OpenPlanner

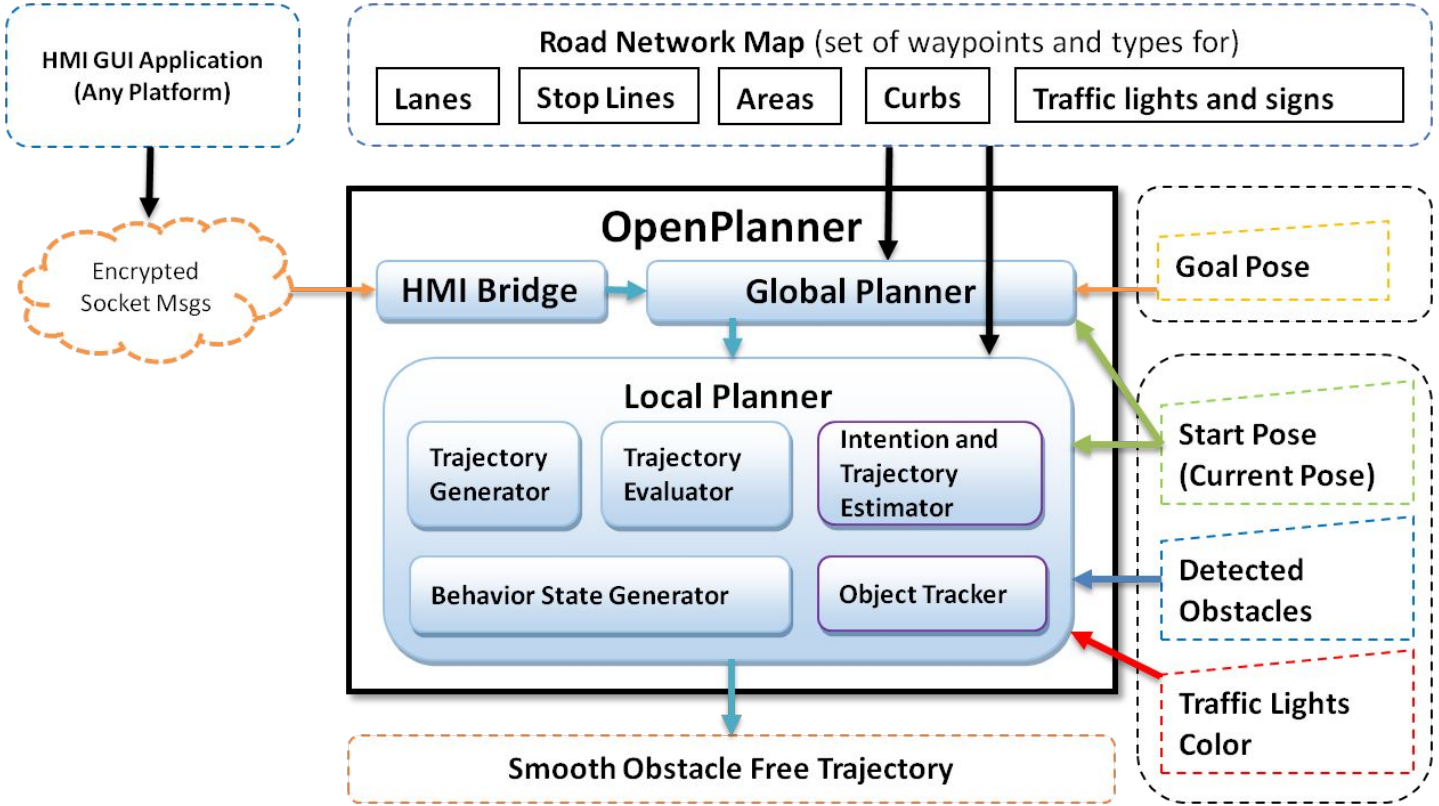
Future Developments and Success Stories

Dr. Hatem Darweesh
Researcher, Takeda-Lab
Graduate School of Informatics, Nagoya University
Founder of ZATiTECH

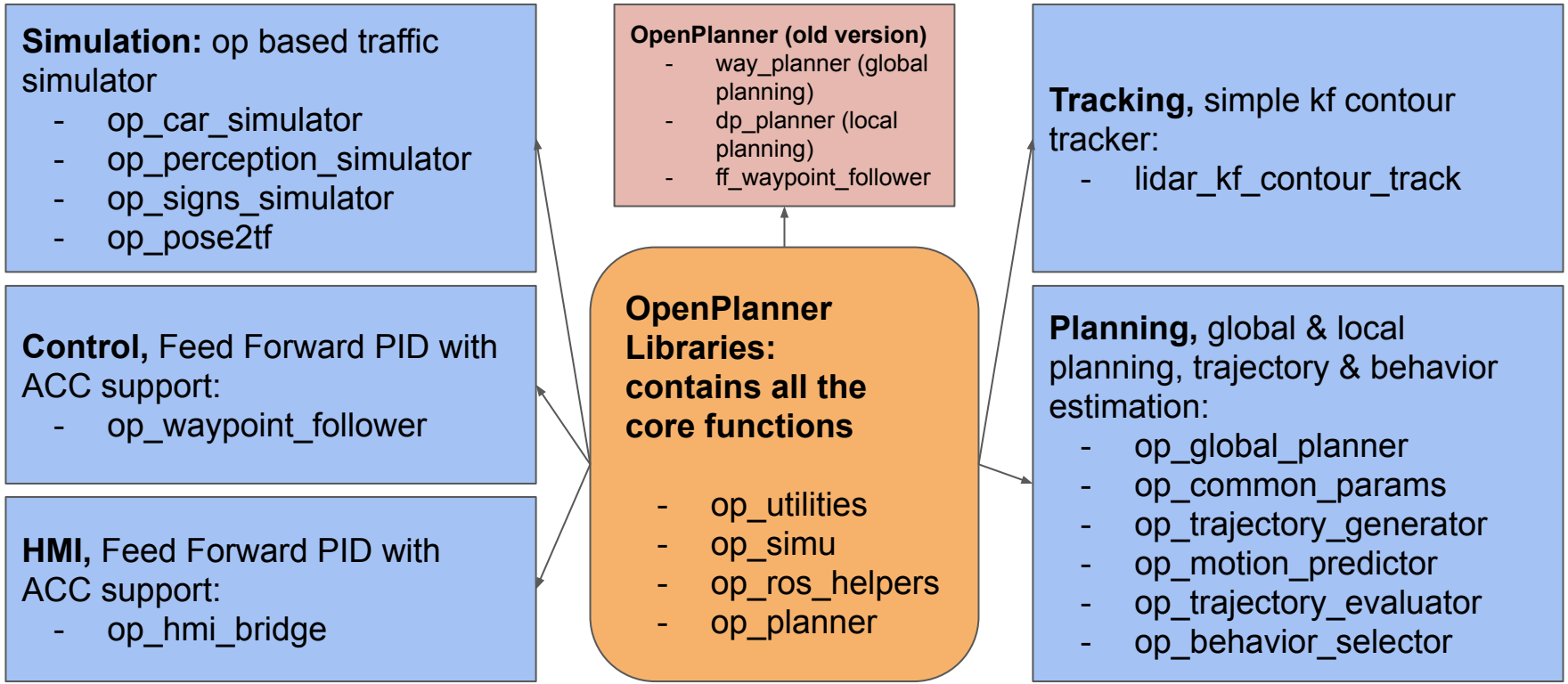
Agenda

1) Introduction to OpenPlanner
    a) Architecture
    b) Design, Libraries and ROS Nodes
    c) Versions and repositories
2) Connection with Autoware
3) OpenPlanner Features
    a) Global Planning
    b) HMI bridge
    c) Local Planning
        i) Local trajectory generation using Forward/Backward simulation
        ii) Lane Change
    d) Trajectory and Behavior Estimation
4) OpenPlanner Road Network Map Support
5) Development Roadmap

