

# Autoware Architecture Proposal

Architecture proposal for AWF by Tier IV Inc.

# Agenda

1. Why we need a new architecture
2. Considered use cases
3. Architecture overview
  - Layered architecture
  - High level introduction of each module
4. Contribution steps to Autoware community
5. Conclusion

# 1

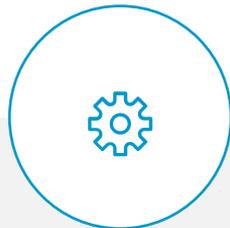
## Why we need a new architecture

# Why we need a new architecture



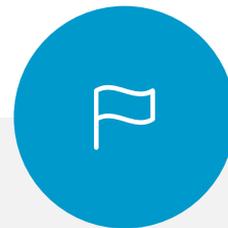
## Problem

- It's difficult to improve Autoware.AI capabilities



## Why?

- No concrete architecture design
- A lot of technical debt
  - Tight coupling between modules
  - Unclear responsibility of modules



## Tier IV's proposal

- Define a layered architecture
- Clarify the role of each module
- Simplify the interface between modules

# 2

## Considered use cases

# Considered use cases

Example use cases that were considered during architecture design:

Module	Use cases
Sensing	<ul style="list-style-type: none"><li>• 360-degree sensing by the camera-LiDAR fusion</li></ul>
Perception	<ul style="list-style-type: none"><li>• Recognition of dynamic objects and traffic lights</li></ul>
Localization	<ul style="list-style-type: none"><li>• Robust Localization using multiple data sources</li></ul>
Planning	<ul style="list-style-type: none"><li>• Route planning, dynamic planning based on vector map (not only waypoint following)</li><li>• Automatic parking</li><li>• Object avoidance</li></ul>
Control	<ul style="list-style-type: none"><li>• High control performance on many kinds of vehicle-controllers</li></ul>

Features that are not considered yet (for the sake of development speed)

- Real-time processing
- HMI / Fail safe / Redundant system / State monitoring system / etc...

Will consider these items at AWF WGs

Module	Contents
Whole	<a href="#">Scenario demo</a>
Sensing+Perception	<a href="#">360° FOV</a> , <a href="#">Prediction</a>
Localization	<a href="#">Robustness of localization</a> , <a href="#">Return from error</a>
Planning	<a href="#">Lane change</a> , <a href="#">Obstacle avoid</a> , <a href="#">Parking</a>
Control	<a href="#">Slow brake (normal stop)</a> , <a href="#">Rapid brake (emergency stop)</a>

