Secure Autonomous and Cyber-Physical Systems

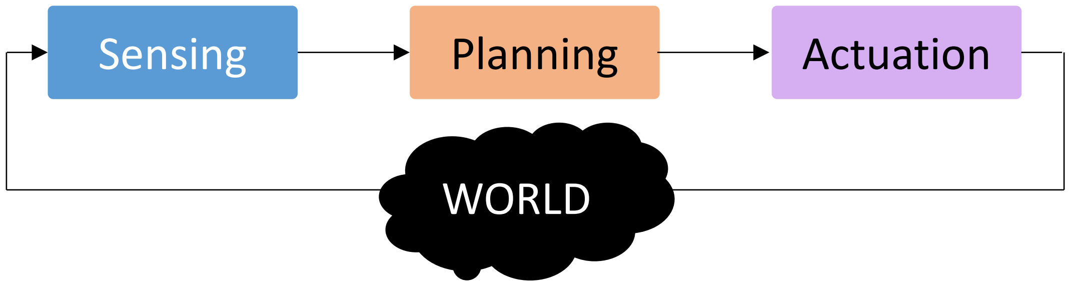
CS 599 001/ECE 599 004

Winter 2022

Prof. Sibin Mohan

https://bit.ly/secureauto2022

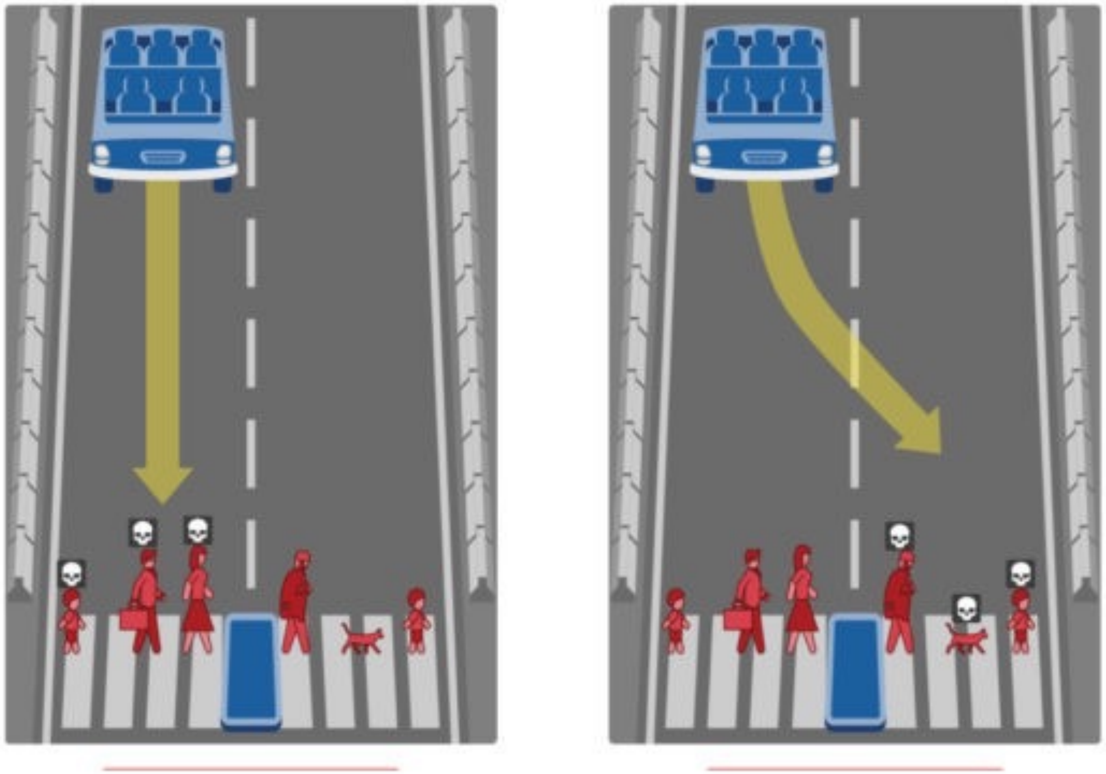
Sensing, Planning, Actuation



Sensors Abstraction	Sensor Fusion	World Model	Behavior Generation	Planning	Vehicle Control	Actuation
RADAR LIDAR Cameras GPS V2X Internal St	Object Det Global Pos Local Pos Vehicle State	Internal Model External Model Data Sinks Bird's Eye View	HMI Route i/p Global Routing Behavior Gen. Behavior Select	Path Planning Monitoring Lane Keeping Lane Change Collision Avoid HMI Driver Alert	Lateral Longitudinal Trajectory Intention Comm.	Steering Brake Throttle Transmission