OpenPlanner 2.5 Architecture



Design - Self contained functionality



Versions and Repos

Autoware.AI Repo : https://github.com/Autoware-AI/autoware.ai

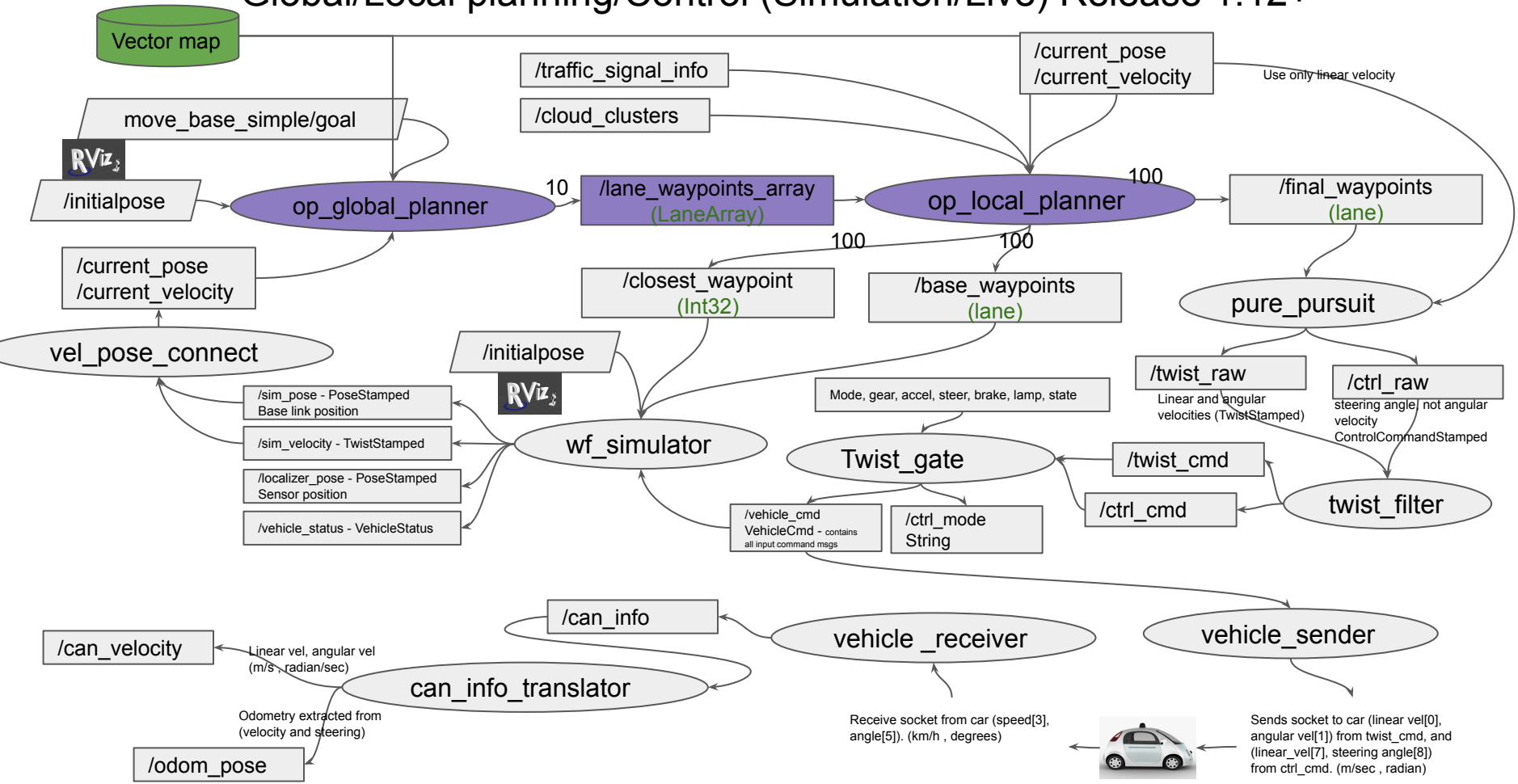
- Include up to OpenPlanner 1.5.

Personal Repo : https://github.com/hatem-darweesh

- Include modification to projects (common, core_planning, core_perception) to accommodate **OpenPlanner 2.0+**
- The GitHub upstream based on **Release 1.13.0**. Means other projects such as (utilities, visualization, simulation, messages) have to checked out as **1.13.0**
- The modifications to Release 1.13 for the mentioned projects are in branch **openplanner.1.13**