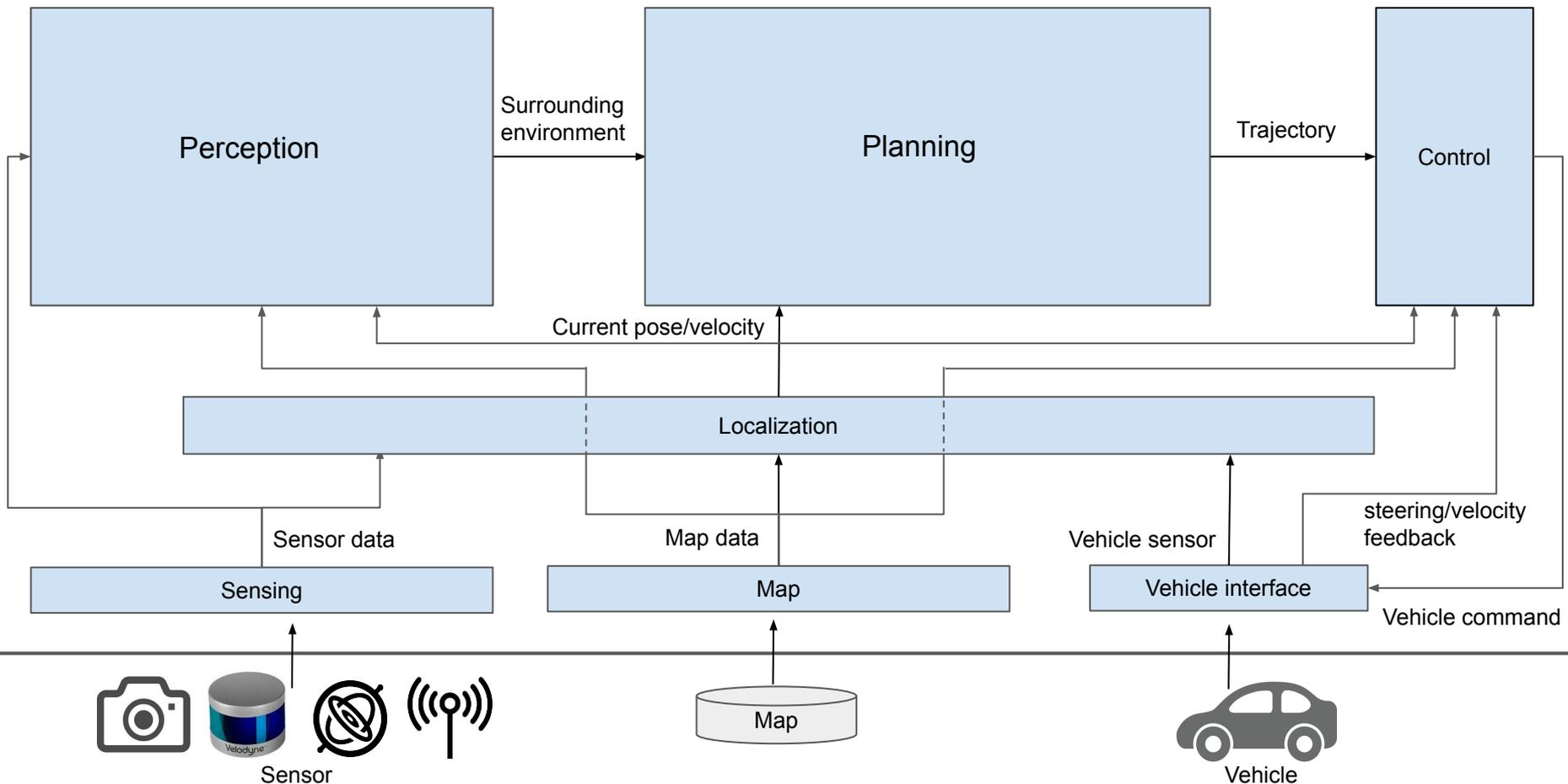
# 3

## Architecture overview

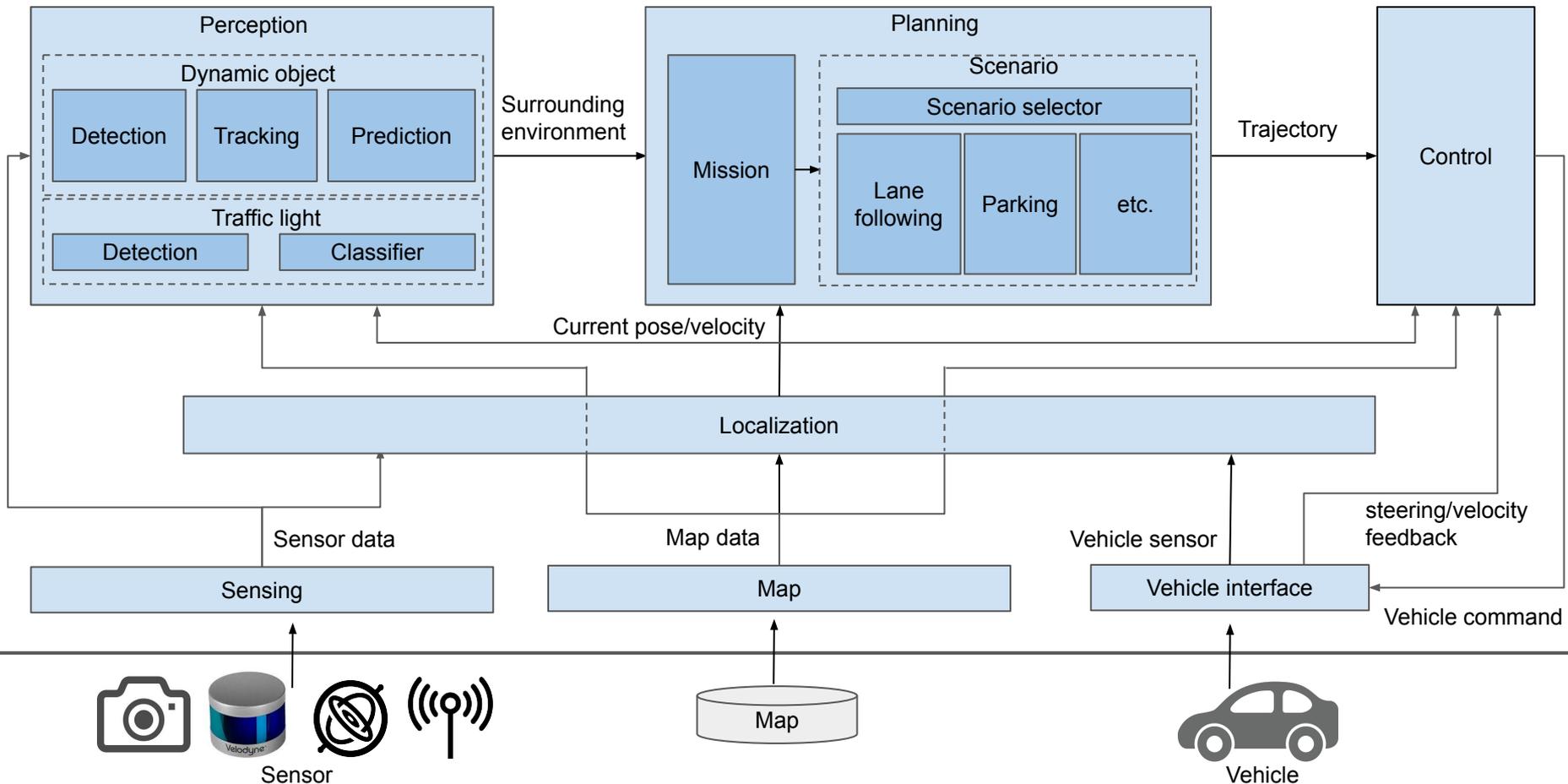
Layered architecture

High level introduction of each module

## Autoware

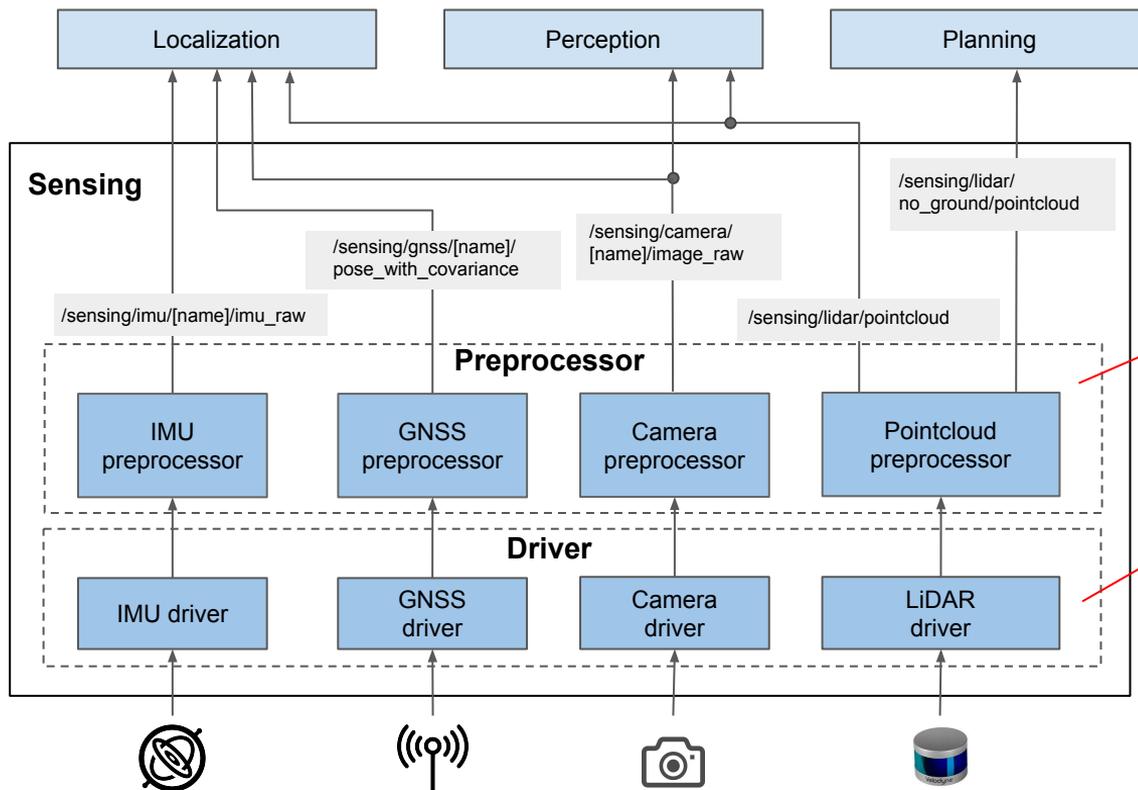


## Autoware





Role : Conversion of sensing data to ROS message

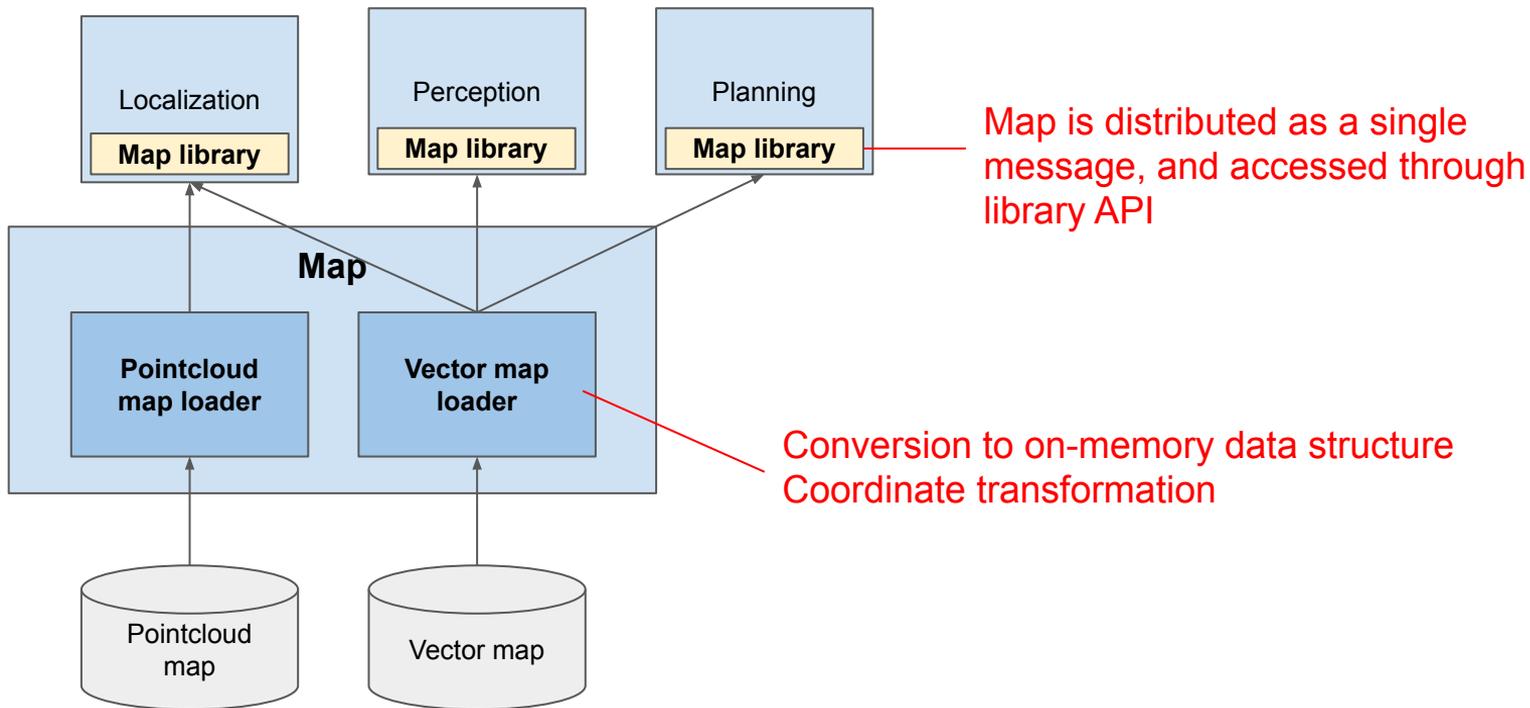


Examples of preprocessors

- Distortion correction