Path Planning for Autonomous Systems

- Decision making
- Predictions
  - Other cars
  - Pedestrians
  - Traffic signals
- Routes must be
  - Safe
  - Convenient
  - Economically beneficial



Path Planning | Terminology

- **Path**: continuous sequence of configurations
  - starting/ending with boundary configurations
- **Path planning**: find a geometric path from initial to given config
  - Each configuration and state on path is feasible
- **Maneuver**: high-level characteristic of vehicle’s motion
  - Encompasses position and speed of vehicle on road
  - E.g.: going straight, changing lanes, turning, right, overtaking, etc.
- **Maneuver planning**: take best high-level decision for vehicle
  - Take into account path specified by planning algorithm
- **Trajectory**: sequence of states visited by vehicle
  - Parameterized by time and velocity
- **Trajectory planning**: real-time planning of vehicle’s moves
  - From one feasible state to the next, satisfied by car’s kinematics

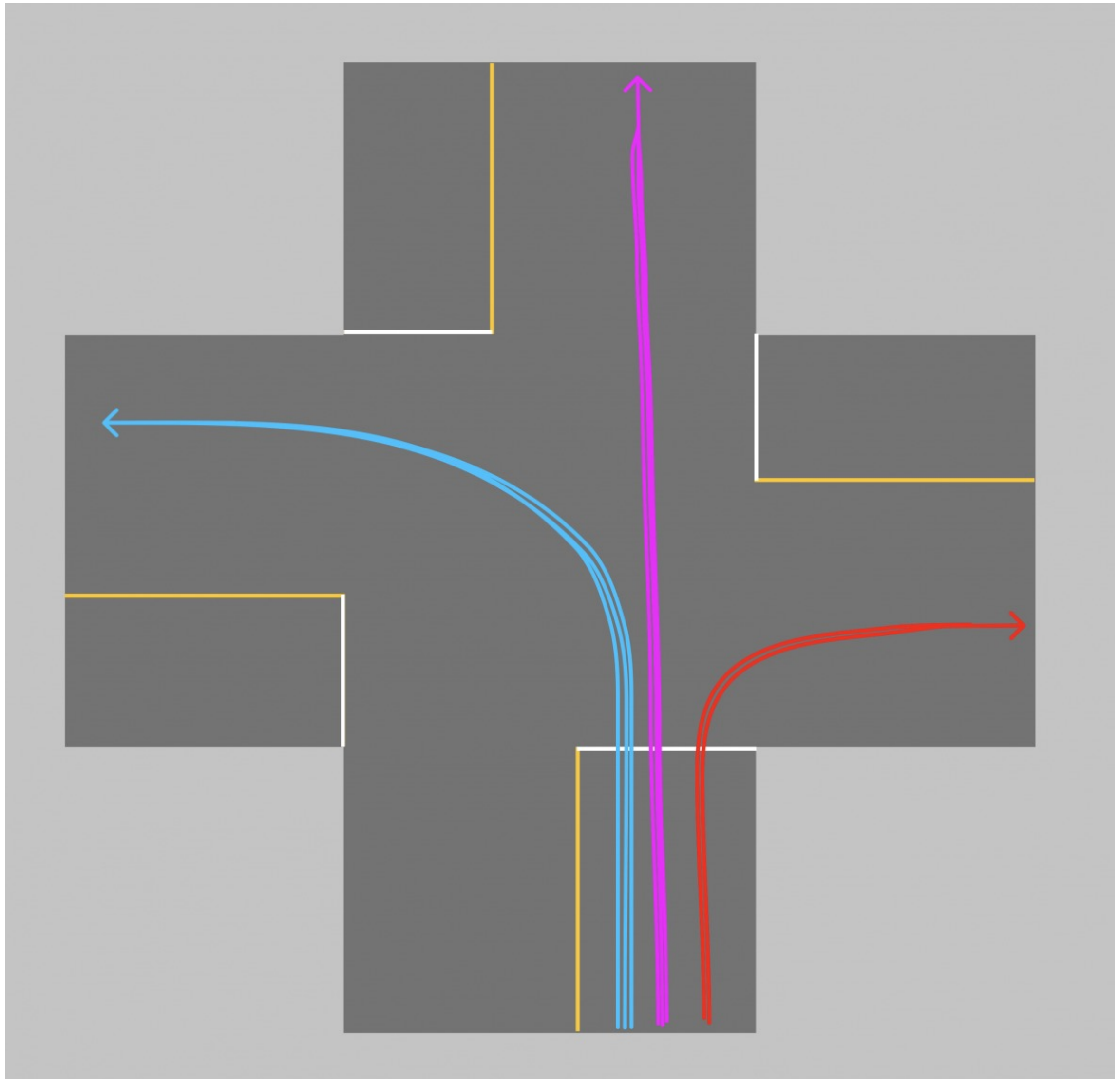
Path Planning | Predictions

- Predict what each element of environment will do
  - A few seconds in the future
- E.g.: pedestrian will move (and direction), traffic sign remains still

- Multiple Approaches
  1. **All possible trajectories** for each possible situation
  2. **Machine learning** to establish similarity with training data