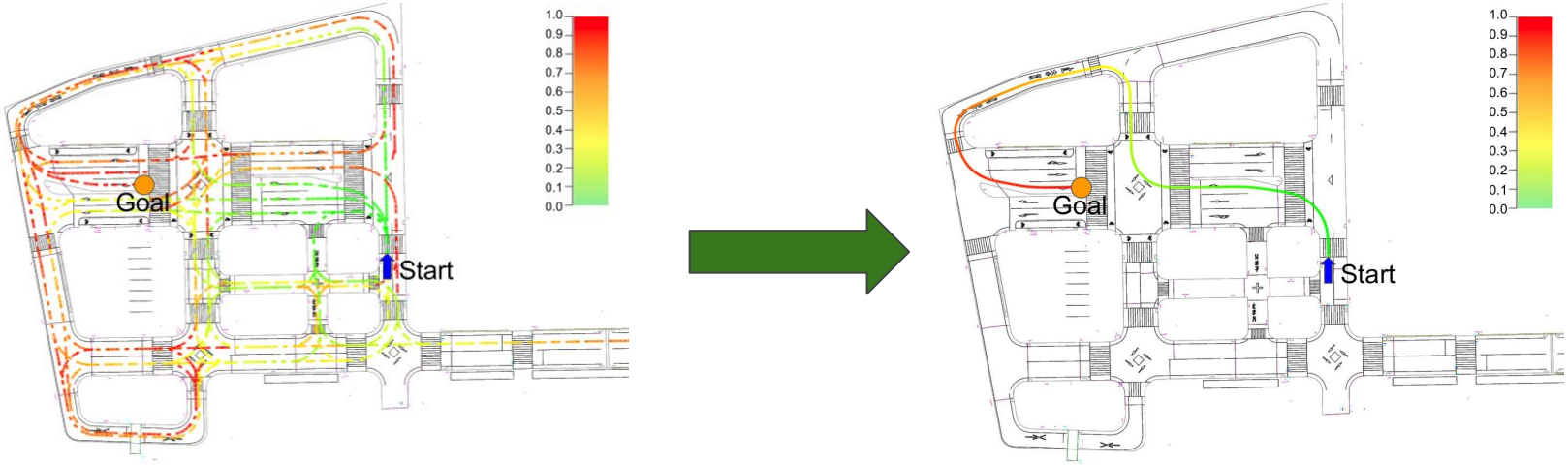
Connection to Autoware.AI

Global/Local planning/Control (Simulation/Live) Release 1.12+

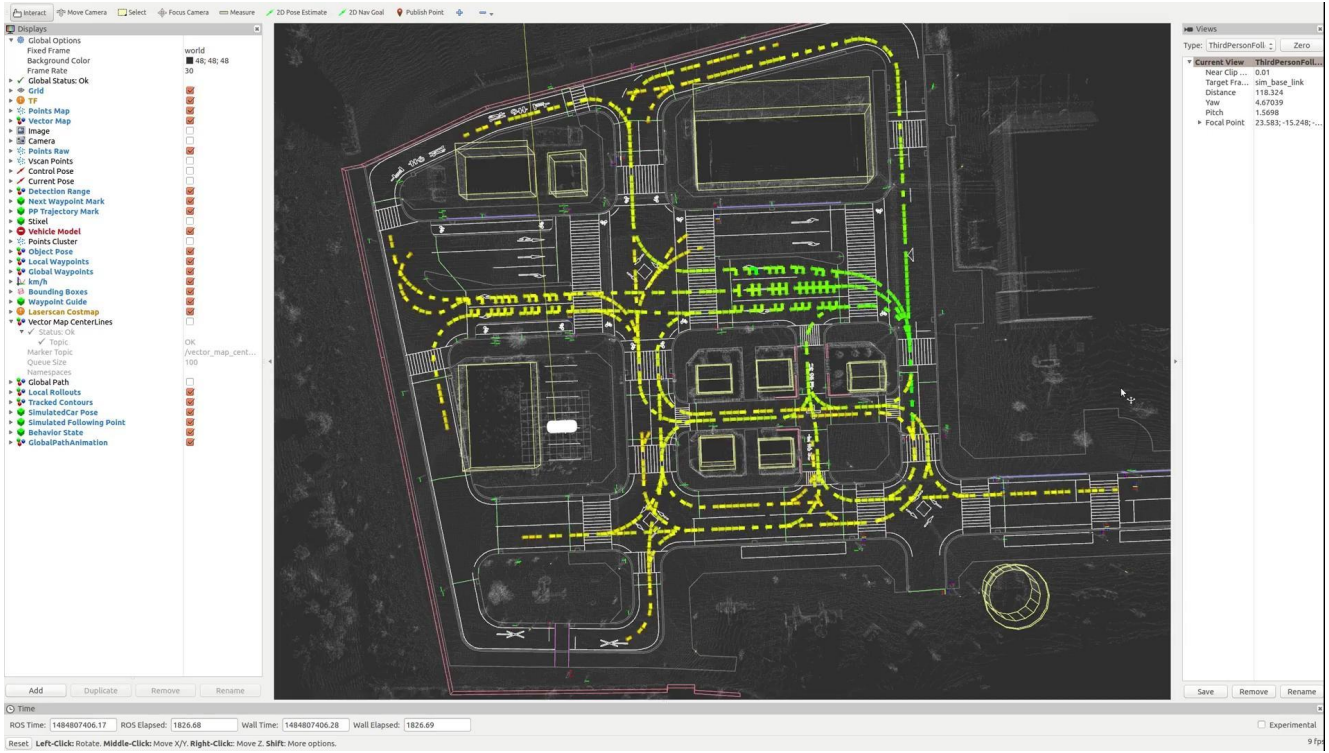


OpenPlanner Features

Global Planning



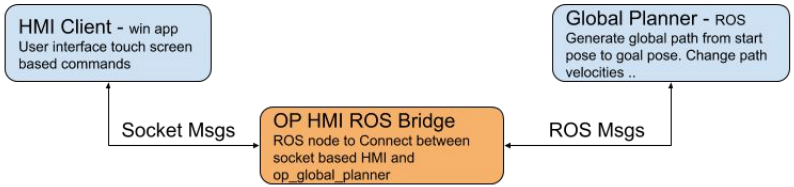
Global Planning



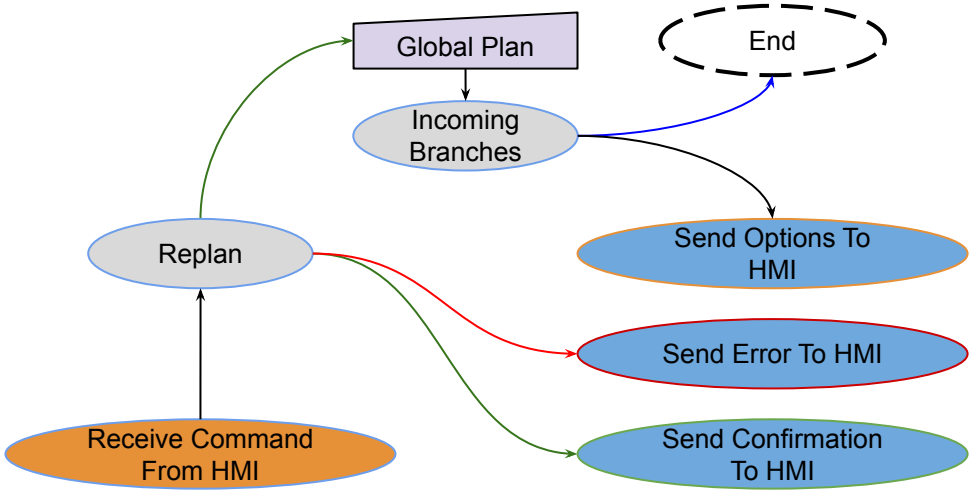
HMI Bridge

