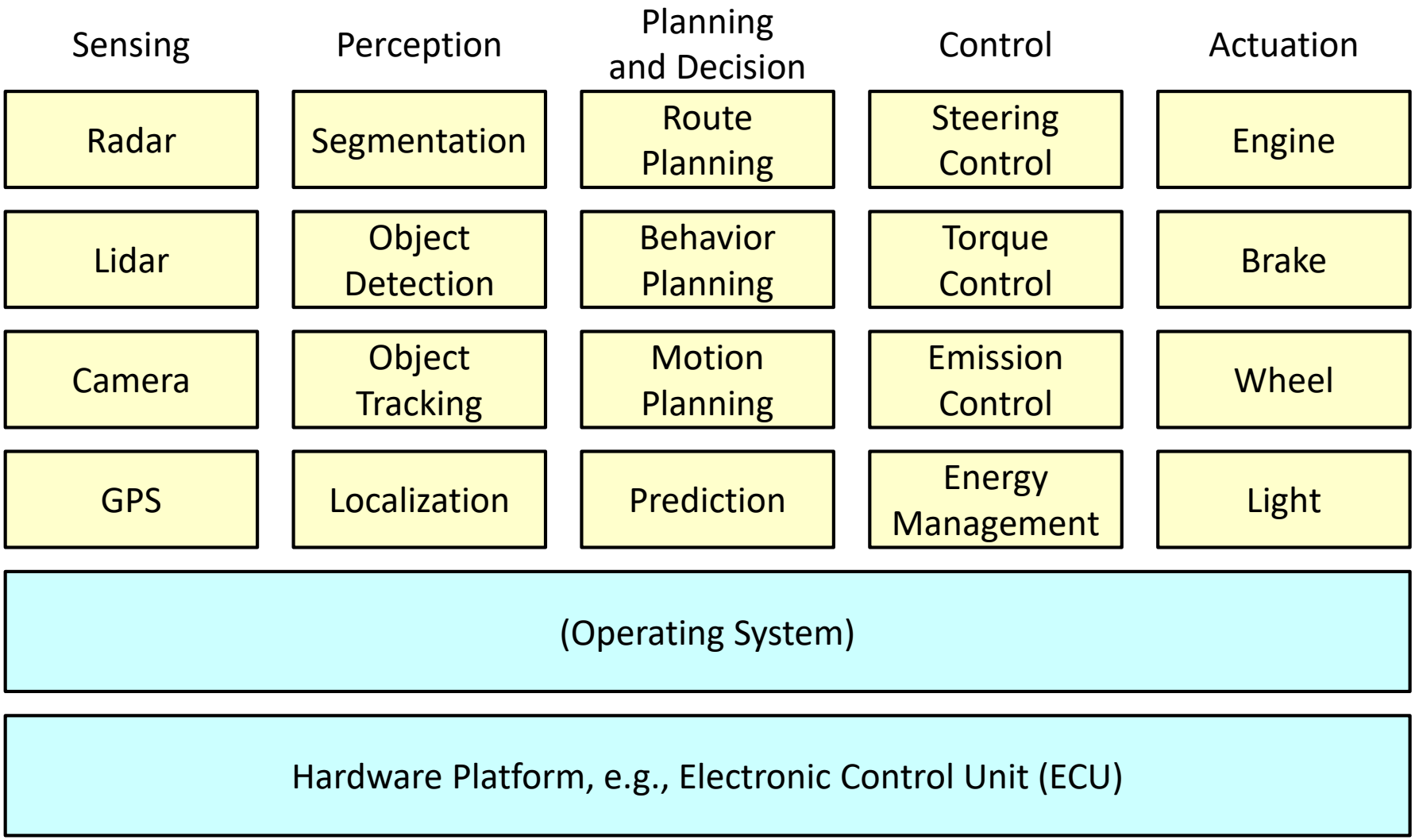
Introduction to Intelligent Vehicles
[9. Planning and Control]

Chung-Wei Lin
cwlin@csie.ntu.edu.tw
CSIE Department
National Taiwan University

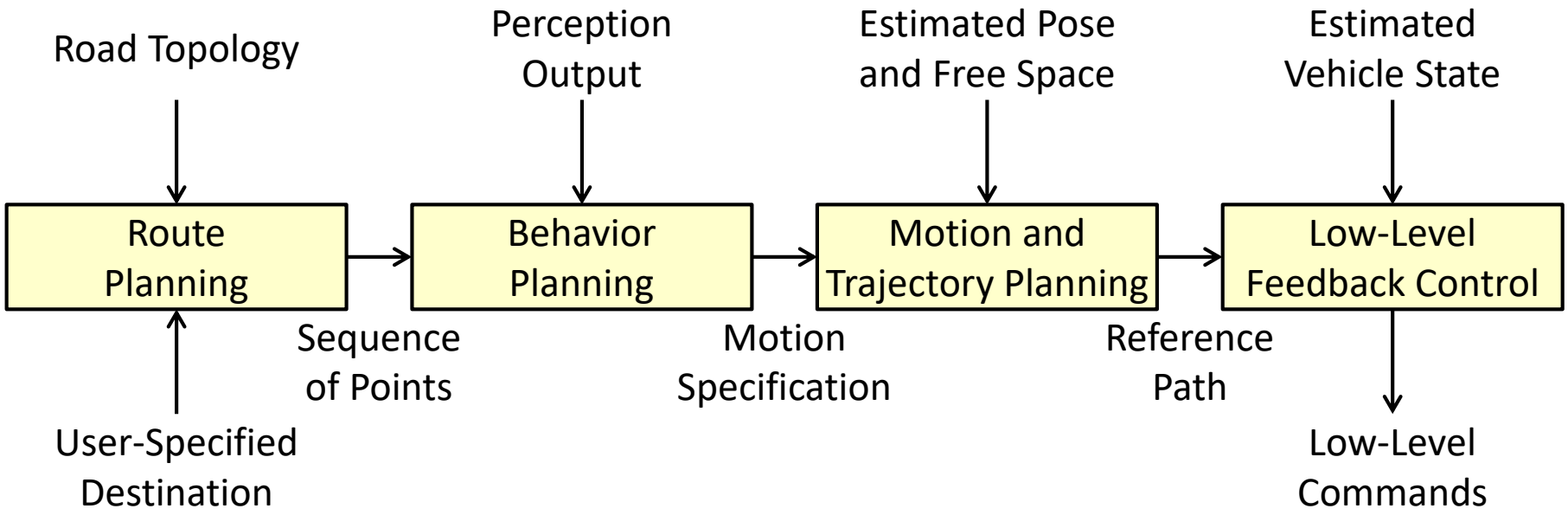
We would like to thank Prof. Jyo Deshmukh (USC) for contributing the lecture material

Layered View of Autonomous Vehicles



Outline

- **Route Planning**
- Behavioral Planning
- Motion and Trajectory Planning
- Low-Level Feedback Control