  3. **Model-based** approach

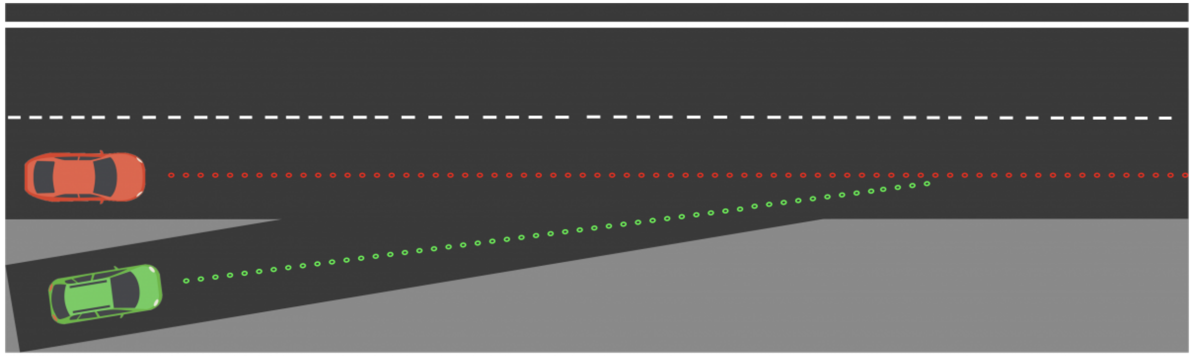
Path Planning | Machine-Learning

- **Training** phase:
  - gather massive history of vehicles and paths
  - Hundreds of vehicles, different actions at intersection
- **Unsupervised** learning
  - Clustering algorithms
  - Each cluster a typical trajectory for vehicle
- More driving leads to more data
  - Past behavior can affect current decisions



Path Planning | Model-based

Imagine **possible choices** for vehicle



- Implements **feasibility** of trajectory
- Eliminates **Impossible Behaviors**
- Focus on what's **possible**, not on past

Four choices for the other car:

1. speed up
2. slow down
3. constant speed
4. change lanes

Each has a **probability** that **changes with observations**

Sensors work in real-time

Path Planning | Decision Making

- With estimate of future environment, make a **decision**
  - Brake if obstacle detected?
  - Accelerate or change lanes?
- Environment Classification
  - Highway vs parking lot?
  - Safety, feasibility, efficiency, legality, passenger comfort
- **Finite State Machines**

Path Planning | Finite State Machines

Define states of a car

E.g., on highway, options:

stay in lane, change to left lane, overtake a car

Cost functions define choice of state

Computed (independently) for each possible scenario

Added up ➔ lowest cost wins

How to **define** cost?