Human-Machine Interface (HMI) Bridge

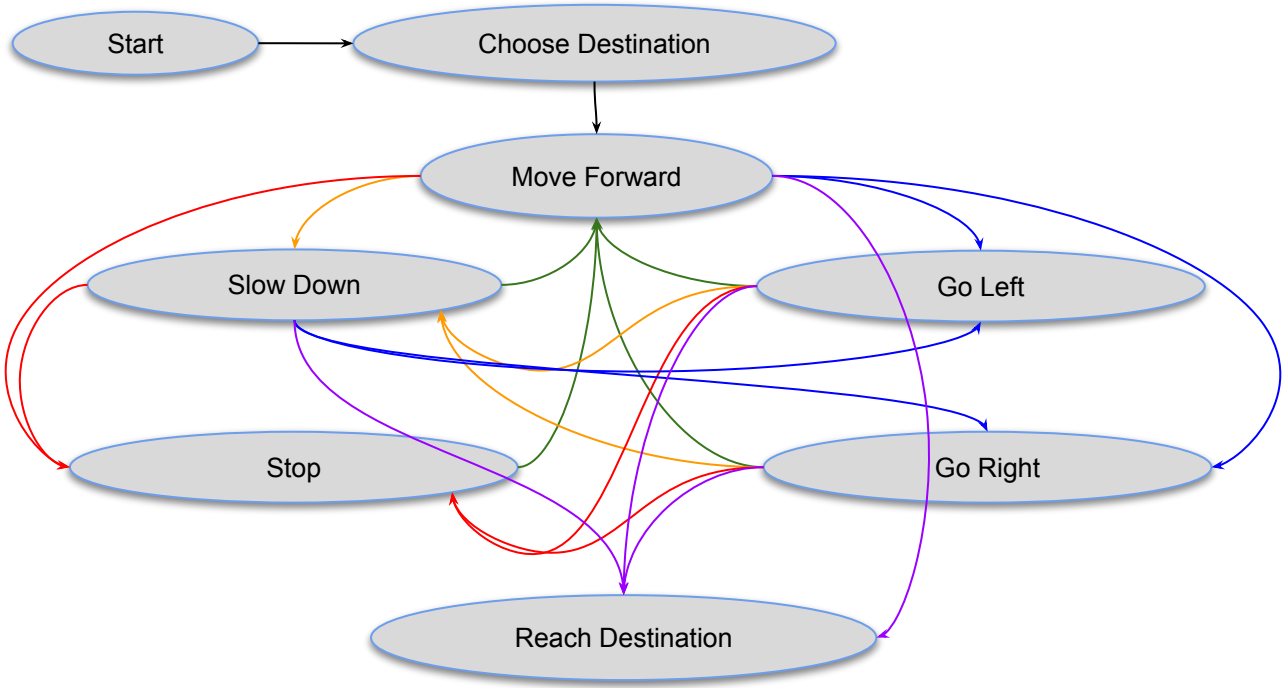
Architecture



System Flow chart

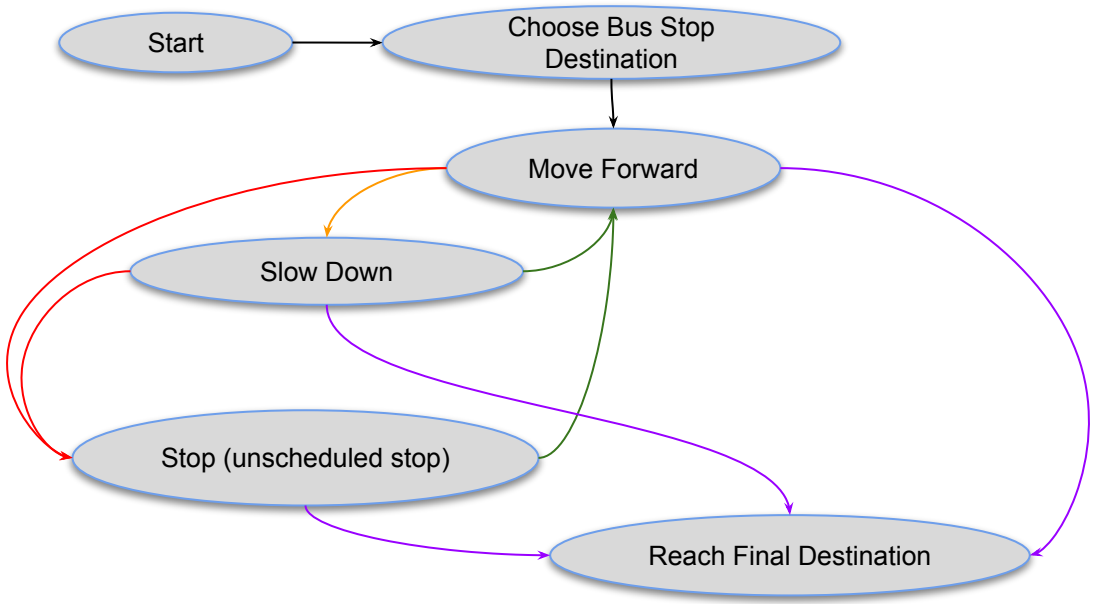


General Case - State Machine



But Routing Case - State Machine

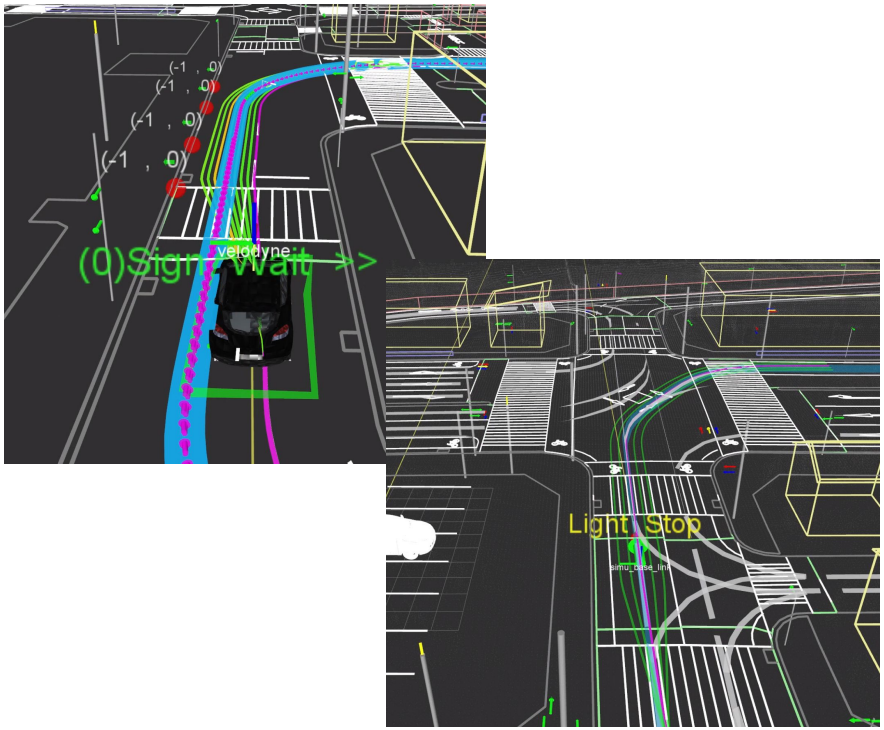
- Many special operation states could be define for a wide range of applications.