- Outlier filter
- Concat filter
- Ground filter

ROS drivers (mainly from upstream)

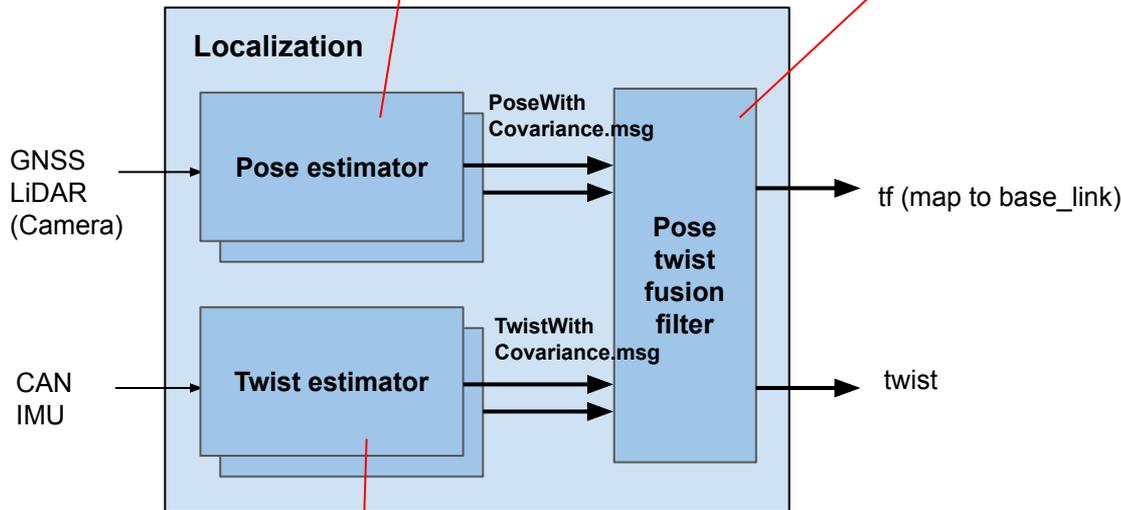
**Role :** Distribute static environment information to other modules



**Role :** Integration of each sensor data and estimation of self-pose and self-twist

Support multiple localization methods  
based on different sensors(LiDAR/Camera/GNSS)