Path Planning | Finite State Machines | Costs

feasibility cost

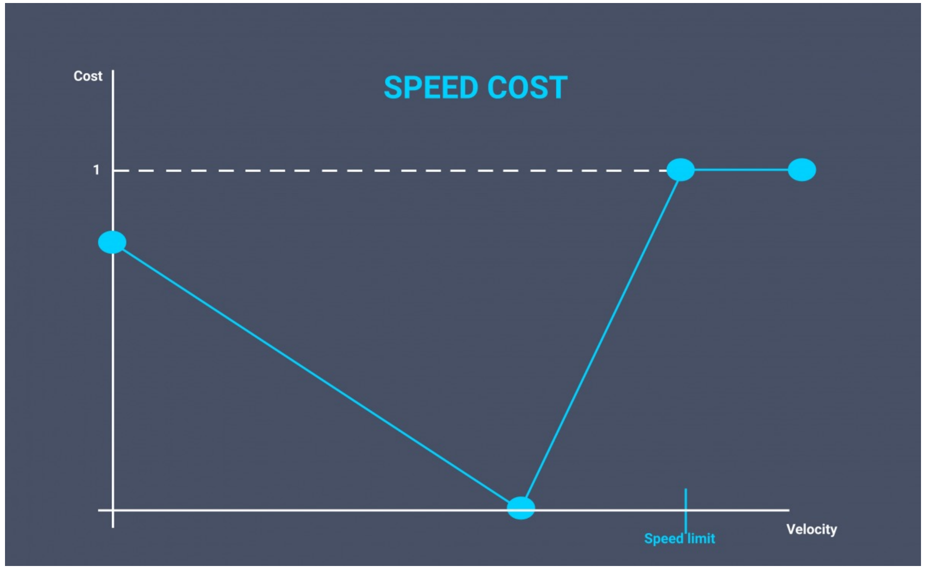
security cost

legal cost

speed cost

$$\text{Total Cost} = F_c * 5 + S_c * 4 + L_c * 3 + C_C * 2 + S_c * 1$$

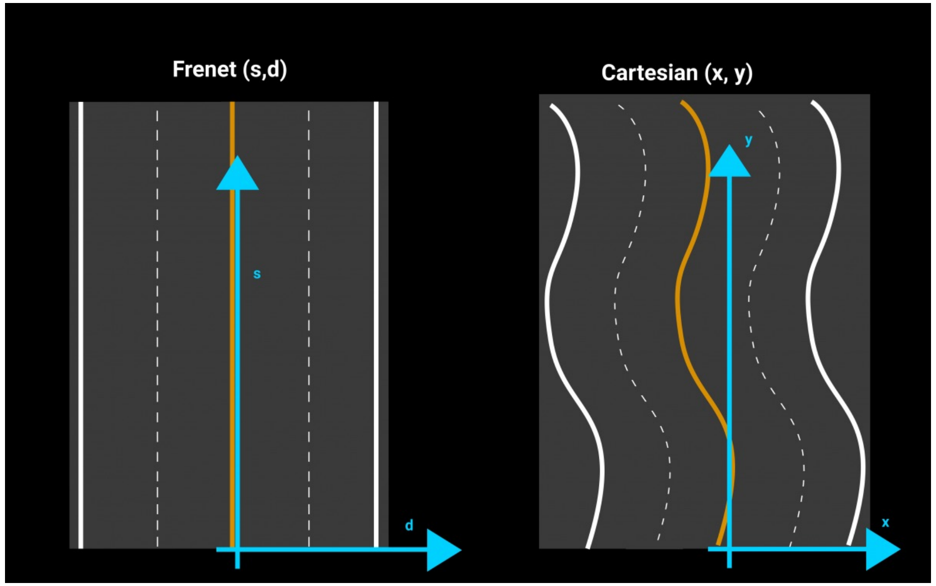
comfort cost



Path Planning | Trajectory Generation