Local Planning

Local Planning OpenPlanner 1.5

- Obstacle avoidance
- Curb avoidance
- Stop sign behavior
- Traffic light behavior
- Yielding for others



Local Planner 2.5+

Local Trajectory Generation Using Forward/Backward Simulation

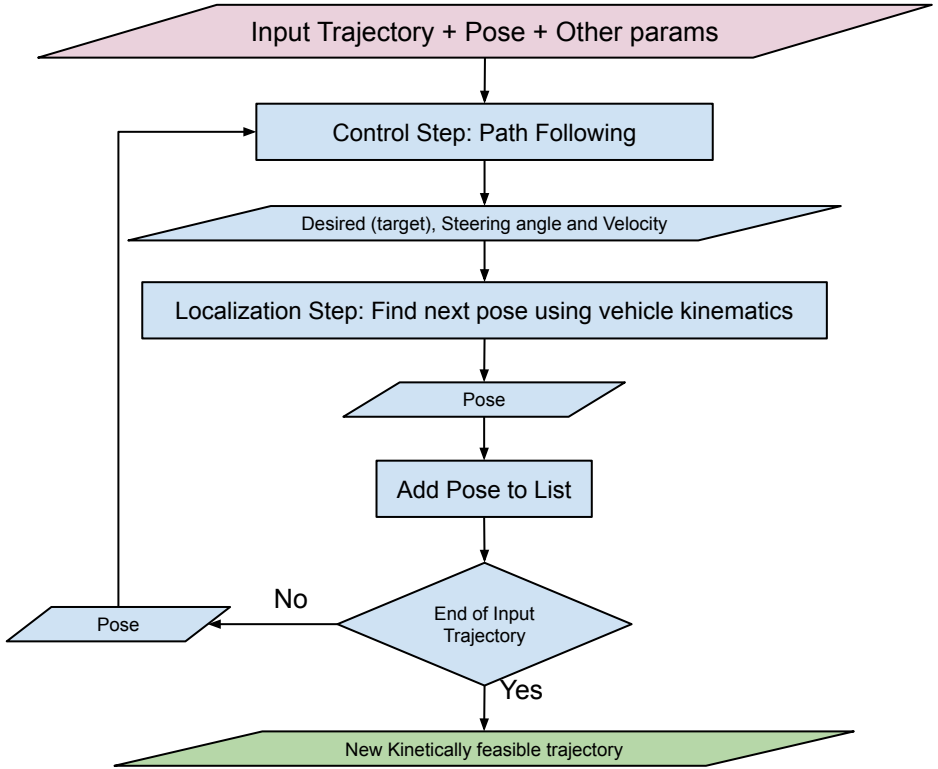
Problems with current trajectory generation approach:

- It doesn't consider any vehicle parameters (kinematics or dynamics)
- Generated trajectory is impossible to follow exactly using waypoint following controllers
- Generating smoother trajectories leads to cutting corners especially in the tight corners
- Because there are big difference between the generated trajectories and the actual vehicle motion path, trajectory evaluation is not accurate

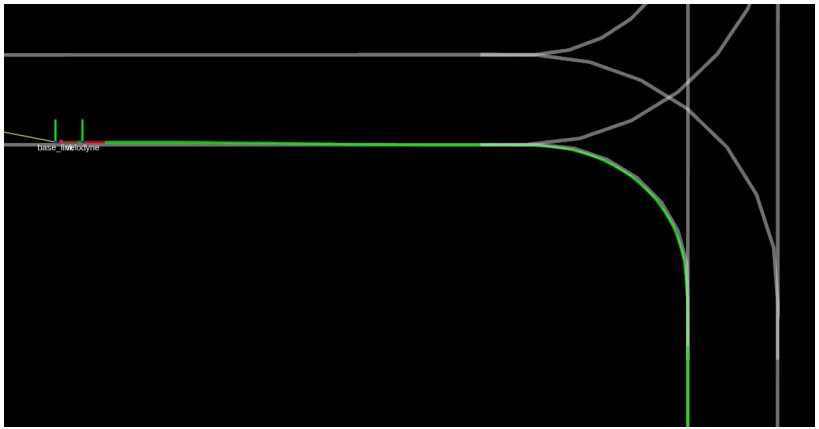
Solution: Forward/Backward Simulation

- Forward simulation consists of two main stages:
  - Motion step (using the kinematic model of the vehicle including the steering delay).
  - Control step (using P controller to find control signals for each moving step)

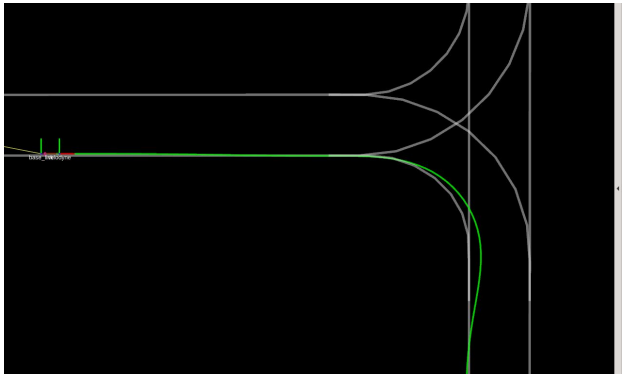
- Backward simulation:
  - Currently achieved by switching start/goal points and simulate the vehicle moving on the opposite direction.



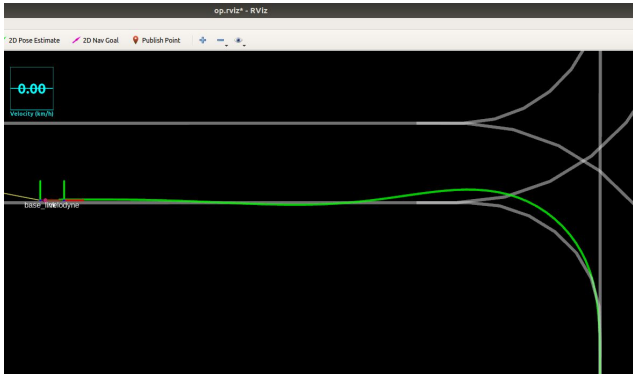
Example - 90 degree turn



Forward Simulation



Backward Simulation



Algorithm analysis

Advantages:

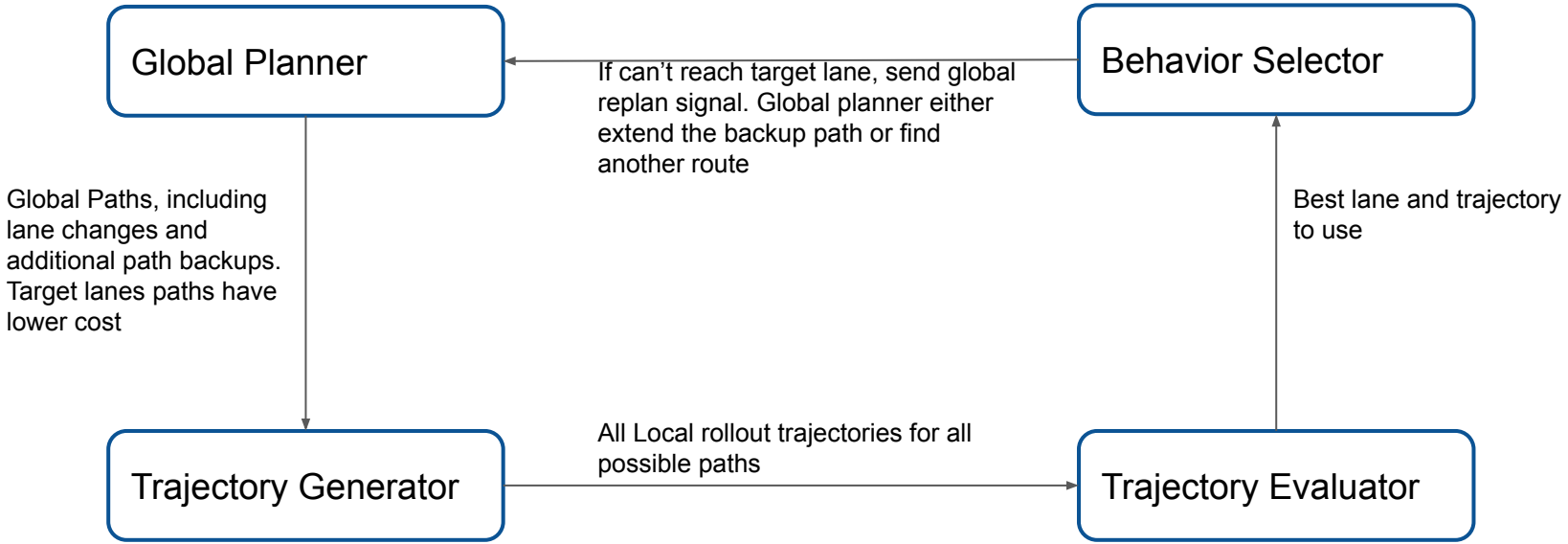
- Less parameters to tune compared to other trajectory optimization techniques
- Once it is tuned it will work within the kinematics limits of the vehicle
- Could plan backward, so it will find the path which can lead to exact target position and orientation. With forward planning only this can’t be guaranteed

