Autonomic Software and Systems

CARLA Project

Florian Dejonckheere

24.05.2023

Overview

Goal

Develop an autonomic manager for a self-driving car using the CARLA simulator

Methodology

Use the MAPE-K architecture to design an autonomic system architecture

Result

Autonomic car that can drive itself to the given target, and react to normal and extraordinary situations in traffic

Autonomic crash recovery

Architecture

Game scenario

Set up scene: autonomic vehicle, traffic

Provide waypoints to navigate to

Determine success or failure of test

Debug helpers (HUD)

Autopilot

Encapsulate MAPE-K architecture

Updates at 10 Hz

Monitor

LIDAR sensor

Mounted on top of vehicle

Rotates at 10 Hz

Generates point cloud

Full image every autopilot loop

Proximity sensor

Mounted on hood of vehicle

Depth camera with narrow FOV

Processed in grayscale

Collision sensor

Lane detector sensor

RGB camera

Analyzer

Analyze data

LIDAR sensor

Proximity sensor

Vehicle location

Vehicle velocity

Collision sensor

Infer knowledge

Potential obstacles

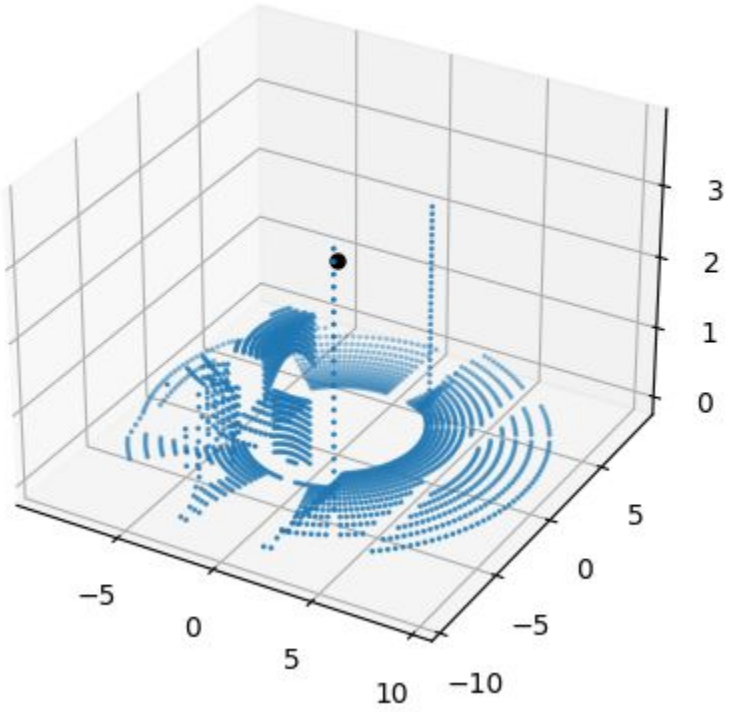
Nearby vehicles

Vehicle location history

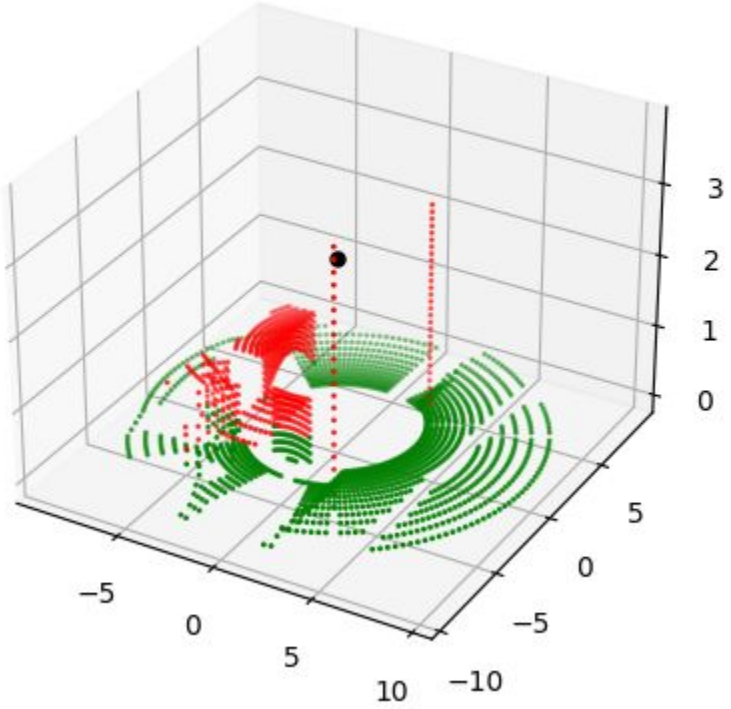
Vehicle speed

System state (driving, crashed, ...)