- Typically use **a Frenet coordinate system**, not Cartesian

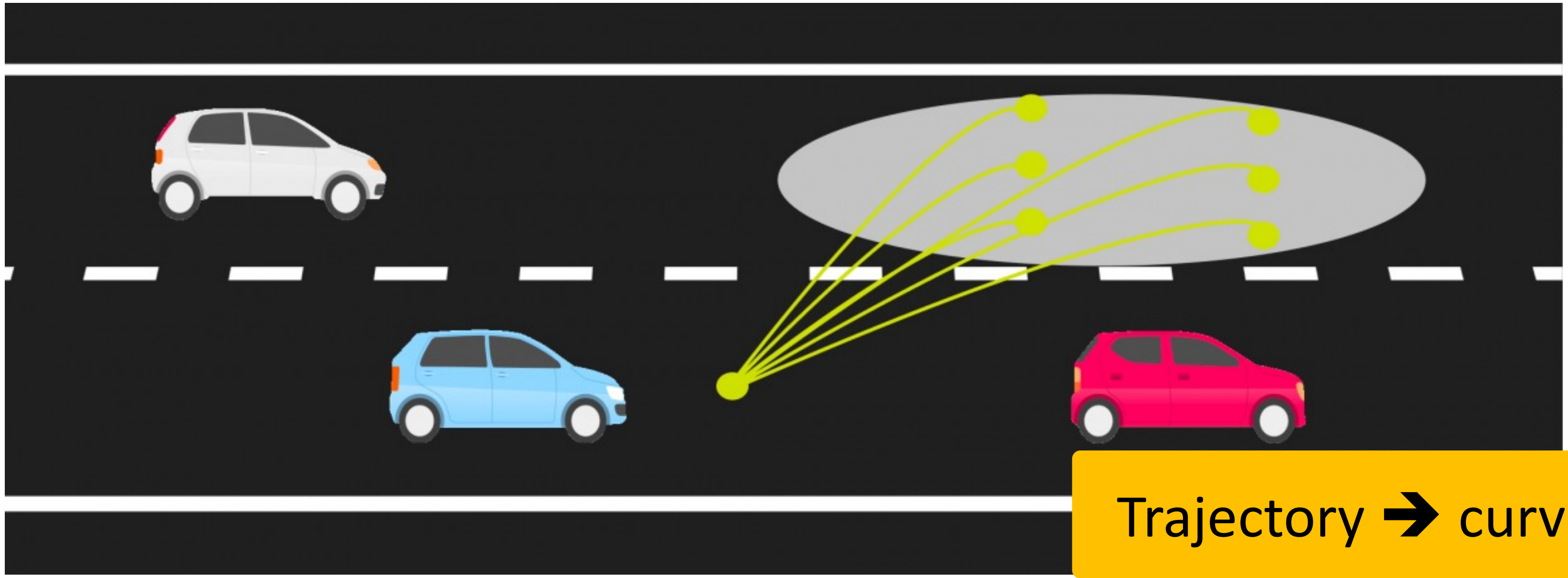
advance relative to track ← (s, d) → distance to center of lane



- After decision (e.g., overtake), algorithm **generates several trajectories**

Path Planning | Trajectories

- Choose best one based on criteria
  - Feasibility, safety, legality, efficiency, comfort