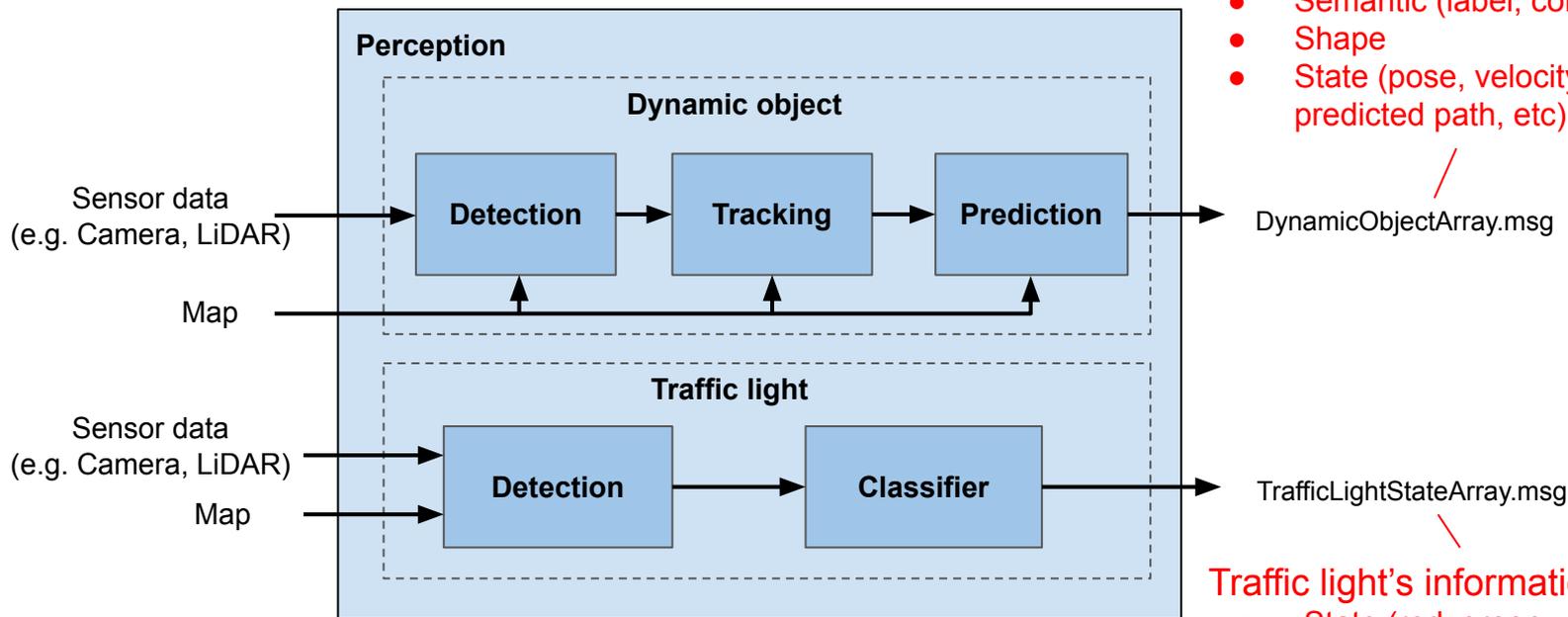
Fuse pose and twist (e.g.EKF)



Merge sensor inputs



Role : Dynamic object recognition and traffic light recognition



Dynamic object's information including:

- ID
- Semantic (label, confidence)
- Shape
- State (pose, velocity, acceleration, predicted path, etc)

DynamicObjectArray.msg

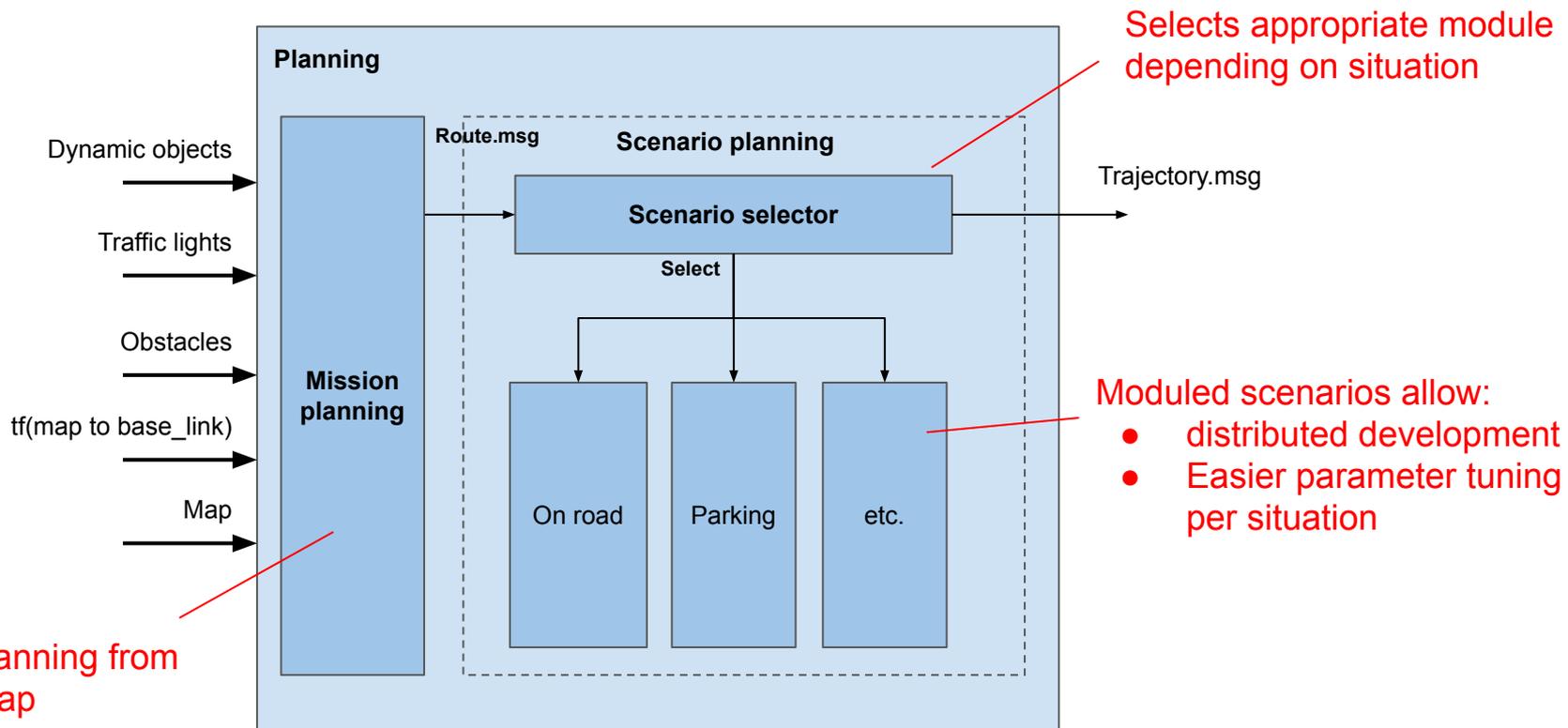
TrafficLightStateArray.msg

Traffic light's information including:

- State (red, green, yellow, etc)
- Could be customized for unique traffic light types.