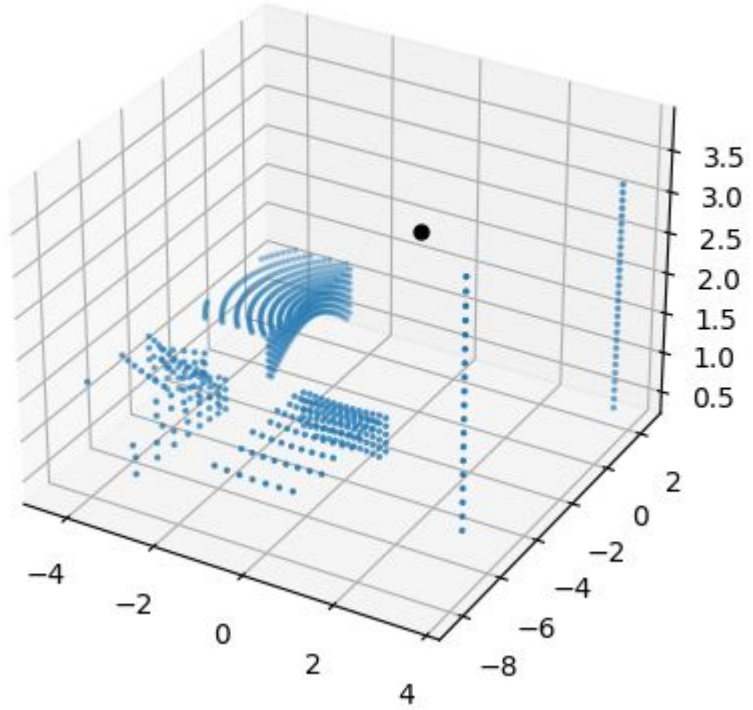
Analyzer



Analyzer

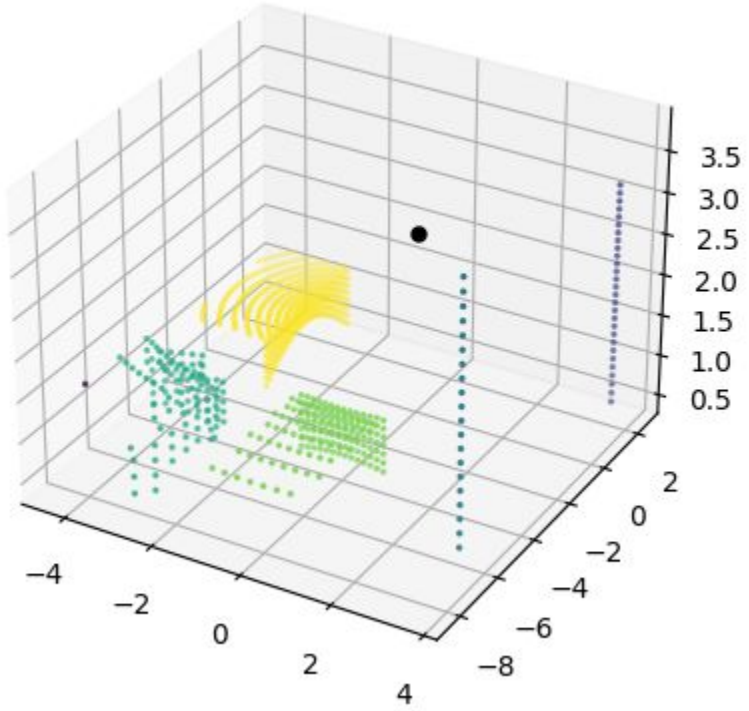


Ground plane detection (RANSAC)