Trajectory ➔ curve through waypoints

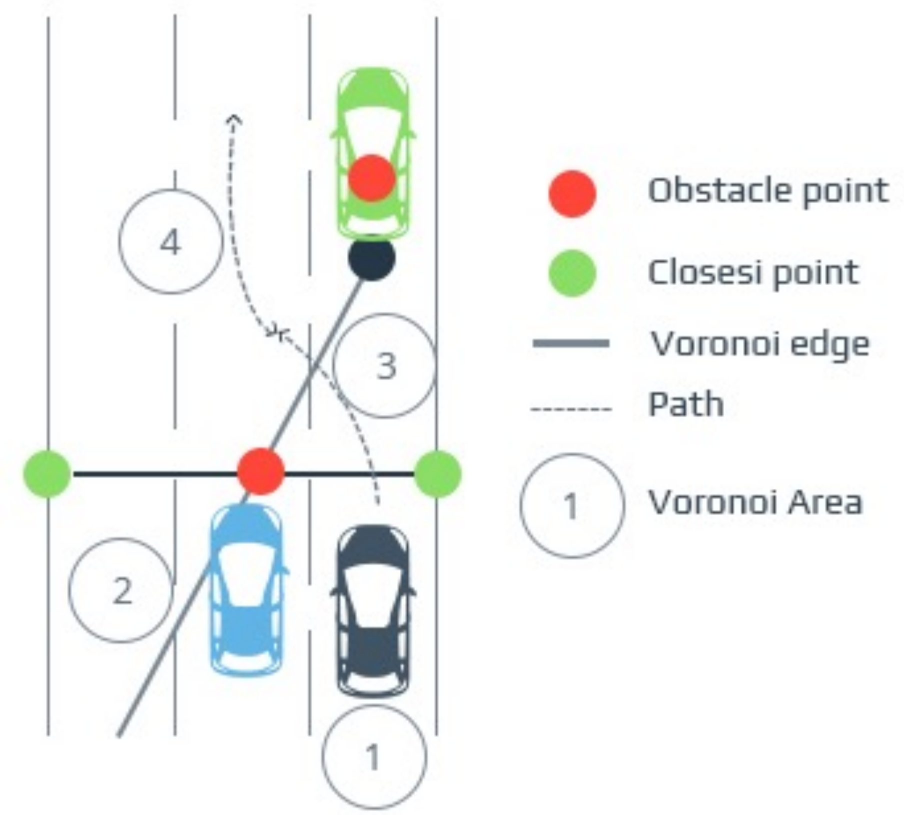
- **Polynomial** that passes through waypoints
  - Waypoints ➔ longitudinal distance, lateral distance, moment of passing

Path Planning Algorithms

Higher-order planning

Voronoi Diagrams

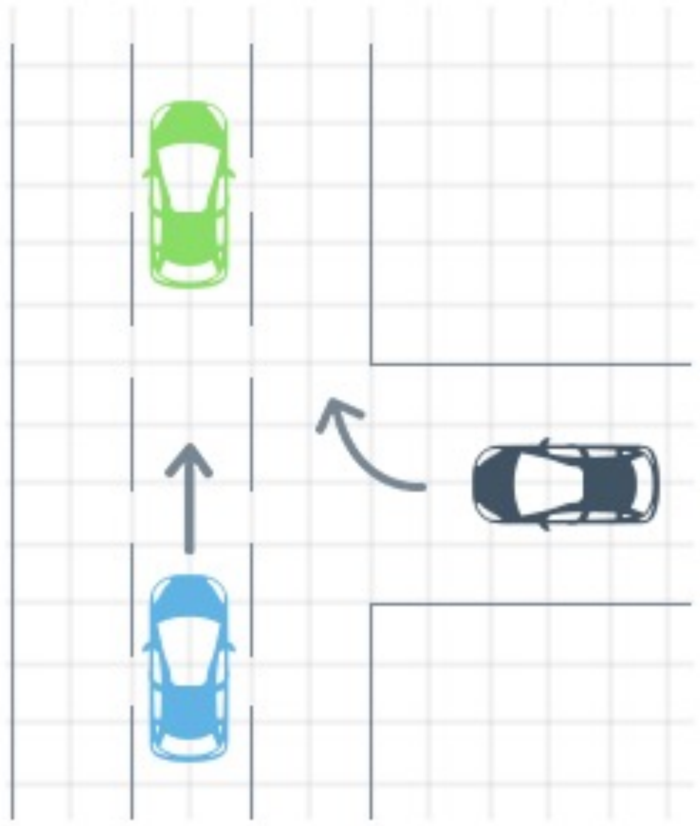
- Generates paths that **maximize distance** between vehicle and surrounding obstacles



Path Planning Algorithms

Occupancy Grid

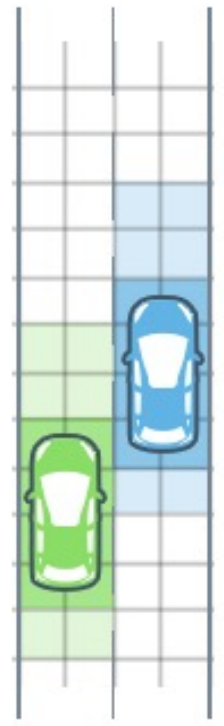
- Similar to Voronoi diagram
- Risk and feasibility computed using obstacles and lane/road boundaries



Path Planning Algorithms

Cost Maps

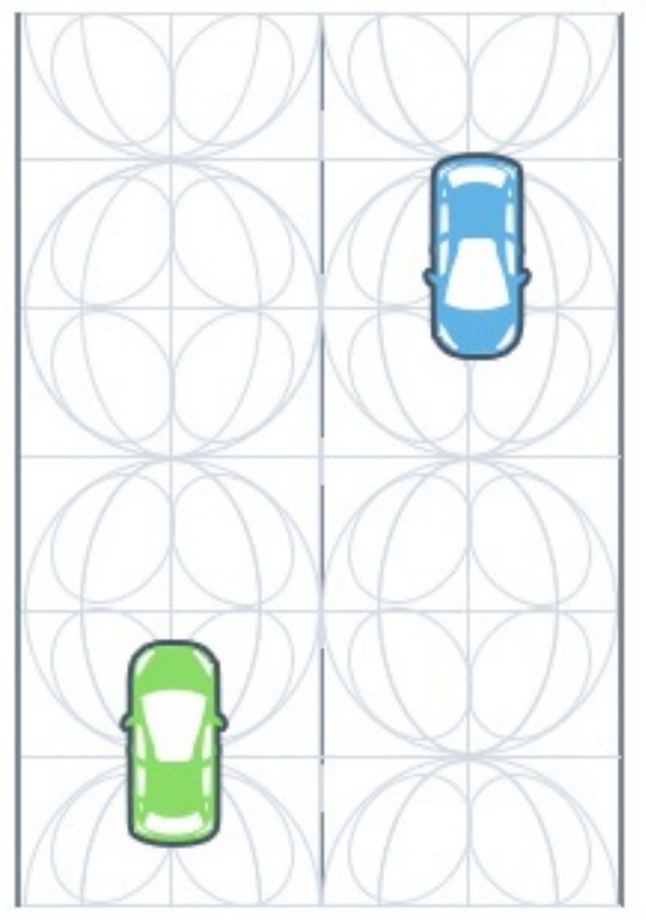
- Similar to occupancy grid
- Higher cost cells get more intense representation on map



Path Planning Algorithms

State Lattices

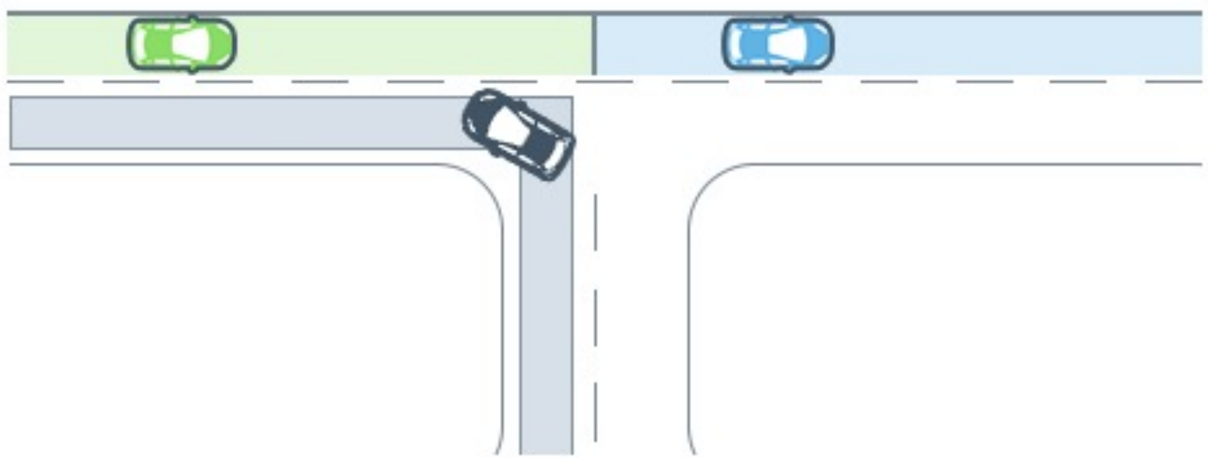
- Generalization of grids
- Grids built using repetition of rectangles or squares that discretize continuous space
- **Lattices** are constructed using regularly repeating primitive paths that connect possible states



Path Planning Algorithms

Driving corridors

- Recreates continuous collision-free spaces bounded by lanes and other obstacles
- Use data from maps and SLAM



Path Planning | Higher-order Decision Making

- Moral Machines
- Self driving car mindmap

Security Classification

How do you define **security**?