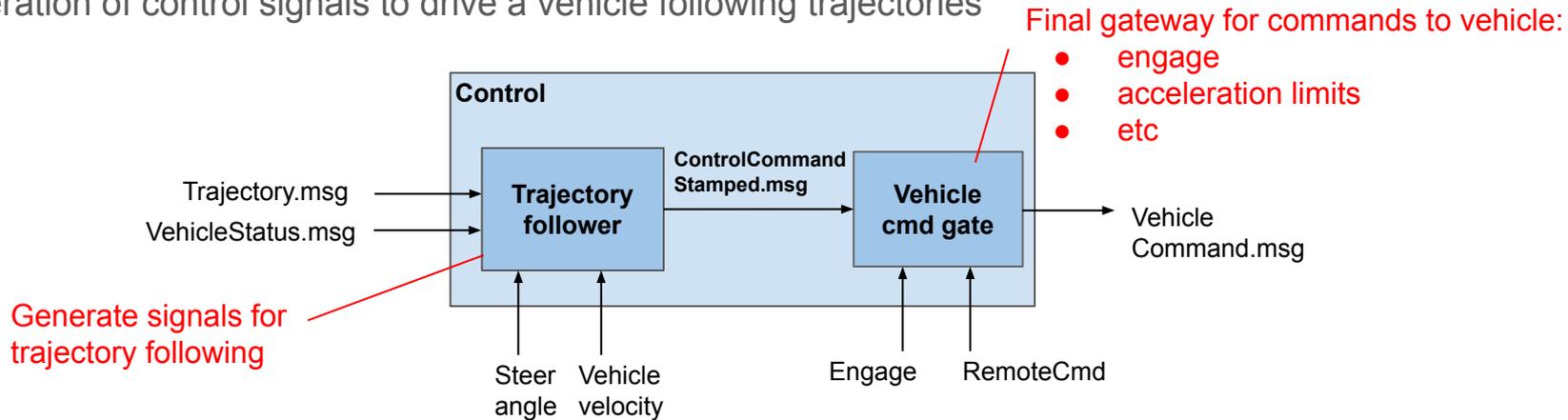


**Role :** From a calculation of the route to the goal to drive trajectory generation





**Role :** Generation of control signals to drive a vehicle following trajectories



## VehicleCmd.msg (autoware.ai)

```
header
steer_cmd
-header
-steer
accel_cmd
-header
-accel
brake_cmd
-header
-brake
gear
twist_cmd
-header
-linear
-angular
ctrl_cmd
-linear velocity
-linear acceleration
-steering angle
```

Remove  
redundant output

## VehicleCommand.msg (proposed architecture)

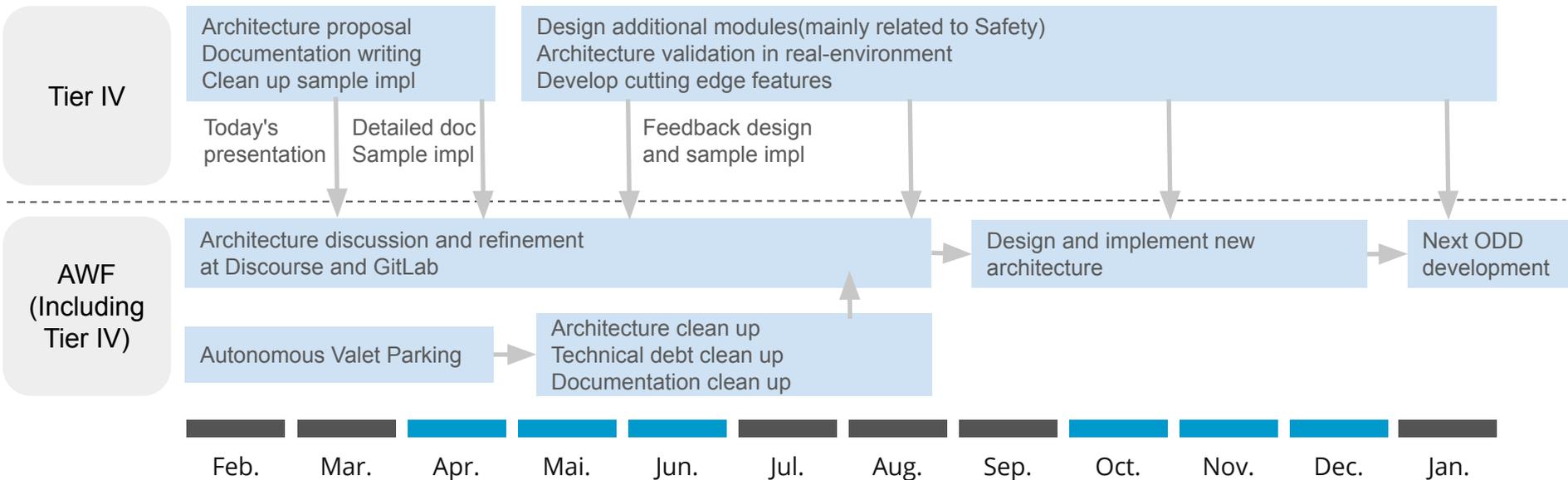
```
header
control
-steering angle
-velocity
-steering angle velocity
-acceleration
shift
-data
```

Support low level control

# 4

## Contribution steps to Autoware community

# Contribution steps to Autoware community



# 5 Conclusion

- We proposed a new architecture of Autoware
- Tier IV will contribute our achievements to Autoware community
  - Detailed design documents
  - Reference source code
- We would like to discuss the new architecture at WGs
  - Any feedback is welcome
- We will make an effort to implement the new architecture in Autoware.Auto

# Backup

- Design alternative
- Module implementation
  - Implementation details
  - Achievements

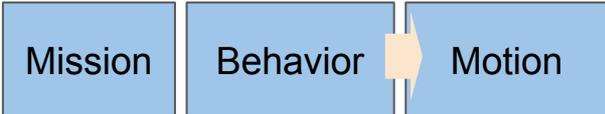
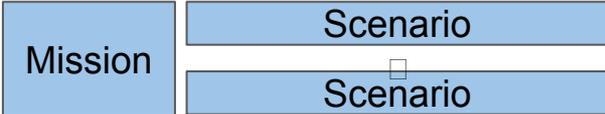
# Backup agenda