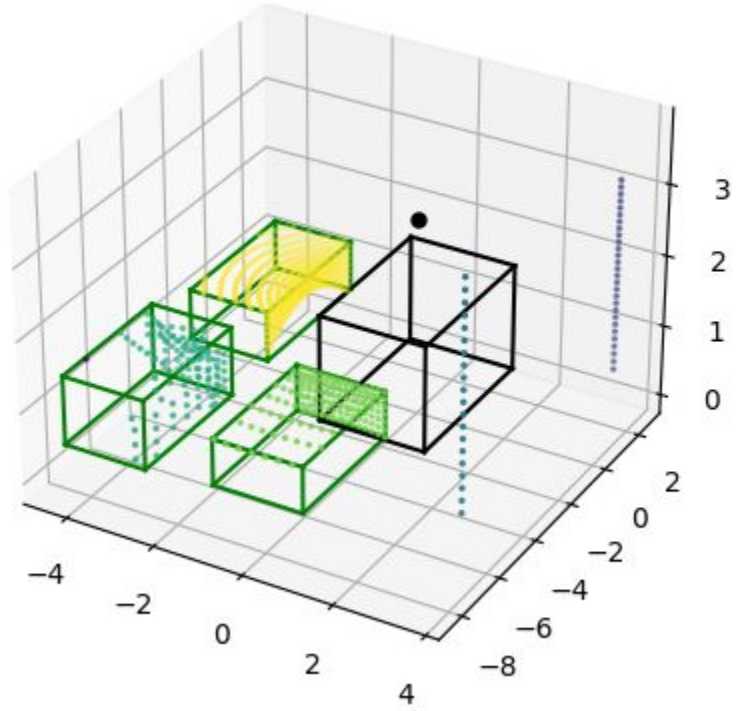


Ground plane removed

Analyzer

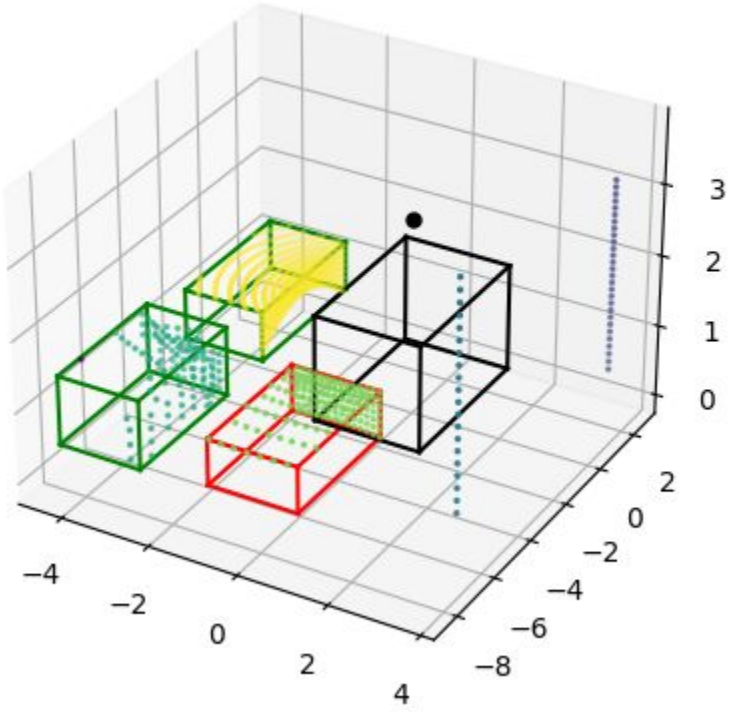


Point cloud clustered (DBSCAN)



Bounding boxes computed

Analyzer



Potential obstacles marked

System state

Idle: initial state

Driving: moving towards destination

Waiting: stopped at traffic light

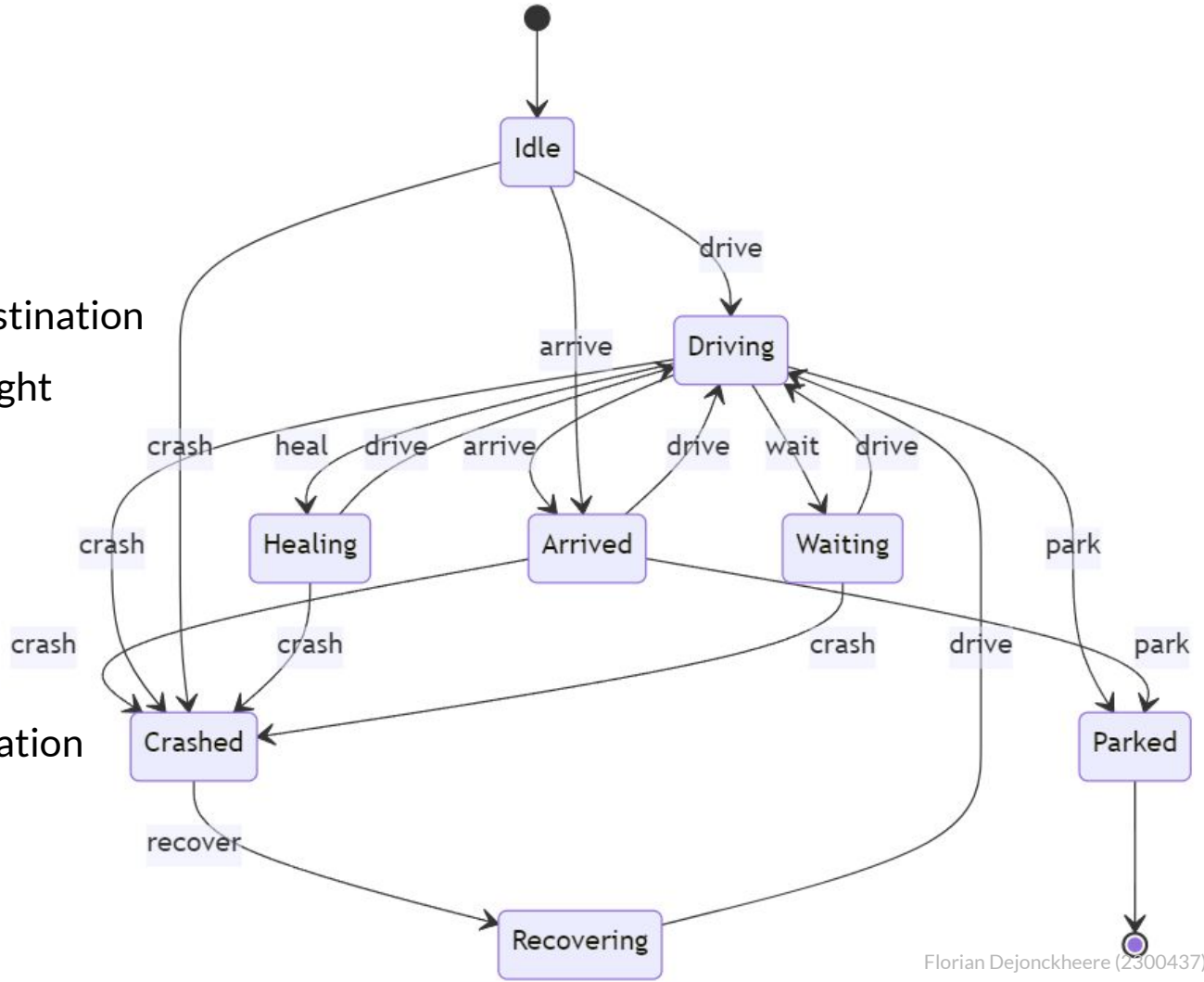
Healing: avoiding collision

Crashed: collision detected

Recovering: crash recovery

Arrived: reached waypoint

Parked: reached final destination



Planner

Execution plan

Planner creates **execution plan** based on system state and knowledge

Execution plan is composed of **goals**, which are composed of **actions**

High-level goals

Drive, **Reverse**, **Stop**, **Park**, **Avoid collision**

Low-level actions

Shift, **Accelerate**, **Brake**, **Emergency brake**, **Steer**, **Swerve**, **Apply handbrake**