Security Classification | Attacks

Passive (stealing data) vs active (causing physical harm)

Sensors hacks

- tags on stop signs, reflective paint on vehicles
- Jamming RADARs
- Physical damage to sensor

Attacks by other vehicles/V2X system hacks

Software hacks

- Operating system
- Entire software stack: Kalman filter, planning algorithms, vision algorithms, sensor fusion, etc.

Attacks on actuation subsystem

- Prevent actuation commands from executing
- Change commands in flight

Denial of service

- Cause wear and tear on devices, e.g., engine/brakes

Security Classification | Attacks

Passive (stealing data) vs active (causing physical harm)

Sensors hacks

- tags on stop signs, reflective paint on vehicles
- Jamming RADARs
- Physical damage to sensor

Malware injection is one way to enter system

Attacks by other vehicles/V2X system hacks

Software hacks

- Operating system
- Entire software stack: Kalman filter, planning algorithms, vision algorithms, sensor fusion, etc.

Attacks on actuation subsystem

- Prevent actuation commands from executing
- Change commands in flight

Denial of service

- Cause wear and tear on devices, e.g., engine/brakes

Security Classification | Attacks [contd.]

Hardware

- Trojan horses in chips
- Hidden backdoors in chips/hardware units

Data attacks

- Maps/other data could be tampered with

Security Classification | Defenses

Passive [only detect/raise alarms] vs active [take action on detection]

- intrusion detection and prevention

Reactive [take action on detection] vs proactive [action without detection]

Use existing (additional) sensors/functions

- To detect anomalous sensor behavior

Warnings

- How to design warnings that will be noticed/heeded?

Security Classification | Defenses [contd.]

Buy cheap chips/supply chains

Few communication end points + encrypt/authenticate

- Other vehicles/traffic signs, sensors (GPS)
- Network updates (OS)
- Entertainment systems
- Computation and memory overheads
- Power consumption

Hardware verification to detect attacks at startup

- Additional chips (ROM)

History of correct behavior → use to check

- Vehicular level, software behavior

Security Classification | Defenses [contd.]

Sensor fusion

- use multiple sensors to verify each other

Use hardware signature

Human input to verify

- humans as failsafe

Security as a first principle

Hide systems design (security through obscurity)

Software security (isolation, VMs, etc.s)

Better systems design/product design

Paper Reading List

- **Three** categories
  - attacks
  - defenses
  - privacy and ethics

Each person picks one paper from each category

- **45 mins** per paper
  - 30 mins presentation
  - 15 mins class discussion → led by speaker
- Two presentations/class

You can use slides from paper authors
Remember to acknowledge!

How to read/critique a paper

Remember to include discussion points for the class!

What is the **problem** being solved?	is the problem of significance?
What **assumptions** are made by the authors?	how realistic are these assumptions?
How does it **compare** with others' work/state-of-the-art?	missing gaps?
What is the **proposed solution**?	architecture/framework/implementation details?
Evaluation setup, experiments, theoretical analysis, **results**?	comprehensive or missing some important evaluations?
Your conclusion about the paper?	would you accept/reject this paper?

References

- Path planning for self driving cars:

https://www.thinkautonomous.ai/blog/?p=path-planning-for-self-driving-cars

- Path planning with some discussion about algorithm classes

https://intellias.com/path-planning-for-autonomous-vehicles-with-hyperloop-option/

- Video that explains the self-driving car mind map:

https://www.thinkautonomous.ai/the-self-driving-car-engineer-mindmap/

- A* search algorithm

https://www.youtube.com/watch?v=ySN5Wnu88nE