---

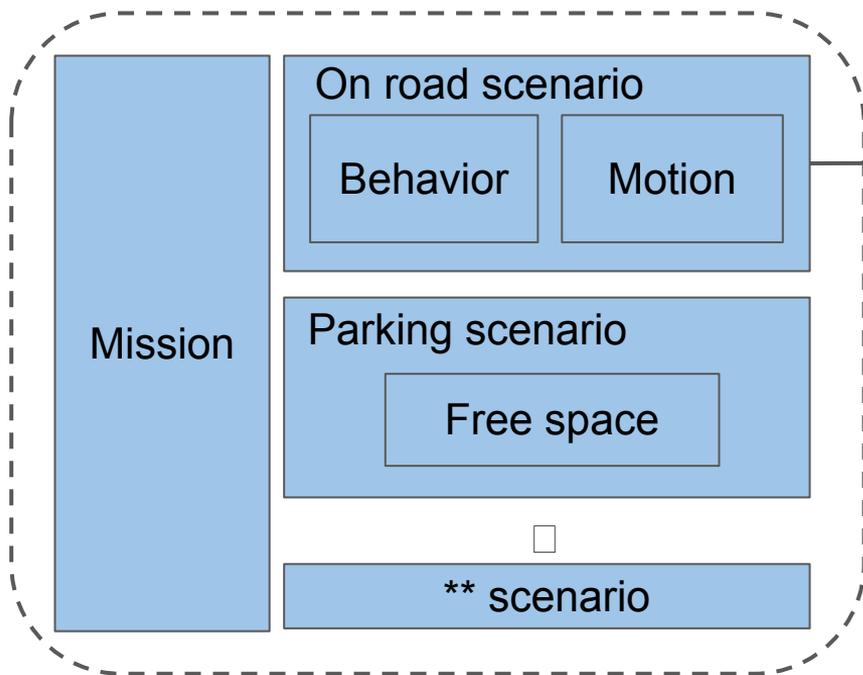
- Design alternative
- Module implementation
  - Implementation details
  - Achievements

# [Design alternative] Planning (1/2)

## Pre-study of the planning architecture

Type	Structure	Pros.	Cons.
BOSS type (Junior type)		<ul style="list-style-type: none"> <li>• Structure is intuitive and easy to understand.</li> <li>• Easy to handle separately.</li> </ul>	<ul style="list-style-type: none"> <li>• Behavior has to make a conservative decision in order to reduce discrepancies with a motion's decision.</li> </ul>
CMU Dr. paper type		<ul style="list-style-type: none"> <li>• It solves and optimizes both behavior and motion simultaneously to overcome a cons. of the BOSS type.</li> </ul>	<ul style="list-style-type: none"> <li>• Difficult to improve if any issue was found.</li> <li>• Optimization of the whole autonomous driving functions is not realistic..</li> </ul>
Victor Tango type		<ul style="list-style-type: none"> <li>• Behavior and motion are tightly coupled to overcome a cons. of the BOSS type.</li> <li>• Easy to improve if any issue was found</li> </ul>	<ul style="list-style-type: none"> <li>• Difficult to take a new research result into the modules.</li> </ul>
Apollo type		<ul style="list-style-type: none"> <li>• Scenario is one layer upper concept.</li> <li>• We can adopt preferred types of the planning structure according to each scenario.</li> </ul>	<ul style="list-style-type: none"> <li>• In the scenario, cons. such as things described above are still remained.</li> <li>• If scenarios increased, switching mechanism can be complicated.</li> </ul>

Finally, we adopted **Apollo + Boss type**



Lane following scenario : BOSS type

- It's clear and convenient to develop as OSS

Whole module : Apollo type

- New scenario can be added as needed

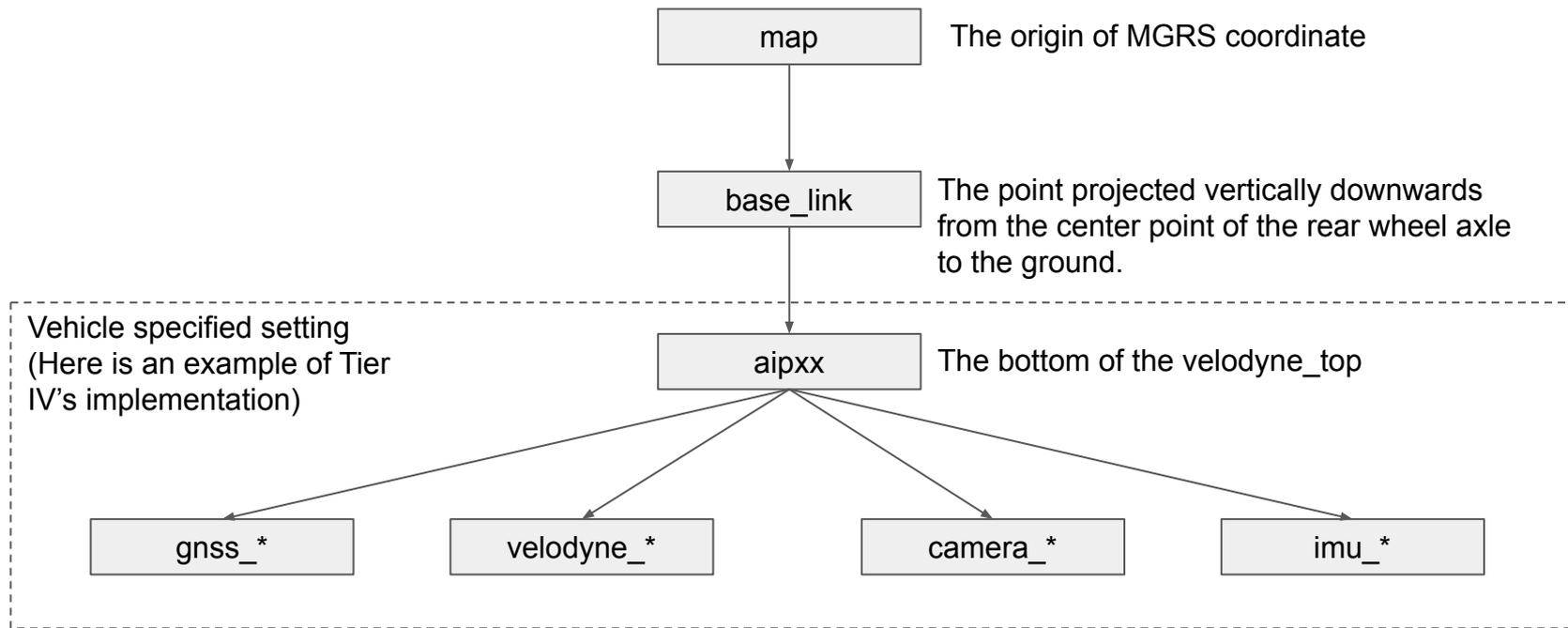
Future challenge

- In lane following scenario, behavior still makes a conservative decision. And an aggressive driving like changing lane by entering into crowded lane cannot be executed.