Disadvantages:

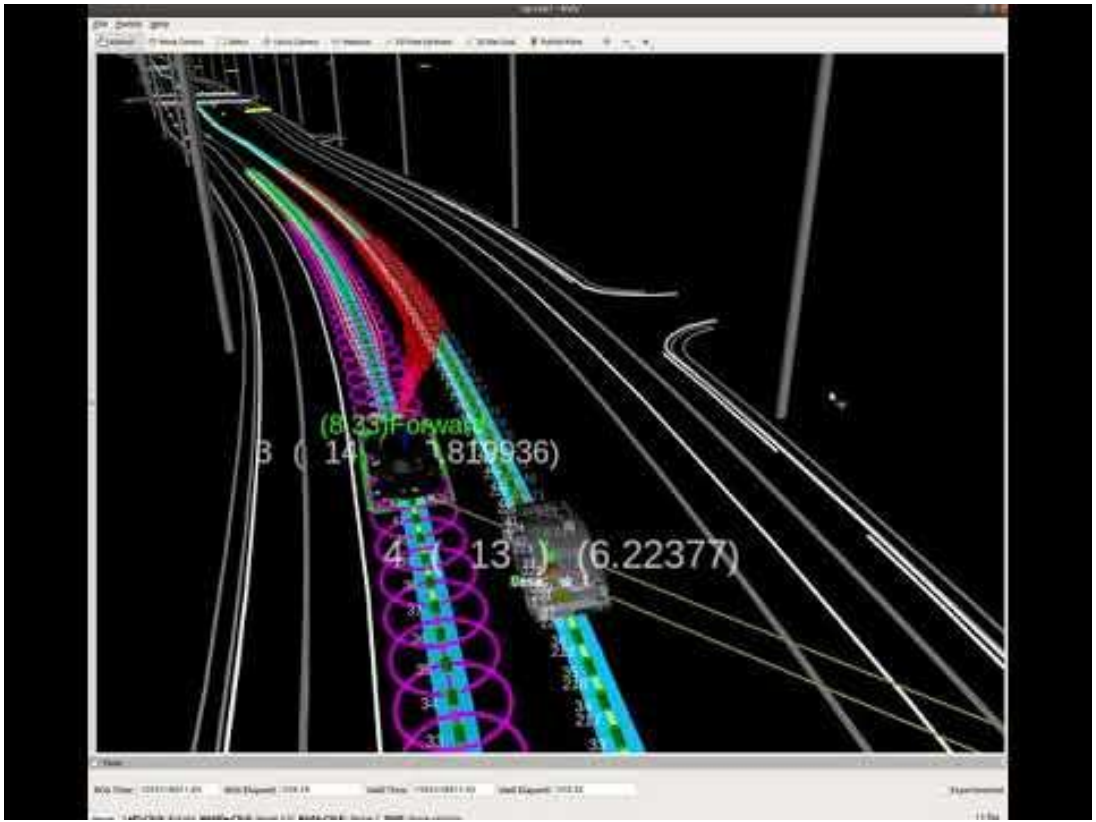
- Same as normal open planner trajectory generation, if the trajectory is switched often that could lead to overshooting or oscillation
- Shouldn't be called while driving in a tight turn, it will overshoot

Lane Change in Local Planner

- In OpenPlanner 1.5, lane change was supported only in Global Planning step.



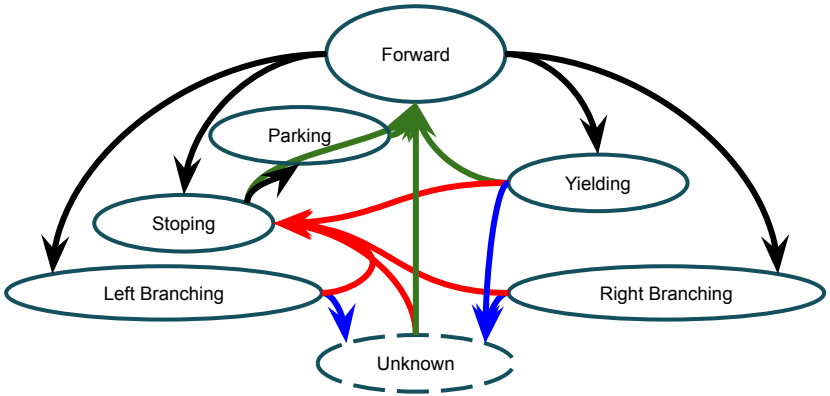
Lane Change - Worst case scenario

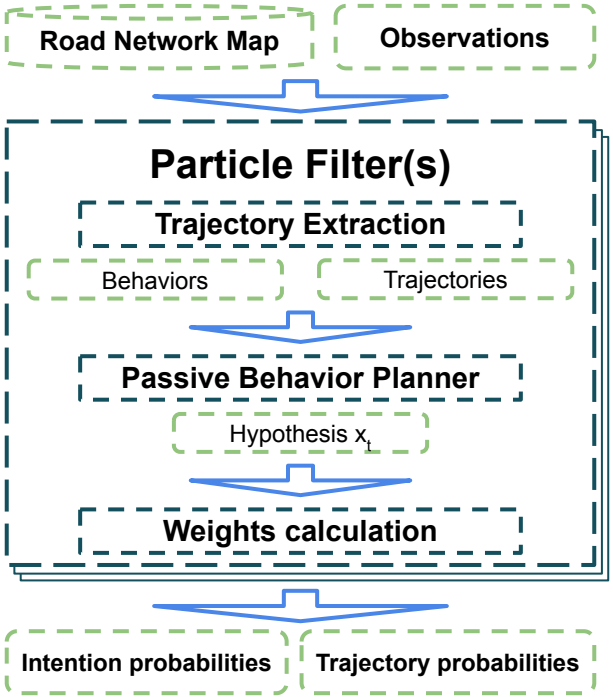


Trajectory and Behavior Estimation

Behavior and Trajectory Estimation *

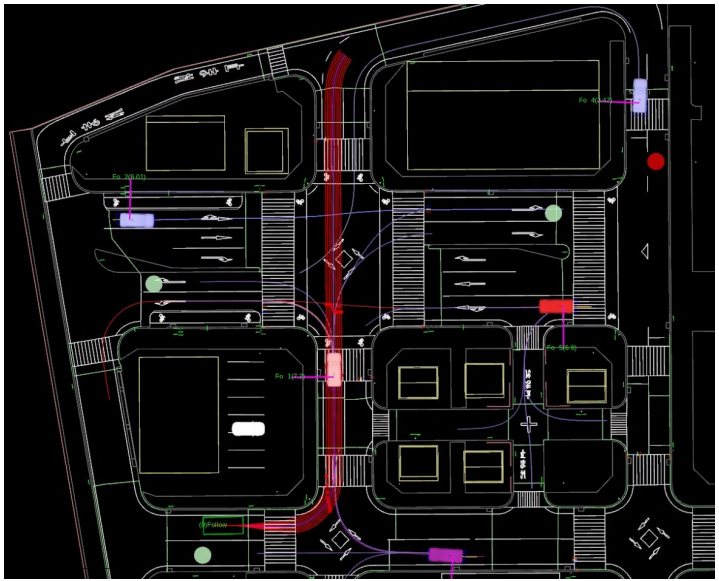
- Novice technique using behavior planner with multi-cue particle filter to estimate intention and trajectory of surrounding vehicles.
- This is an important step for the planner to be able to handle complex social situation.

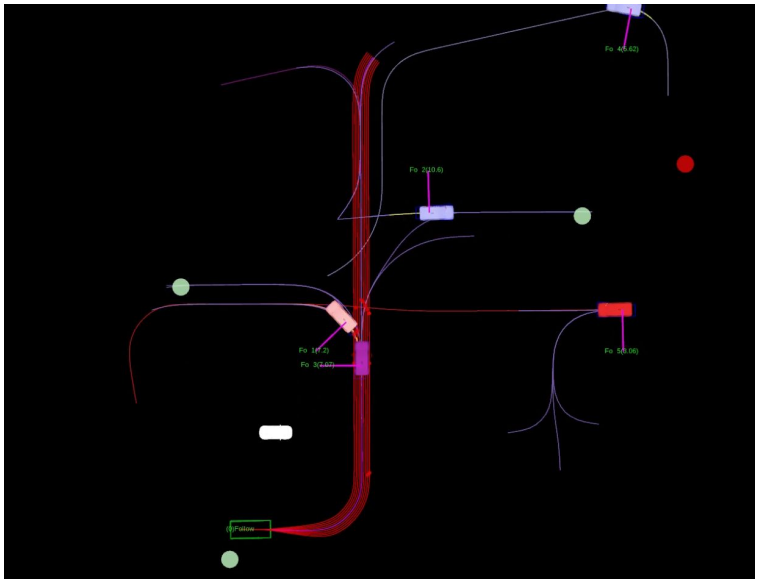




* “Estimating The Probabilities of Surrounding Vehicle’s intentions and Trajectories using a Behavior Planner”. IJAE Vol.10 No.4, 2019

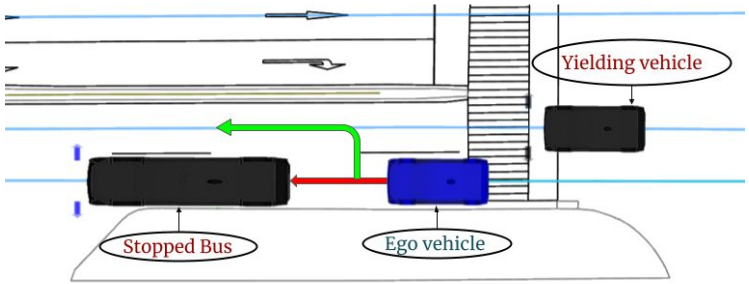
Step 1) Trajectory Extraction from Road network map

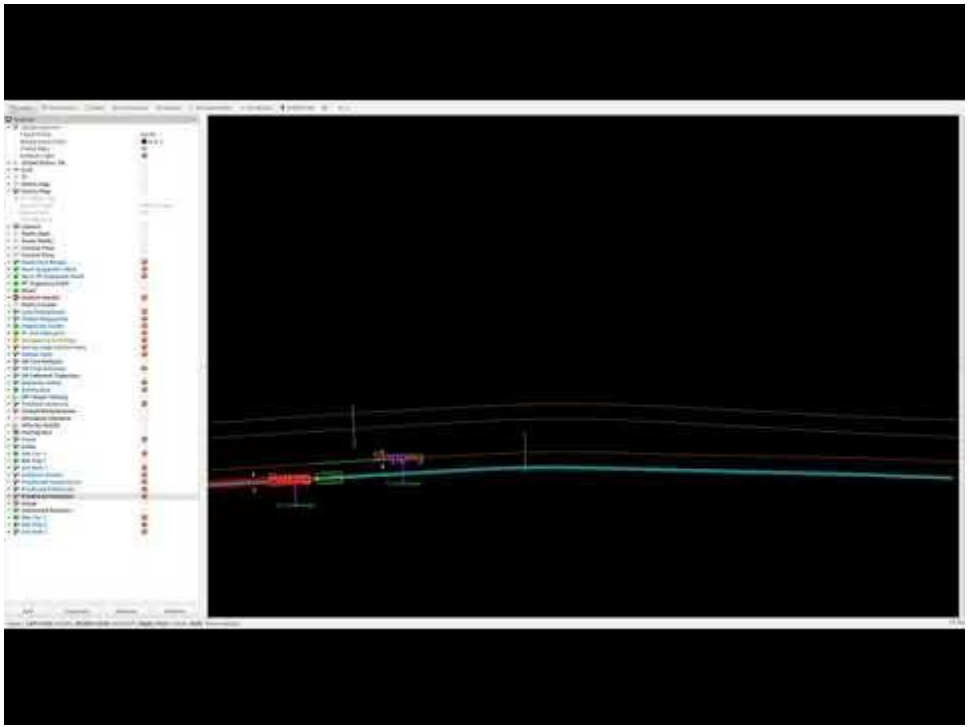




Step 2) Assign probabilities to trajectories and behaviors

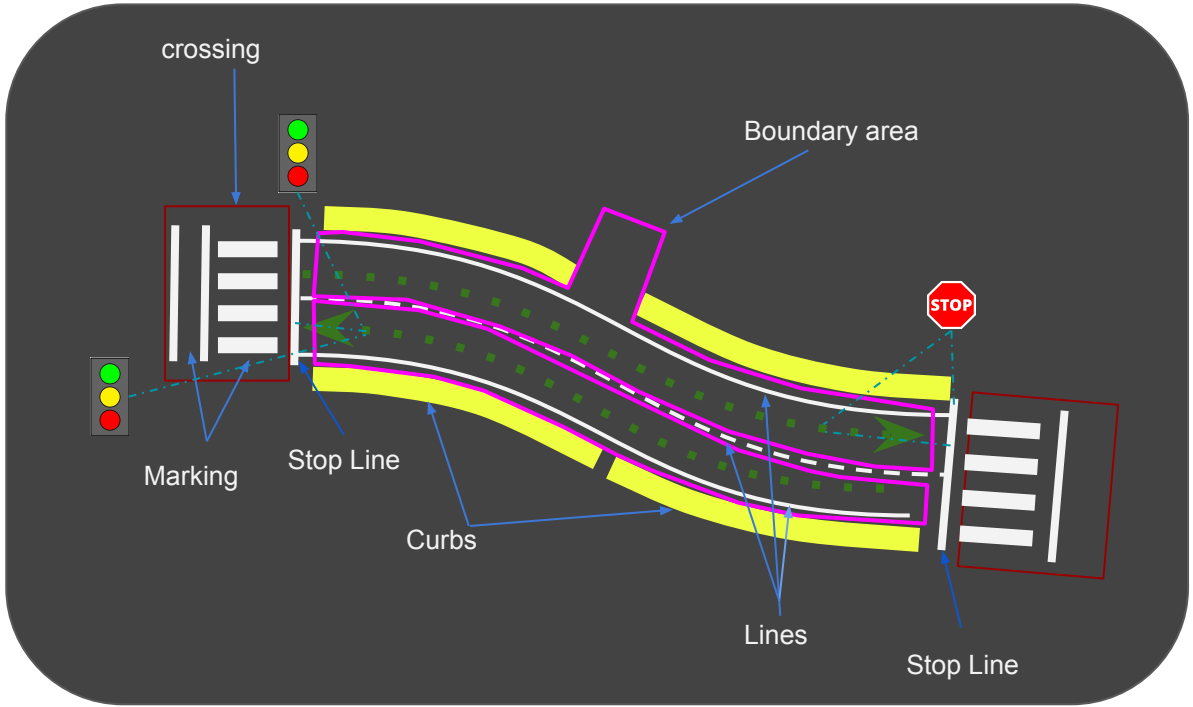
- It could understand both intentions of the Bus and the vehicle on the other lane.
- It is easy for decision make now to decide between overtake or wait.



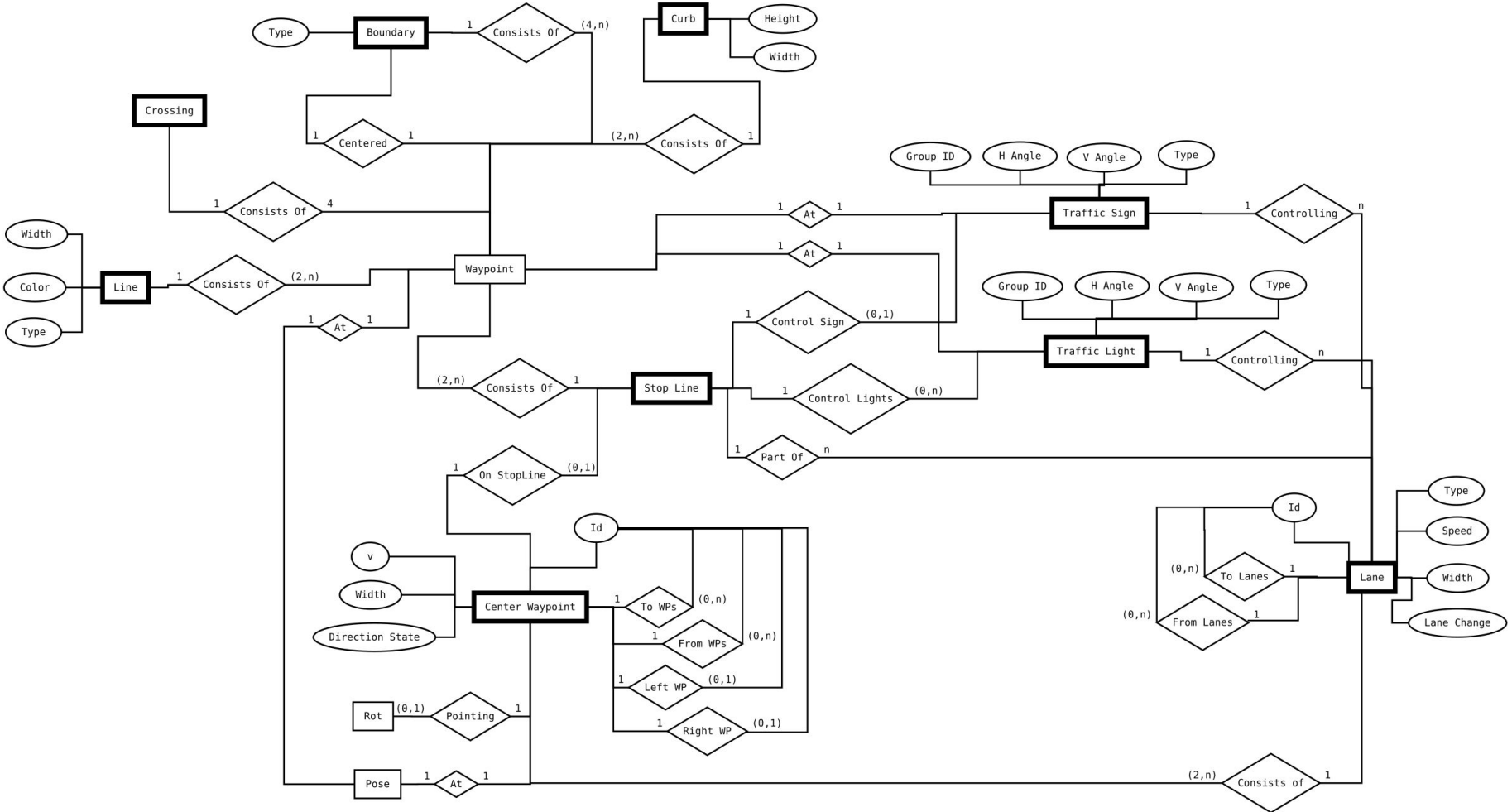


Road Network Map Support

HD Map supported by OpenPlanner



Internal Road Network Map ER design (KML format)



Supported formats vs Versions

AISAN Vector Maps (.csv files), Supported by all versions:

- Receive from Autoware then convert to internal format
- Parse .csv files then convert to internal format

Lanelet2 (.osm file), Supported by OpenPlanner 2.0+ :

- Parse .osm file then convert to internal format

KML (.kml file), Supported by all versions:

- Parse .kml file then convert to internal format

OpenDRIVE (.xodr file), Third party conversion :

- Convert offline using ASSURE mapping tool to KML or Lanelet2.

Development Roadmap

OpenPlanner Future development

- Use **Markov Decision Processes (MDP)** instead of deterministic trajectory evaluation.
    - That will replace finding only “best trajectories” with finding “best policy”, which include both safe trajectory and recommended velocity.
- **Overtake behavior** state should be integrated with Global planning similar to Lane change.
- OpenPlanner 2.0+ rebase to **Autoare.AI 1.15** Release. Release is still pending for testing.
- ROS2 implementation for OpenPlanner 2.0+ nodes so it could be easily integrated to **Autoware.Pilot & Autoware.Auto**.

Thank You