Planner

Goals and actions

Vehicle is driving: **Drive** goal: **Shift**, **Accelerate** and **Steer** actions

Vehicle has arrived: **Stop** and **Park** goals: **Brake** and **Handbrake** actions

Vehicle detected obstacle: **Avoid collision** goal: **Swerve** or **Emergency brake** actions

Vehicle is at traffic light: **Stop** goal: **Brake** action

Planner

Crash recovery

Autonomic system has ability to recover from crashes

Swerve or **Emergency brake** to avoid collisions

If a crash occurred, wait a few seconds, then reverse a few meters to previous location

Restart navigation

Planner

Control variables

Output variables **linear combination** of input variables

Input variables: distance, angle, speed, proximity to obstacles, ...

Output variables: throttle, brake, steering

How to define **output** variables in function of **input** variables and **knowledge**?

Fuzzy control system

Fuzzy variables

Crisp value: e.g. 30m away, 47°, 20 km/h

Fuzzy terms: e.g. “far away”, “very close”, “low speed”

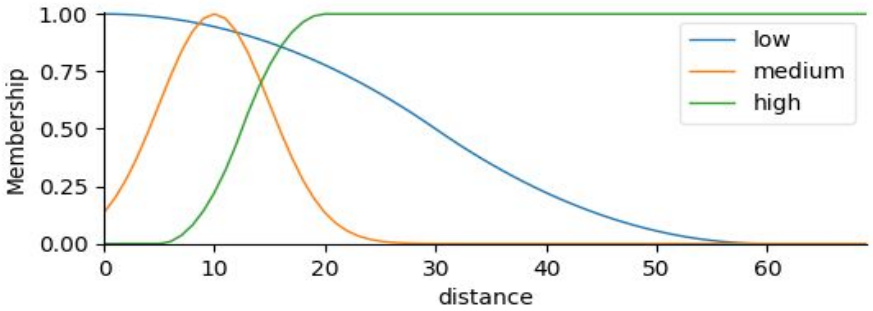
Fuzzy rules

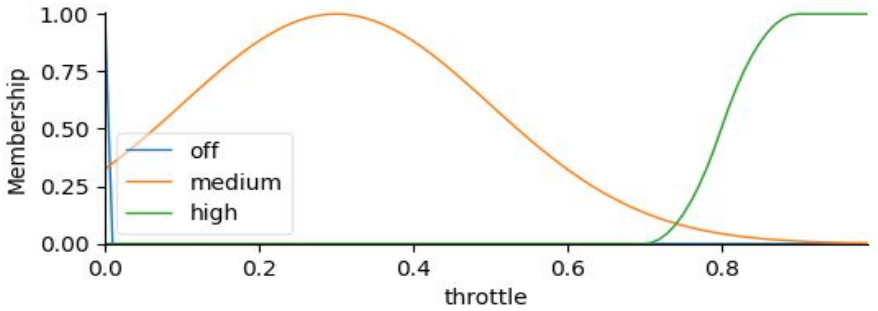
“If the vehicle is **far away** from the target, and **not close** to the speed limit, then **accelerate**”

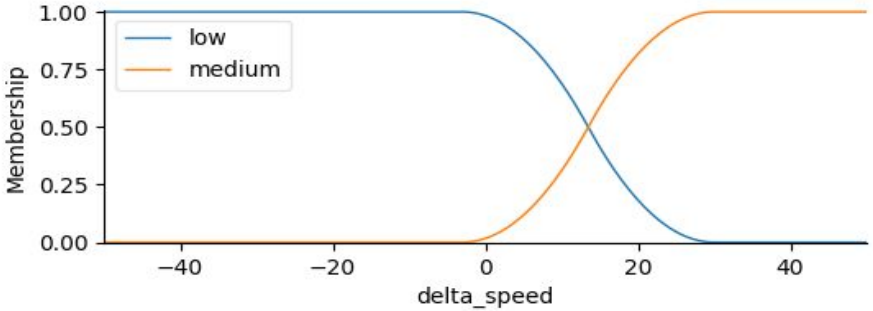
“If the vehicle is **close** to the target or **close** to the speed limit, then don’t **accelerate** and **brake** softly”