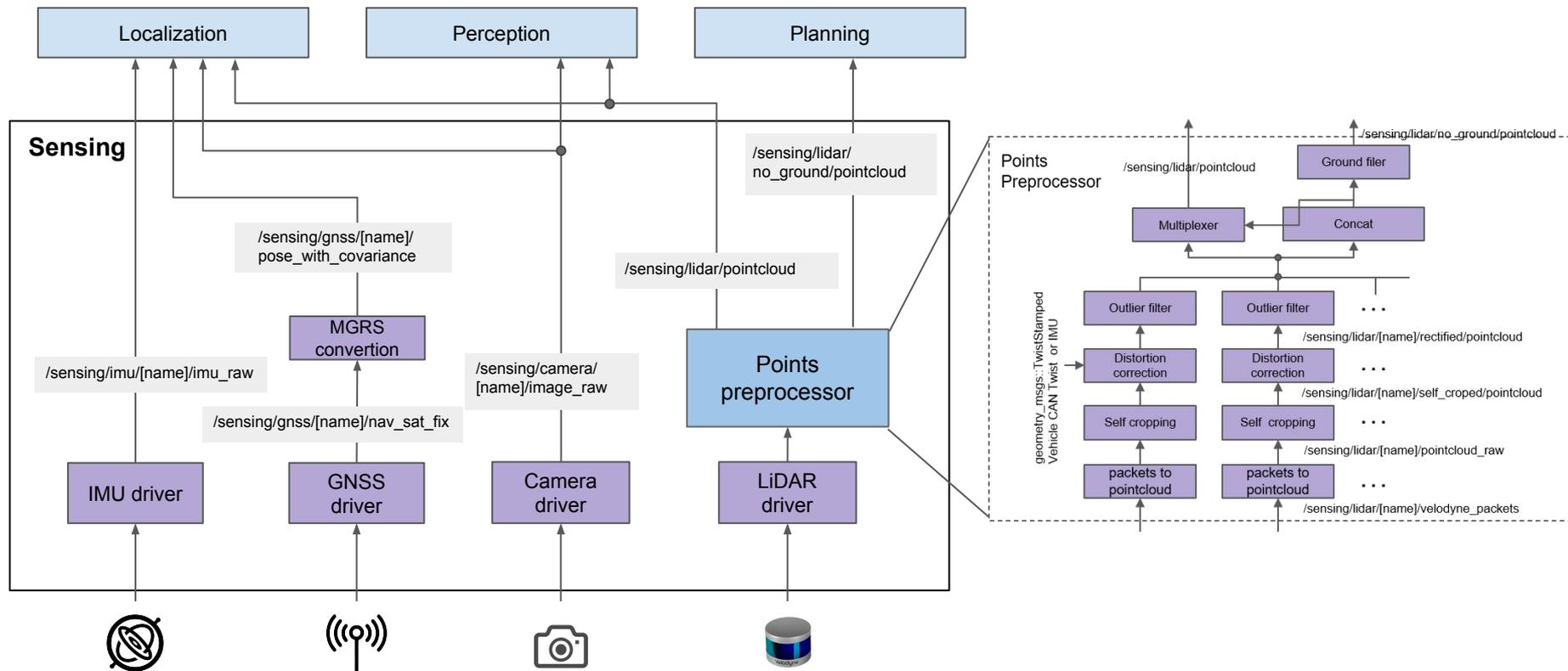
# Backup agenda

---

- Design alternative
- **Module implementation**
  - Implementation details
  - Achievements

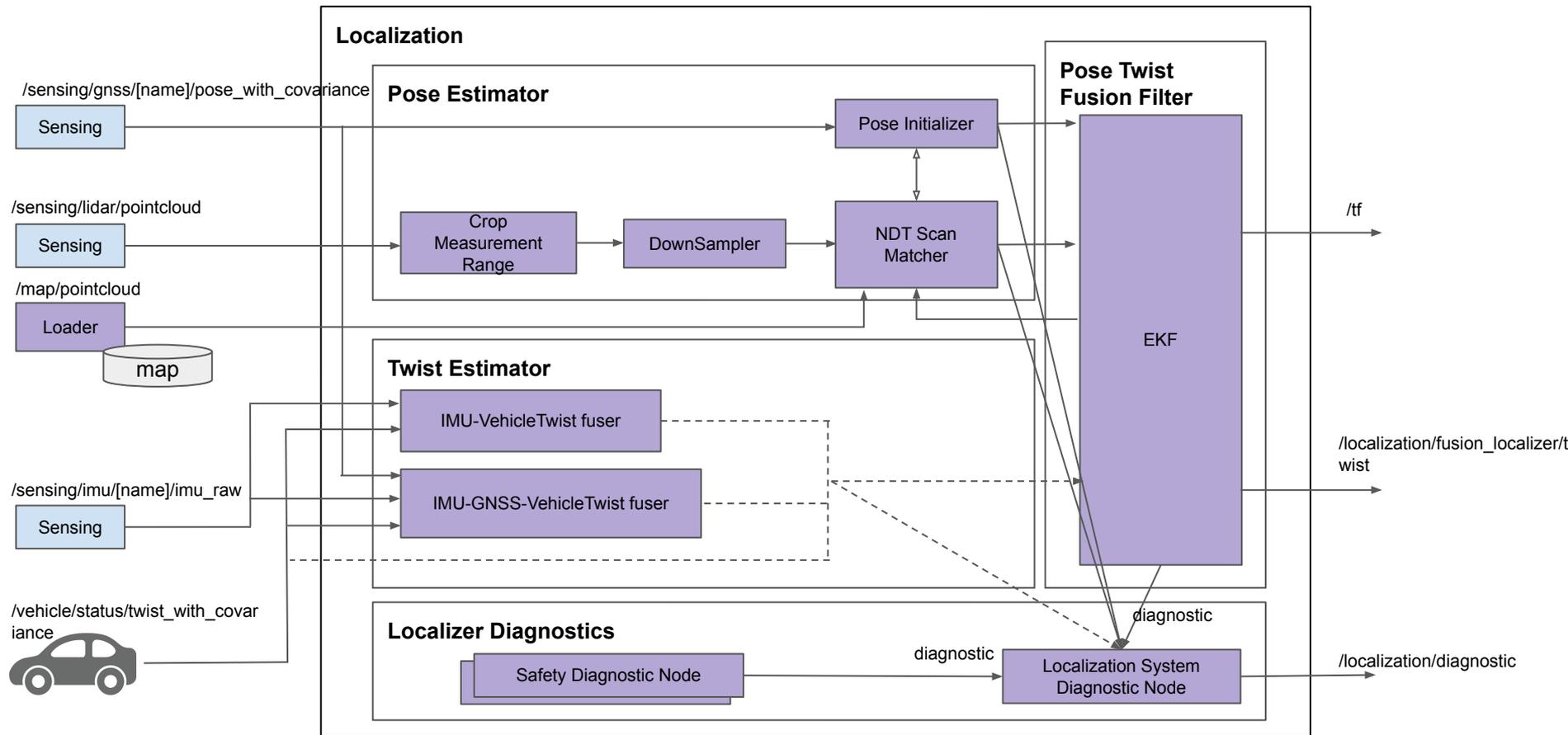


# [Sensing] Components



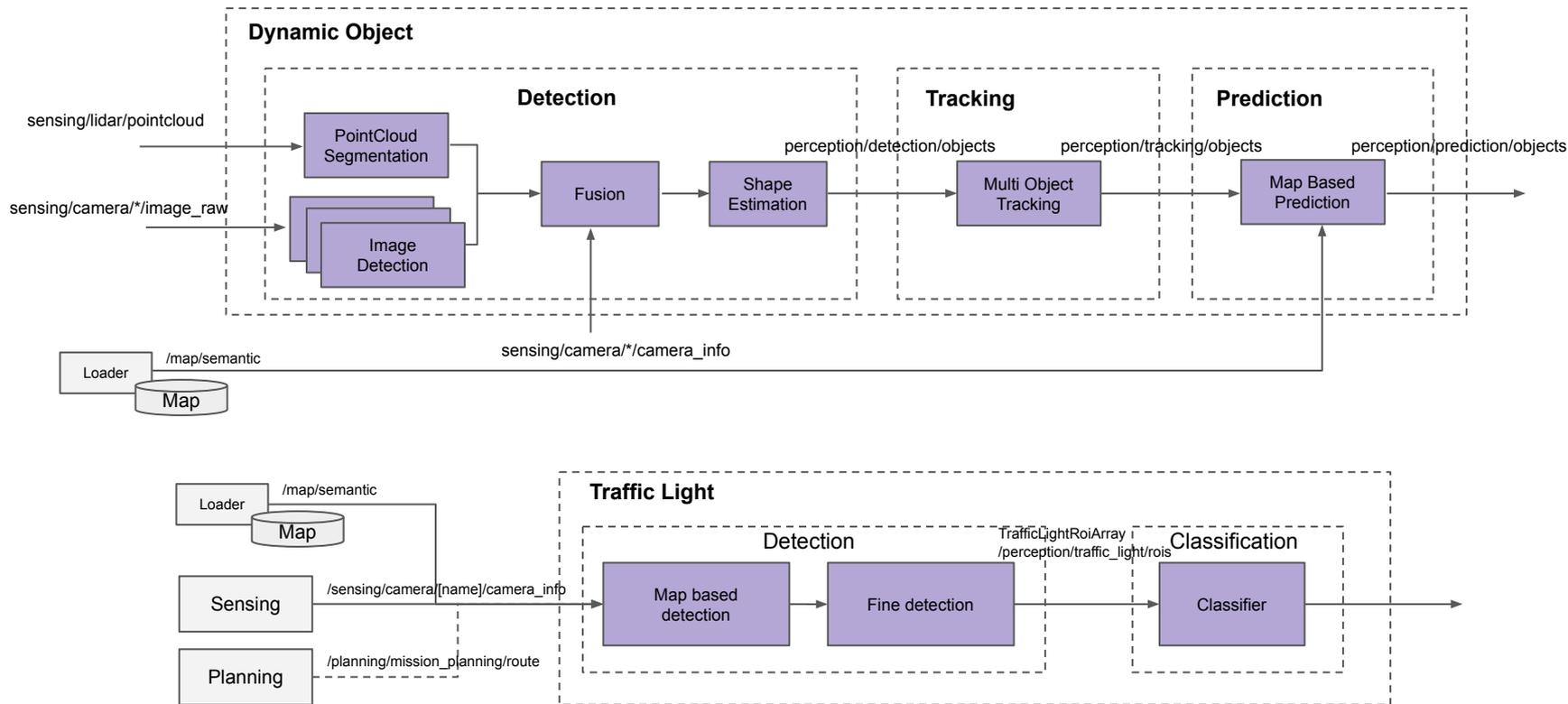
- **Distortion correction**
  - Corrects a distortion of the pointclouds due to an observation time gap
    - ⇒ Accuracy of the localization is improved
- **Ground filter**
  - Removes a ground from pointclouds (.ai has same feature)
- **Outlier filter**
  - Removes outliers by ring-base method ⇒ Reduce impacts of leaves and insects
- **Concat filter**
  - Integrates some pointclouds
  - Reduces an impact when a part of sensors stop
  - Corrects time gaps between each LiDAR with odometry ⇒ Accuracy of the localization is improved

# [Localization] Components



- **Pose initializer**
    - Automatic initial self position estimation by GNSS + Monte-Carlo method
  - **NDT scan matcher**
    - Uses an estimated value of EKF as an initial position of the scan matching ⇒ If scan matching was failed, localization can be returned
    - Performance and accuracy are improved (Open-MP implementation, accuracy improvement of the initial position, improvement of the gradient method, distortion correction of pointclouds, and etc.)
  - **Scan matching failure judgement**
    - Monitors statuses of scan matching based on a score
    - If score is lower than a threshold, an estimated result isn't output
  - **EKF localization**
    - Integrates the estimated self position of the scan matching and the velocity of CAN+IMU
    - If scan matching broke down, the vehicle can drive a certain distance with odometry only
  - **IMU Vehicle-twist fusion**
    - Uses a translation velocity of CAN and yaw rate of IMU ⇒ Accuracy of odometry is improved
  - **Localizer Diagnostics**
    - