Planning Data Structure for Autoware

Angelo Mastroberardino

07-11-2019
1. Our goal in planning
2. Data for Global planner
3. Data for Lane planner
In the Maps Working Group #7, this design came up

How to go from London to Birmingham?

Simplified Node/Edges → A Directed Graph

Definitions
- **lane**: segment of a road
- **road**: path from a specific poi to another (i.e., junction)
- **node**: group of 1+ poi
- **edge**: group of 1+ lanes

**Global Planner**

Nodes/Edges (simplifying drivable lanes)

**Lane Planner**

Drivable paths (changing lane, turning left/right, exiting a junction…)

**Local Planner**

Trajectories (avoiding obstacles, overtaking vehicles, stop/pass according to the traffic rules)
Our goal in planning

To achieve that, we need **two data structure** capable of referencing each other to **zoom-in** and **zoom-out** the level of detail the car needs.
Data for Global planner

**RoadGraph**
- id(): string
- nodes(): map<node_id, node>
- edges(): map<edge_id, edge>
- path(node_id, node_id): RoadPath

**Edge**
- id(): string
- length(): double
- direction(): string
- nodein(): node_id
- nodeout(): node_id
- reflanes(): list<lane_id>
- nearpois(): list<node_id>

**Node**
- id(): string
- type(): string
- inputs(): list<edge_id>
- outputs(): list<edge_id>
- refpoi(): list<obj_id>
- nextnodes(): list<id>

**RoadPath**
- length(): double
- edges(): list<edge_id>
- refgraph: RoadGraph

---

Type: “junction”, “parking space”, “entry/exit” …

obj_id: reference to id of an object in the Lane/Local Data

lane_id: reference to id of a Lane in the Lane Data

direction: ONE_WAY | TWO_WAY
Data for Lane planner

Just an example (Autoware already has its data structure)

LaneGraph
+ id(): string
+ lanes(): map<lane_id, lane>
+ path(lane_id, lane_id): LanePath

Lane
+ id(): string
+ type(): string
+ prev(): lane_id
+ next(): lane_id
+ borders(): BorderData
+ edgeid(): edge_id
+ nodeid(): node_id | null

- data: LaneData

LaneData
+ speed: double
+ parking: list<parking_id>
+ signals: list<signal_id>
+ obstacles: list<obj_id>

BorderData
+ left: LineString
+ left_type: border_id
+ right: LineString
+ right_type: border_id
+ mid: LineString

LanePath
+ length(): double
+ lanes(): list<lane_id>
- refgraph: LaneGraph

Parking: there should be a data structure holding them (they hold a node_id)

Signals: map of known types (traffic light, crossings …)

Objects: map of known objects on the road (speed bumps …) or Landmarks (they hold a node_id)

Border: map of known types (dashes, straight line, …)