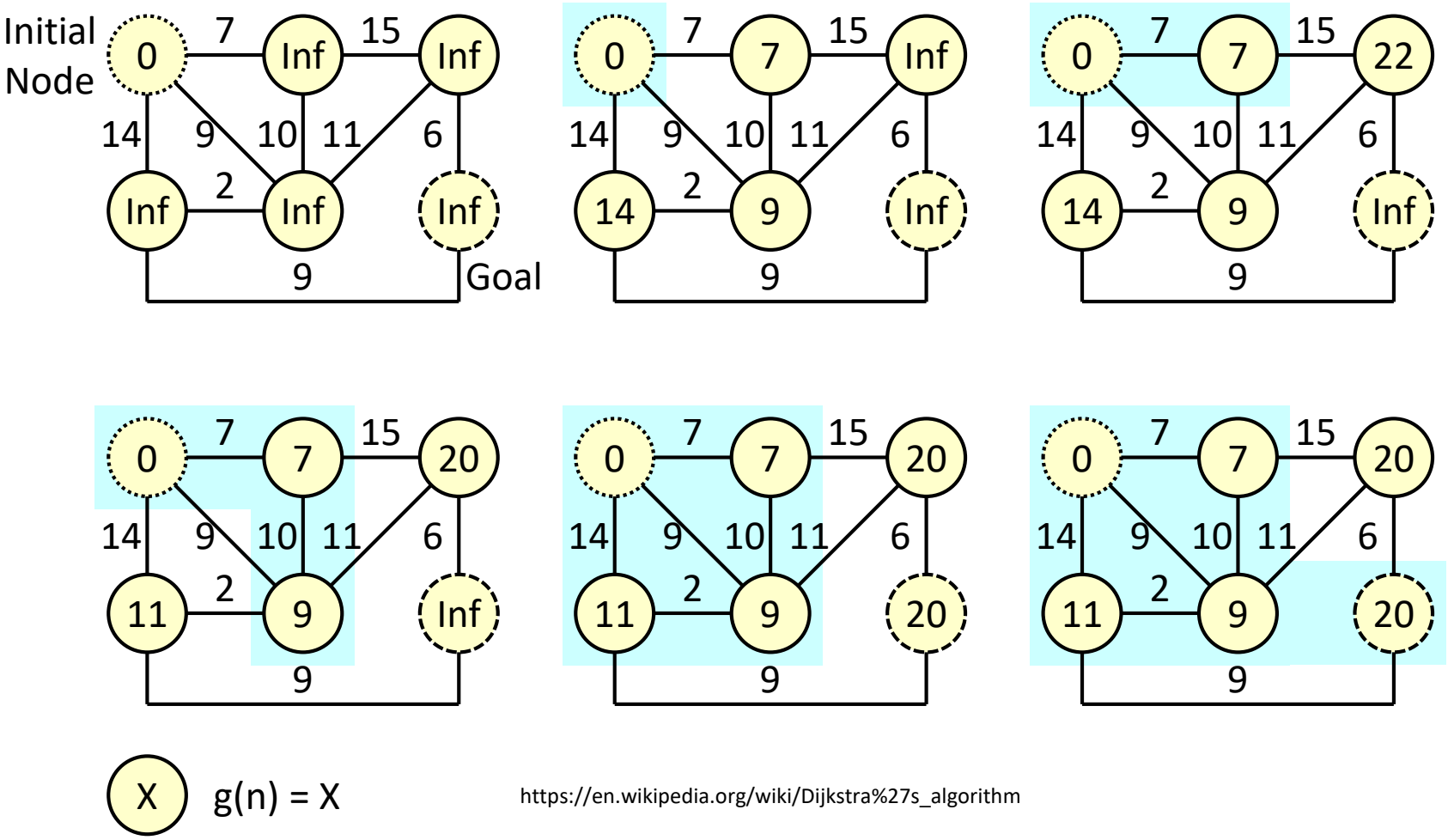
Route Planning

- Goal
  - Travel from place A to place B
- Road network and topology represented as a directed graph
  - Nodes represent way-points
  - Edges represent a cost of traveling between nodes
    - A cost can represent time, distance, fuel economy, a weighted combination of some of these, etc.
- Simplest solutions based on Dijkstra's shortest-path and A* search algorithms
  - They do not scale well in practice
  - Lots of different algorithms proposed in the literature

A* Search Algorithm

- Construct a tree of paths from the initial node and expand paths one step at a time until one path reaches the goal
- In each iteration, it determines which of its partial paths to expand based on an estimate of the cost
  - It selects the path minimizes f(n) = g(n) + h(n)
    - g(n): cost of path from the initial node to n
    - h(n): estimated cost from n to the goal
- Dijkstra's shortest path algorithm is a special case of A*
  - h(n) = 0

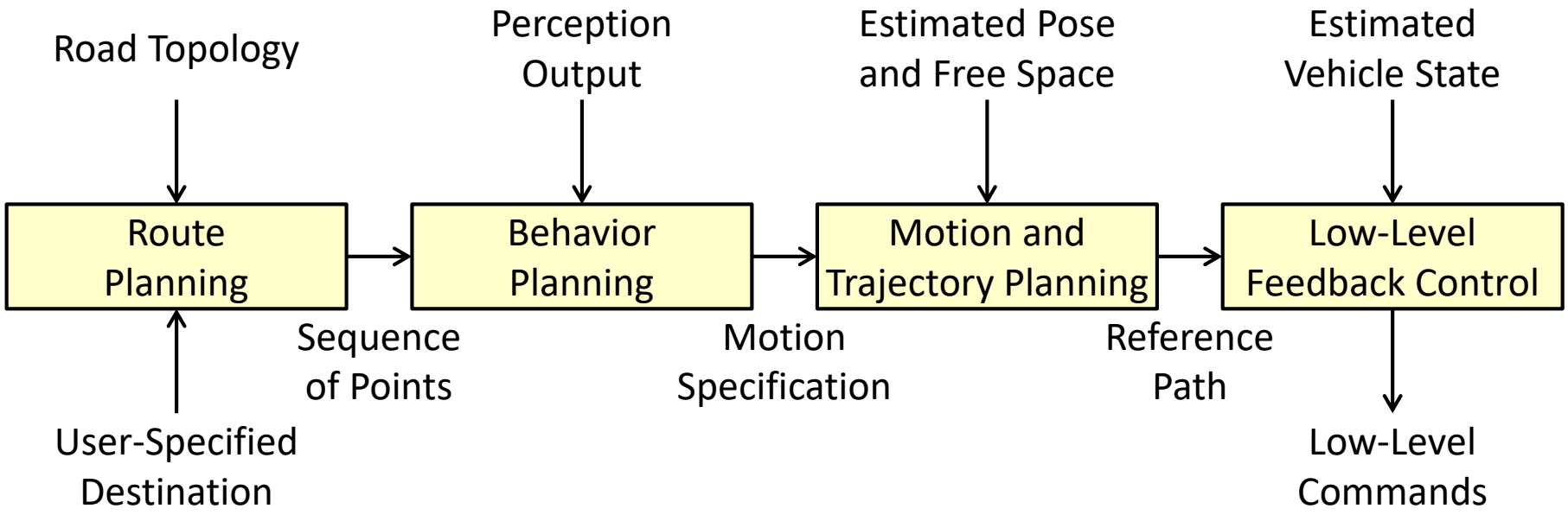
Dijkstra's Shortest Path Algorithm



https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

Outline

- ❑ Route Planning
- ❑ **Behavioral Planning**
- ❑ Motion and Trajectory Planning
- ❑ Low-Level Feedback Control