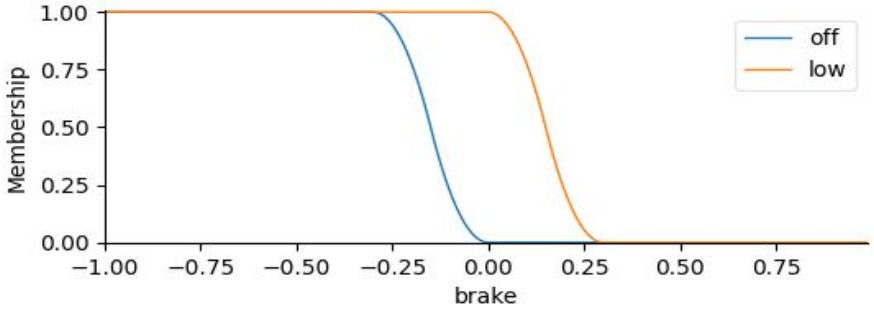
Antecedent **Consequent**

Membership functions

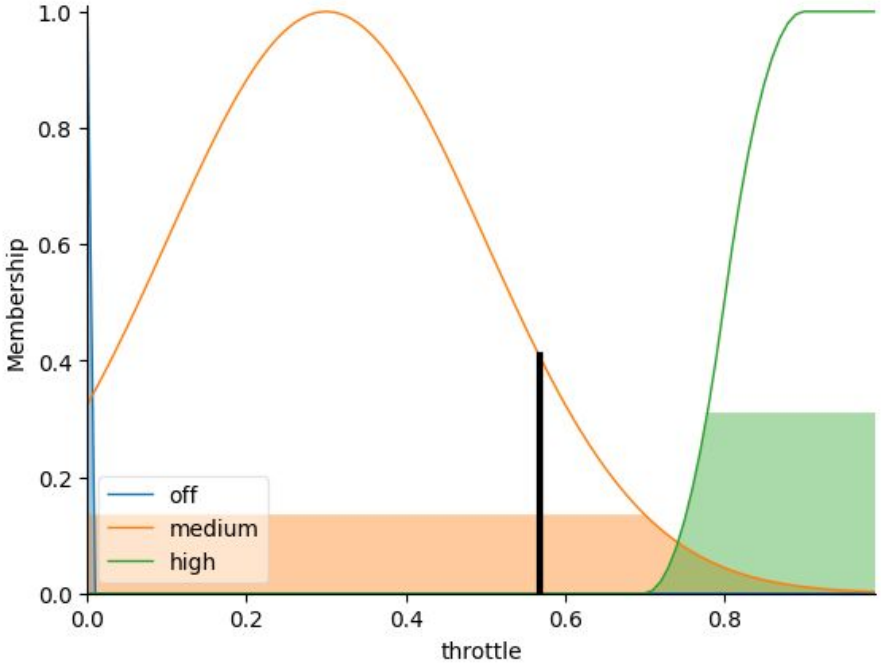


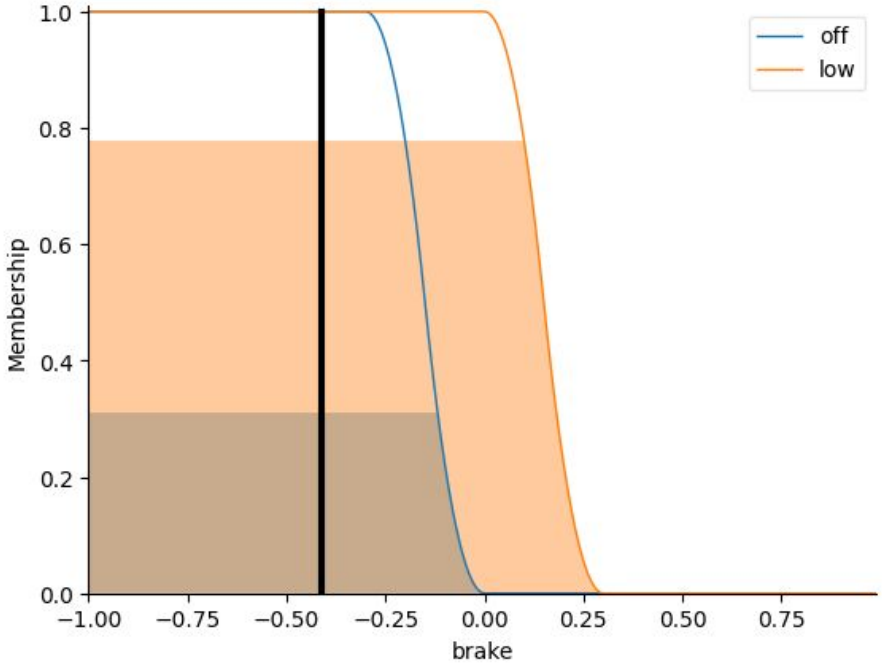






Output






```
distance=20, delta_speed=10 => throttle=0.57, brake=-0.41
```

Planner

Navigator

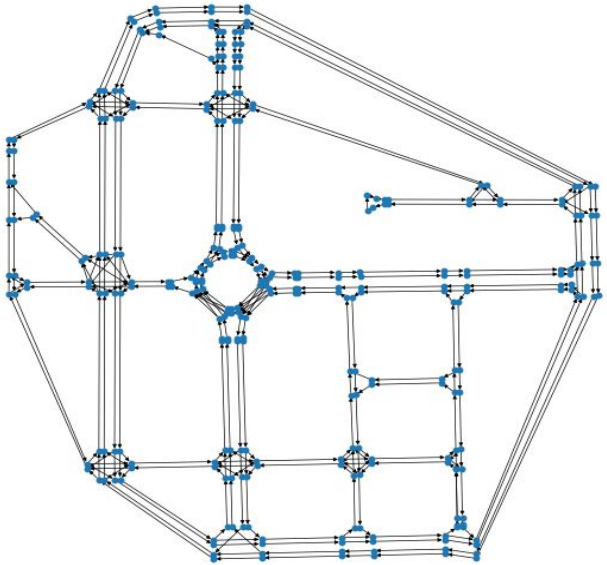
Topological path: based on OpenDRIVE waypoints

- Waypoints on beginning and end of roads (50-200m)
- Find shortest path using Dijkstra's algorithm
- Good for high-level movements

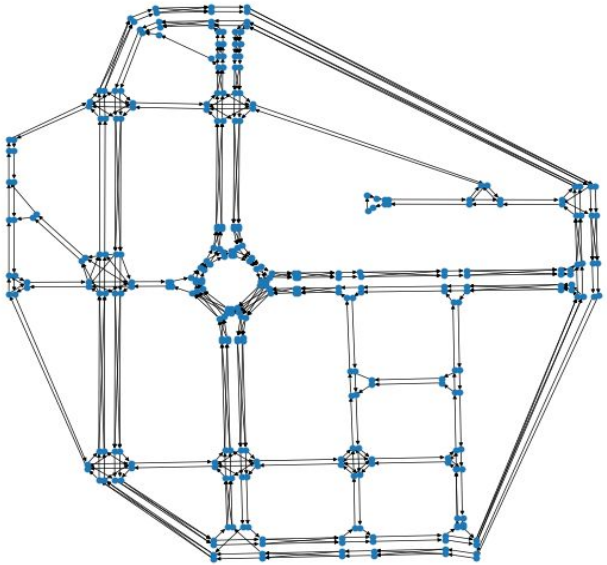
Detailed path:

- Find exact path using local waypoints (every ~2m)
- Good for small movements

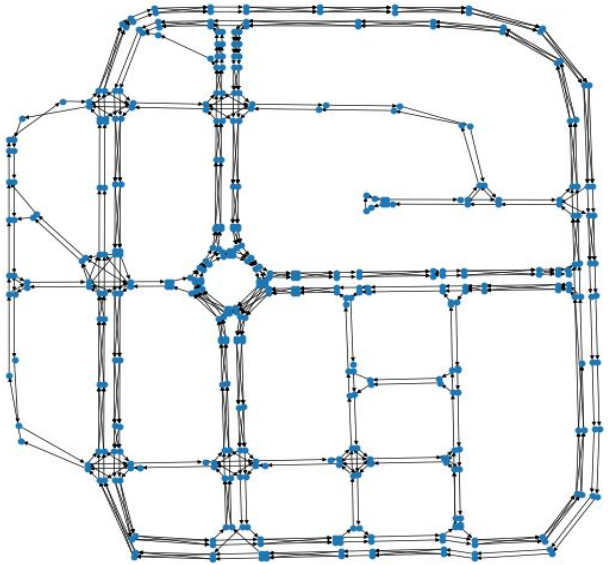
Planner: weighted digraph



309 nodes, 396 edges

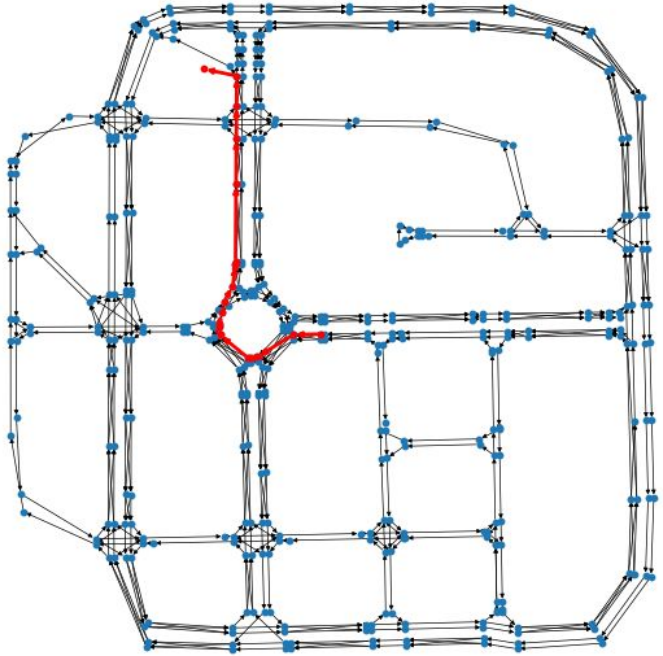


309 nodes, 492 edges

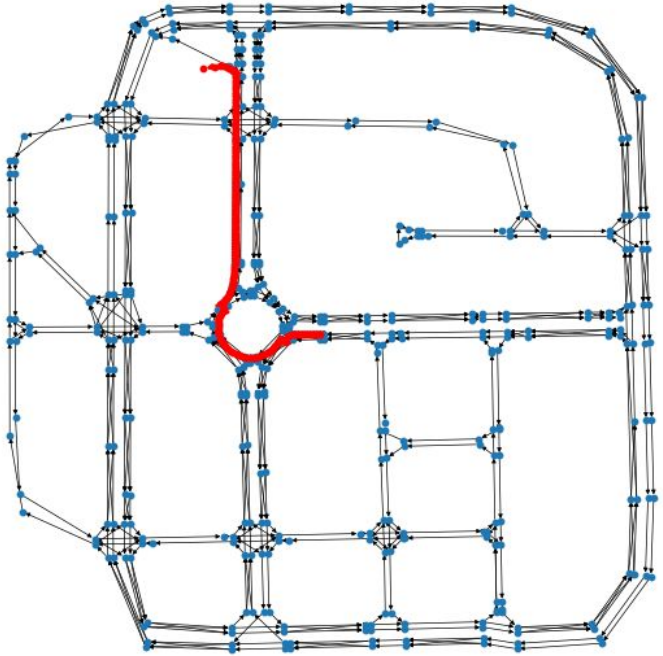


410 nodes, 655 edges

Planner: shortest path



18 nodes, 17 edges



131 nodes, 130 edges

Executor

Apply **goals** and **actions** in execution plan

Use of PID controller to smoothen movements (throttle, brake, steering)

- Proportional: fuzzy control system
- Integral/Derivative: history of control values
- Not for **emergency** actions: Emergency brake, Swerve

Knowledge

Raw **sensor** data

Information about vehicle **environment**

System **state**

Demo

Florian Dejonckheere (2300437)

Questions?