Behavior Planning (1/2)

- Goals
  - Navigate selected route (by route planning) and interact with traffic participants (pedestrians, bicycles, other cars)
  - Select appropriate **driving action** (e.g., cruise in lane, turn right, turn left, etc.) based on perceived mode
- Finite state machine (FSM) based approach
  - Model driving contexts and behaviors as finite sets and use finite state machines to represent possible behaviors
  - Choose appropriate transitions that ensure a desired behavior
    - A 2-player game between the vehicle and the environment
    - Game-theoretic approaches can identify winning strategies for the vehicle
- Connections to ADAS, CACC, and intersection management
  - Modes of ADAS and CACC
  - Decision and strategy in a game

Behavior Planning (2/2)

- Behavioral planning under uncertainty
  - FSM-based view may not adequately address uncertainty
    - Uncertainty exists in the intention of traffic participants
  - Intention prediction of vehicles, bicycles, and pedestrians
    - Techniques based on Gaussian mixture models
    - Switching linear dynamical and stochastic systems
    - Gaussian Process Regression
  - Probabilistic planning formalisms
    - Markov Decision Processes such as Partially Observable Markov Decision Processes (POMDPs)

POMDPs (1/3)

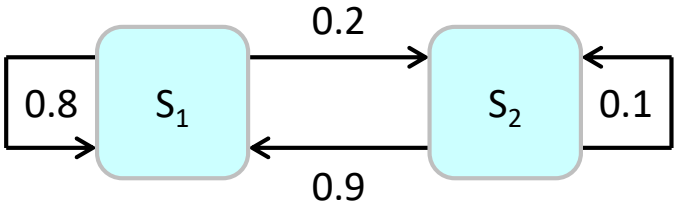
- 6-tuple (Q,A,O,T,Z,R)
  - Q: set of states
  - A: set of actions
  - O: set of observations
  - T: transition function
    - T(q,a,q'): the probability of transiting to q' under action a in state q
  - Z: observation function
    - Z(q,a,o): the probability of observing o (under action a) in state q
  - R: reward function
    - R(q,a): a reward for taking action a in state q
    - Reinforcement learning

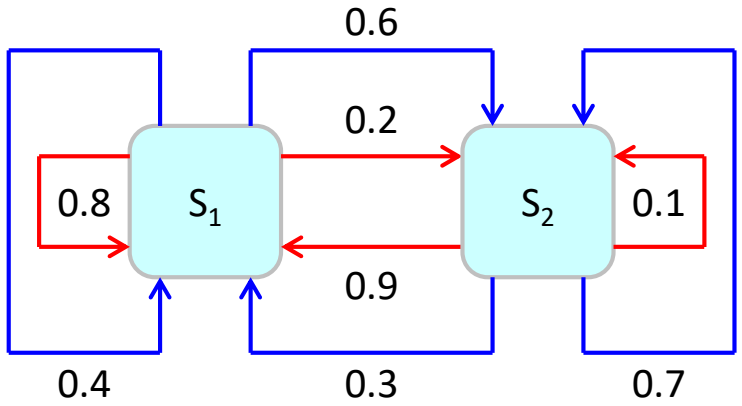
POMDPs (2/3)

Markov Models		Do we have control over the state transitions?	
		No	Yes (Controller Actions)
Are the states completely observable?	Yes	Markov Chain	Markov Decision Process (MDP)
	No	Hidden Markov Model (HMM)	Partially Observable Markov Decision Process (POMDP)

https://www.cs.cmu.edu/~ggordon/780-fall07/lectures/POMDP_lecture.pdf

POMDPs (3/3)



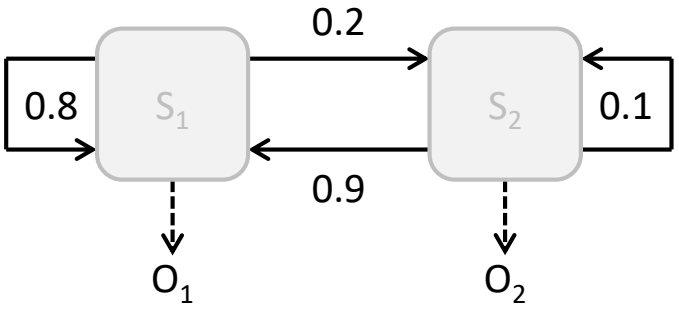


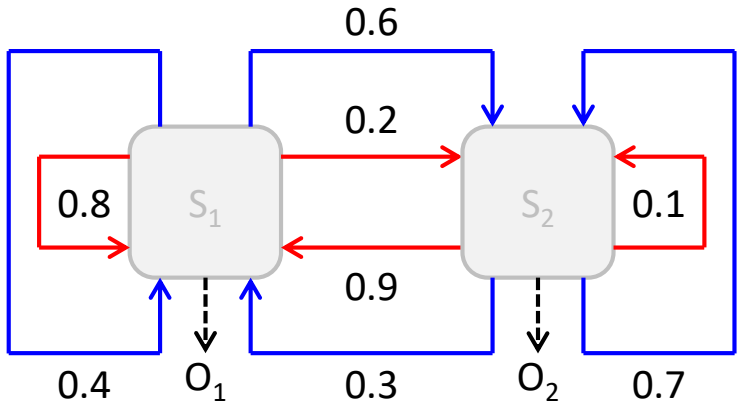
observable

non-observable

action 1

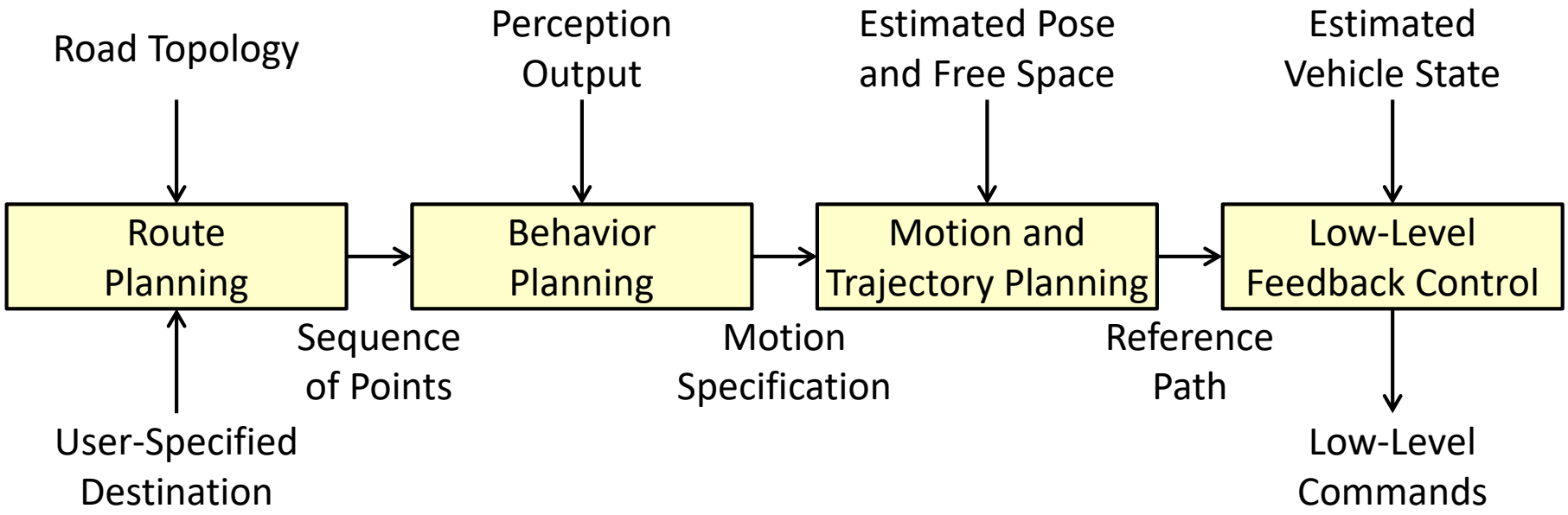
action 2





Outline

- ❑ Route Planning
- ❑ Behavioral Planning
- ❑ **Motion and Trajectory Planning**
- ❑ Low-Level Feedback Control



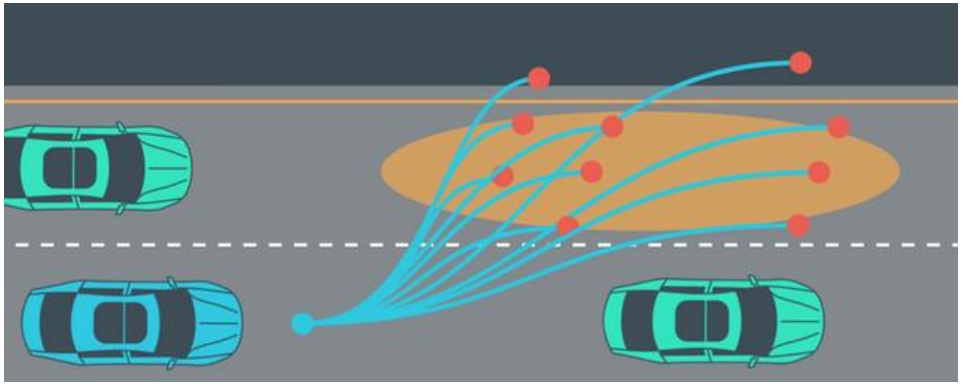
Motion and Trajectory Planning

❑ Goals

- ➢ Translate behavior plan into a reference trajectory (or path) to be followed by the vehicle
  - Behavior planning decides a high-level set of maneuvers, e.g., cruise in lane, turn right, turn left, etc.
- ➢ Compute a feasible reference trajectory (or path) that respects feasibility and obstacle avoidance and optimize certain costs, e.g., maximizing comfort, minimizing battery usage, etc.

❑ Popular approaches

- ➢ Variational methods
- ➢ Graph-search methods
- ➢ Tree-based methods



https://medium.com/udacity/self-driving-path-planning-brought-to-you-by-udacity-students-13c07bcd4f32

A General Formulation (1/2)

- X: set of configurations
  - X_{init} in X: set of initial configurations (e.g., locations and other things)
  - X_{goal} in X: set of goal configurations
  - $X_{free}(t)$ in X: set of allowed (feasible) configurations at time t in [0,T]
    - This can encode path constraints, dynamic and static obstacles, etc.
- T: planning horizon
- $\prod(X,T)$: set of all continuous functions from [0,T] to X
  - π in $\prod$: a continuous function from [0,T] to X = a trajectory
- $D(\pi(t), \pi'(t), \pi''(t), \ldots)$: differential constraint
  - Specify constraints on smoothness of the path, velocity, acceleration, jerk constraints, etc.
- J: cost function mapping any trajectory π to a cost

A General Formulation (2/2)

- Optimal trajectory can be stated as follows

$$\pi^* = \min J(\pi)$$

$$\text{s.t. } \pi(0) \text{ in } X_{init}$$

$$\pi(T) \text{ in } X_{goal}$$

$$\pi(t) \text{ in } X_{free}(t) \text{ for all } t \text{ in } [0,T]$$

$$D(\pi(t), \pi'(t), \pi''(t), \ldots)$$

Variational Methods

- Rely on results from nonlinear continuous optimization
- Standard form

$$\pi^* = \min J(\pi)$$

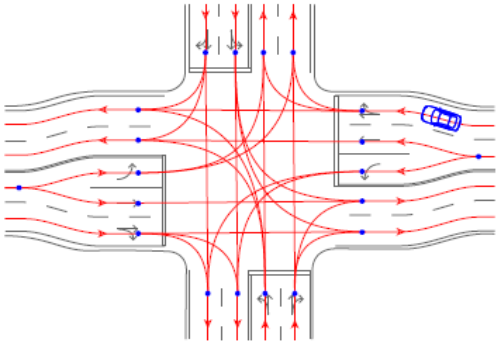
$$\text{s.t. } f(\pi(t), \pi'(t), \pi''(t), \ldots) = 0$$

$$g(\pi(t), \pi'(t), \pi''(t), \ldots) \leq 0$$

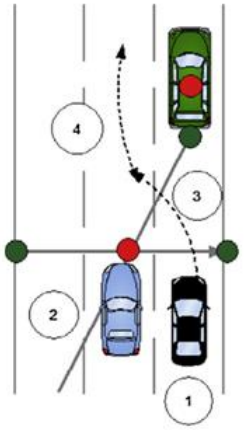
- Standard trick to go from constrained to unconstrained optimization is to introduce penalty functions
  - Example: $J^{+}(\pi) = J(\pi) + C \int_{0 \text{ to } T} [\, \| f(\ldots) \|^2 + \| \max(0, g(\ldots)) \|^2 \,] \, dt$

Graph-Search Methods

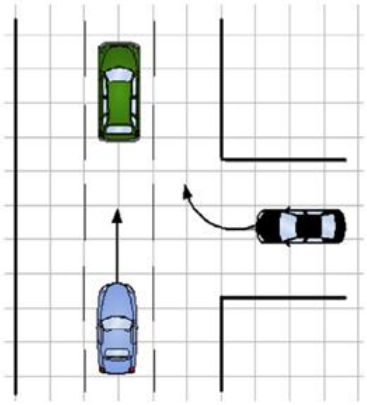
- Require discretization of the configuration/state space
  - Hand-crafted lane graphs (right figure)
  - Geometric representations (bottom figure)
  - Sampling-based discretization
    - Explore reachability of feasible space by trying to apply controls and checking for collisions
      - Control functions: random, heuristic, exact, optimal
    - Main purpose: construct a roadmap for planning



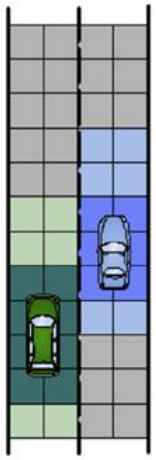
Paden et al, "A survey of motion planning and control techniques for self-driving urban vehicles," T-IV, 2016



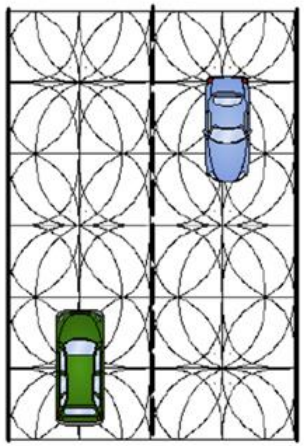
Voronoi Diagram



Occupancy Grid Map



Cost Map



State Lattice

Driving Corridor

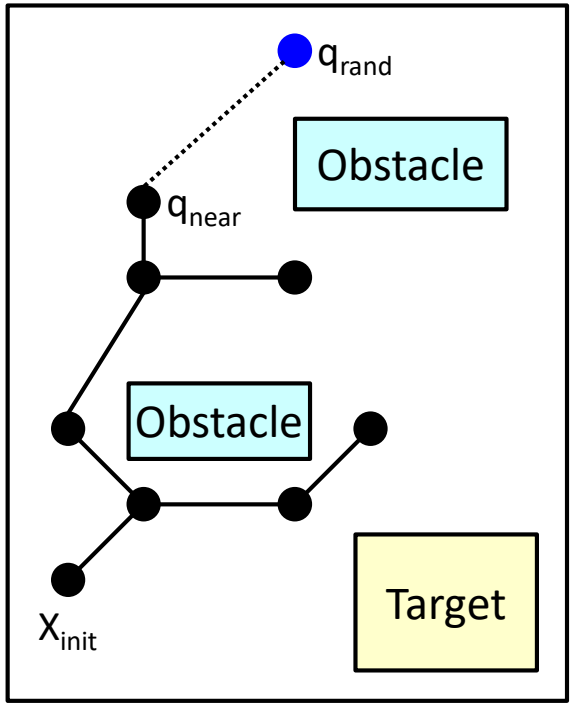
Tree-Based Methods (1/2)

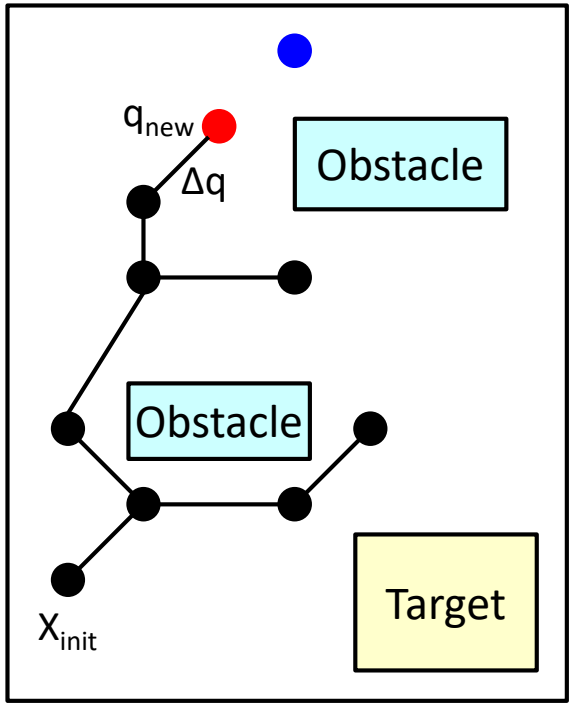
- Rapidly exploring Random Trees (RRTs)
- Algorithm [Wikipedia]
  - Input
    - Initial configuration q_{init}, number of vertices in RRT K, incremental distance Δq
  - Output
    - RRT graph G
  - Pseudocode

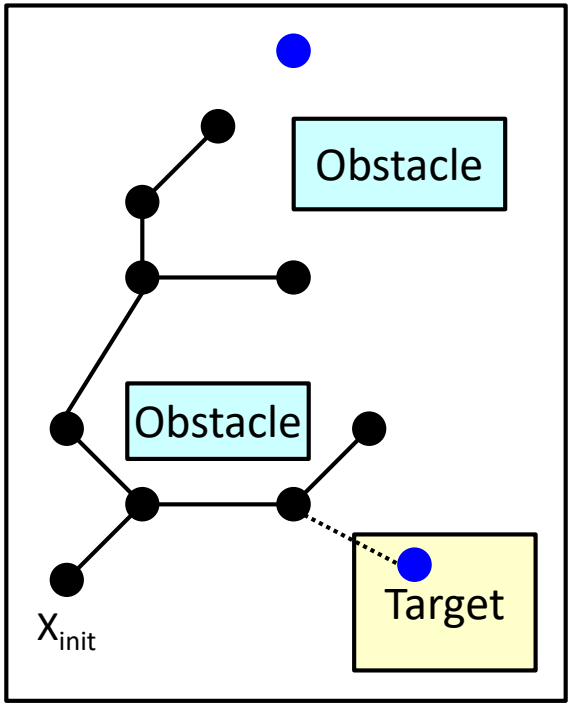
```
G.init(q_init)
for k = 1 to K
    q_rand ← RAND_CONF()
    q_near ← NEAREST_VERTEX(q_rand, G)
    q_new ← NEW_CONF(q_near, q_rand, Δq)
    G.add_vertex(q_new)
    G.add_edge(q_near, q_new)
return G
```

Tree-Based Methods (2/2)

- Rapidly exploring Random Trees (RRTs) in action

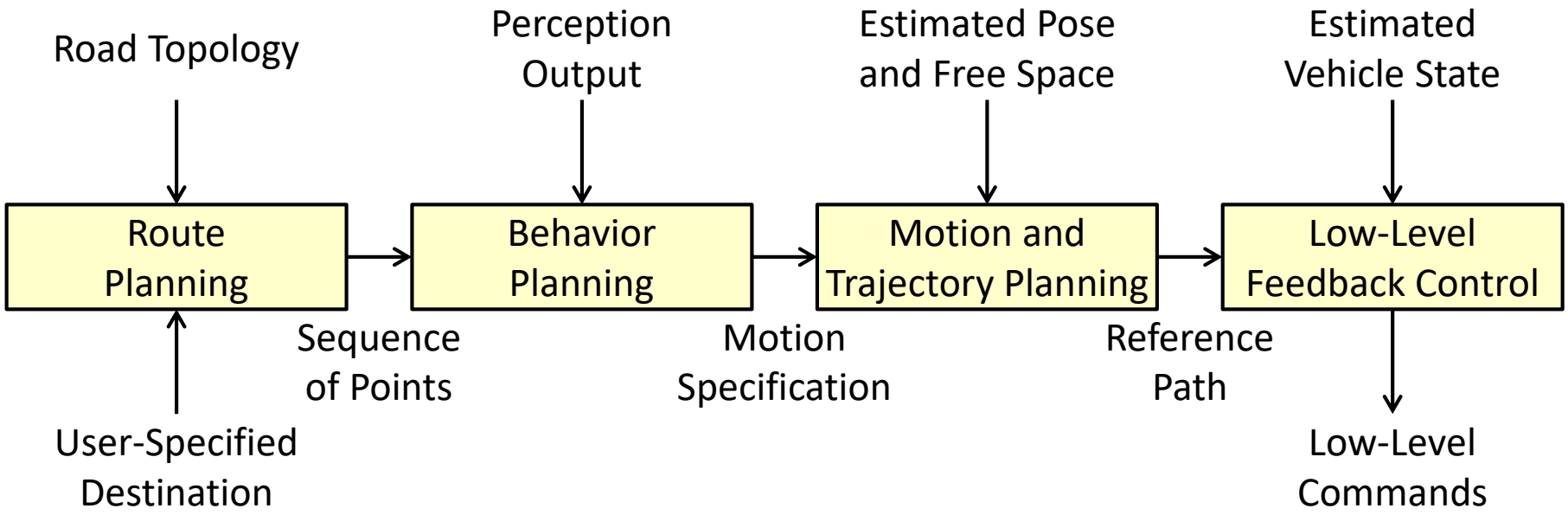






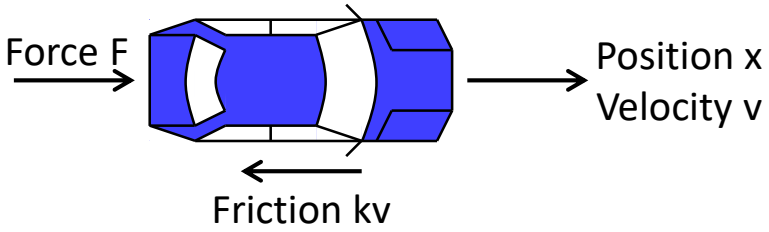
Outline

- ❑ Route Planning
- ❑ Behavioral Planning
- ❑ Motion and Trajectory Planning
- ❑ **Low-Level Feedback Control**



Physics and Differential Equations (1/3)

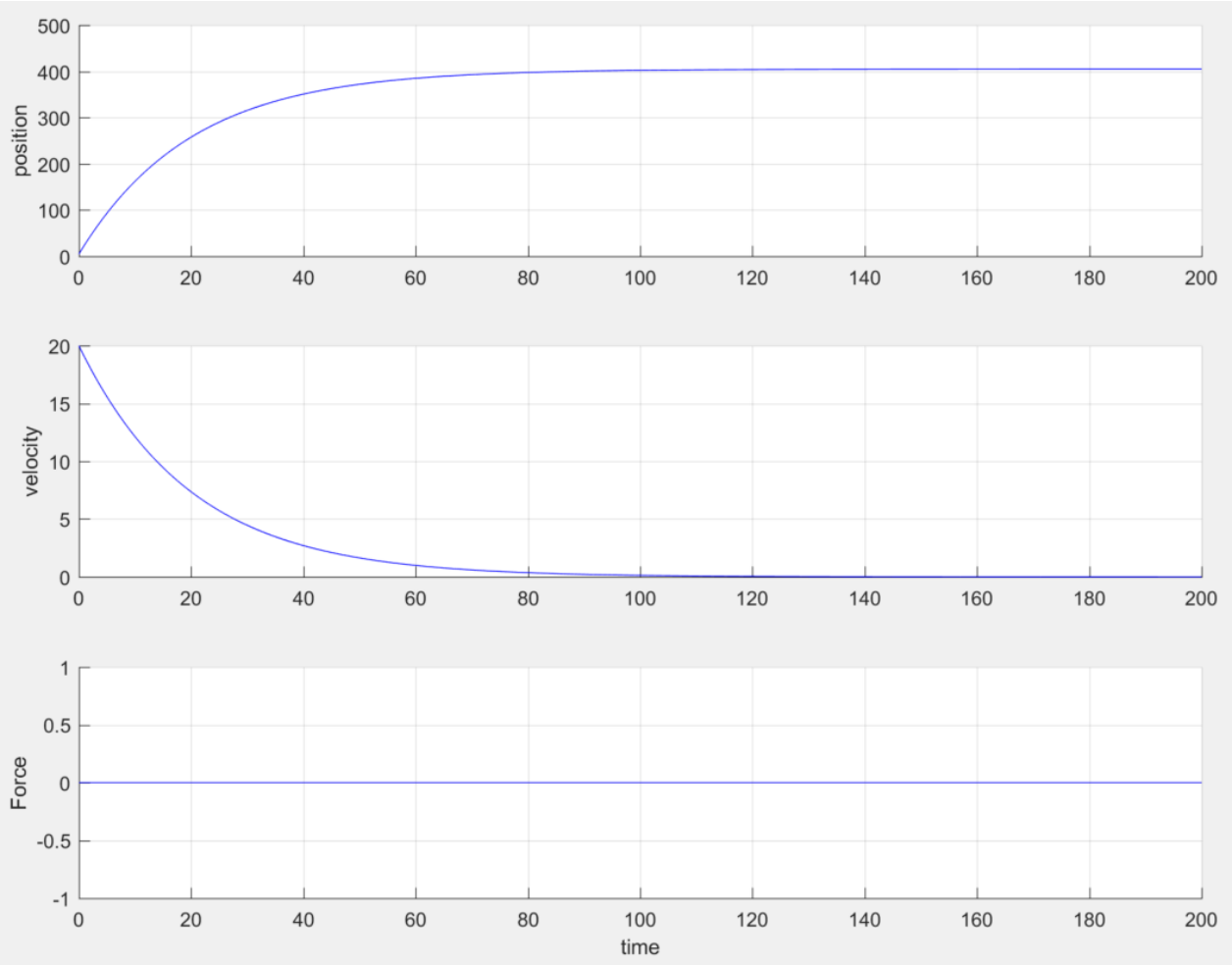
- Newton's law of motion
  - $F - kv = m\,x''$
  - $v = x'$



- Given an initial state (x_0,v_0) and an input signal F(t), the **execution** of the system is defined by
  - State-trajectories x(t) and v(t)
    - $x(0) = x_0$
    - $v(0) = v_0$
    - $v(t) = x'(t)$
    - $v'(t) = (F(t) - kv(t)) / m$

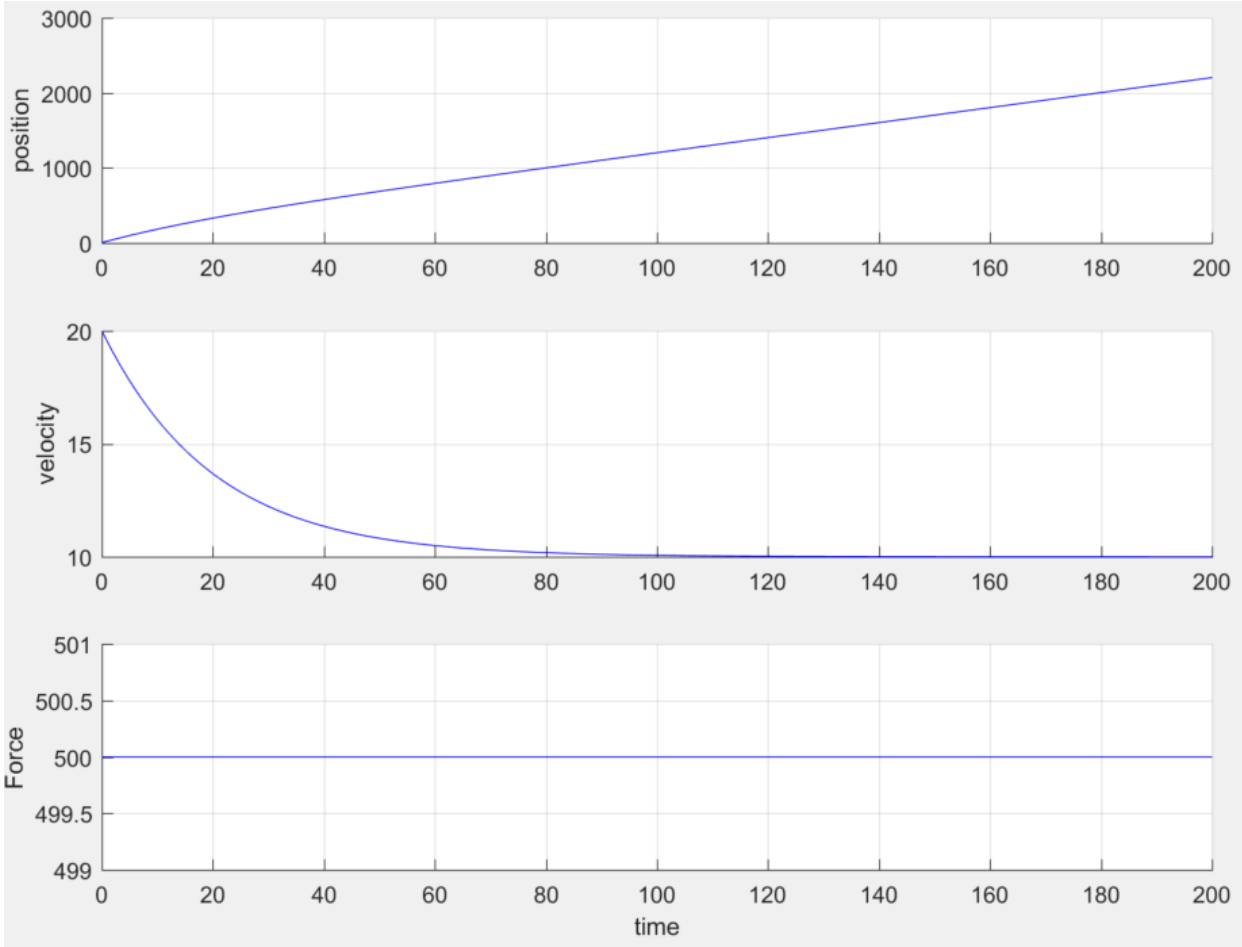
Physics and Differential Equations (2/3)

- Example: $x_0 = 5$, $v_0 = 20$, $F(t) = 0$



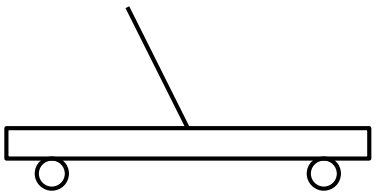
Physics and Differential Equations (3/3)

- Example: $x_0 = 5$, $v_0 = 20$, $F(t) = 500$



Properties of Execution

- Existence: exist at least one solution?
- Uniqueness: exist exactly one solution?
- Stability
  - The ability of a system to return to a quiescent state after perturbation
  - Almost always a desirable property for a system design
- Fundamental problem in control: design a controller to **stabilize** a system
  - Problem: keep an "Inverted Pendulum" stable on a moving cart
  - Solution: move cart in the direction as the pendulum's falling direction
    - Design a controller to stabilize the system by computing the velocity and direction for the cart travel

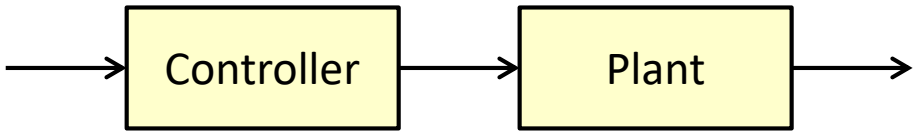


Controller Design

- Design a controller to ensure desired objectives of the overall physical system
- Mechanical, hydraulic, or electro-mechanical control
  - First papers from 1860s
- Digital control
  - Began roughly in the 1960s
  - Use of software and computers/microcontrollers to perform control tasks

Open-Loop (Feed-Forward) Control

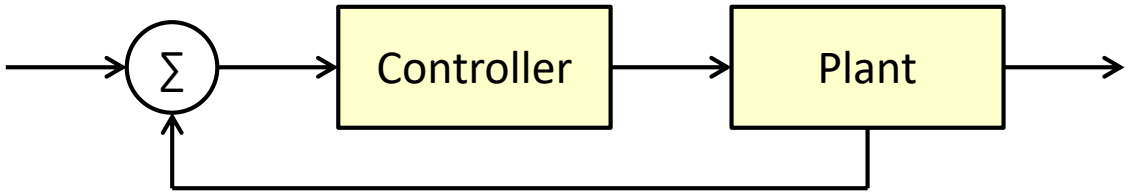
- Control action does not depend on plant output



- Example
  - Older-style air-conditioning in a vehicle
    - Input: user's temperature dial setting
    - Vehicle's response: change the temperature of air-conditioning
  - The vehicle does not actually measure temperature in the cabin to adjust the temperature and air-flow
- Advantage
  - Cheaper (no sensor required)
- Disadvantage
  - Poor quality of control, without human intervention

Closed-Loop (Feedback) Control

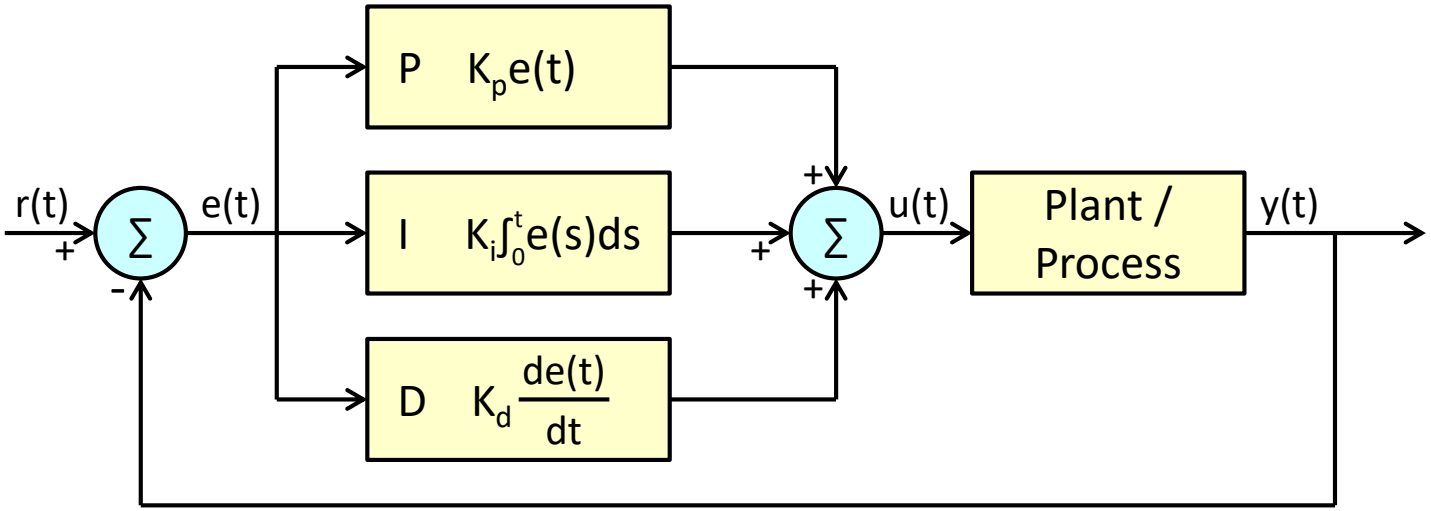
- Control action does depend on plant output
  - Better response to variations in disturbances
  - Performance depends on
    - How well outputs can be sensed
    - How quickly controller can track changes in output



- Advantage
  - Better quality of control (automatic)
- Disadvantage
  - More expensive

PID Controllers (1/2)

- Proportional Integral Derivative (PID) controllers are the most widely-used and most prevalent in practice
  - Many other controllers use systematic use of linear systems theory

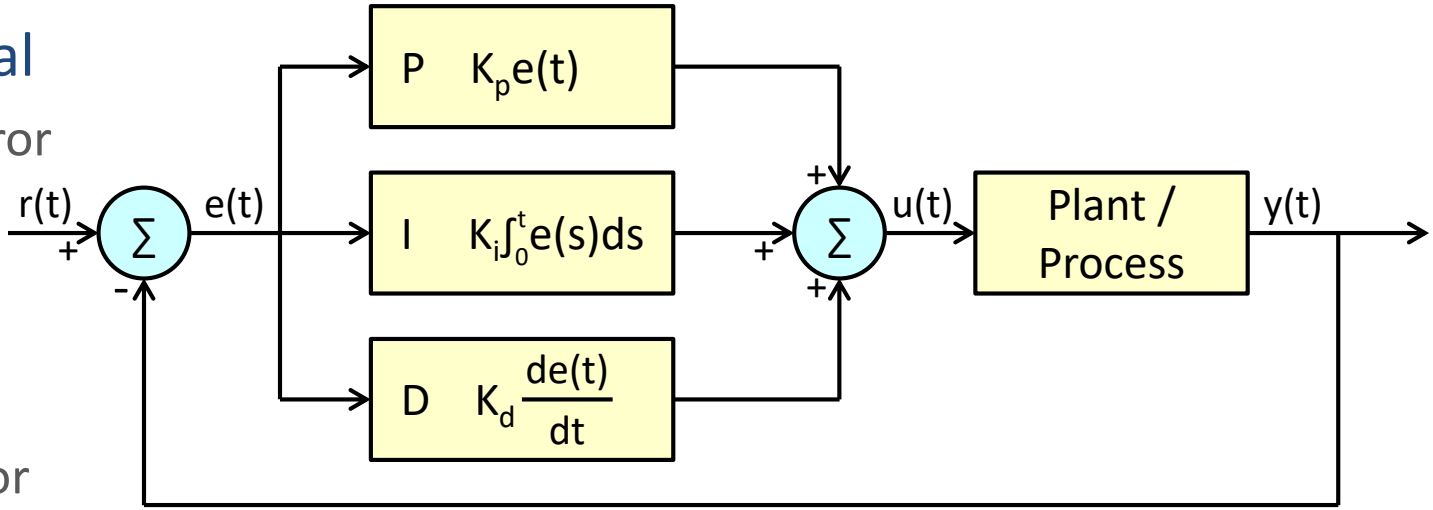


r(t): setpoint (desired value)
e(t): error value
u(t): control variable
y(t): measured value

https://en.wikipedia.org/wiki/PID_controller

PID Controllers (2/2)

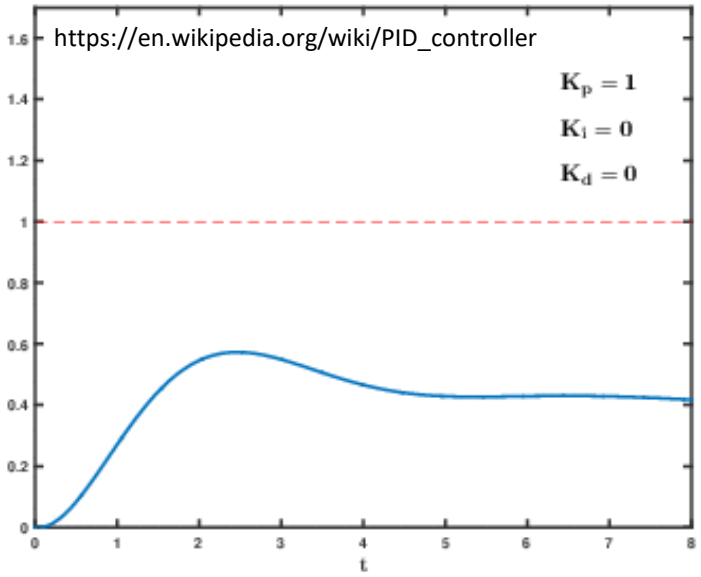
- Proportional
  - Current error
- Integral
  - Past error
- Derivative
  - Future error



Effects of *increasing* a parameter independently

Parameter	Rise time	Overshoot	Settling time	Steady-state error	Stability
K_p	Decrease	Increase	Small change	Decrease	Degrade
K_i	Decrease	Increase	Increase	Eliminate	Degrade
K_d	Minor change	Decrease	Decrease	No effect in theory	Improve if K_d small

https://en.wikipedia.org/wiki/PID_controller



Q&A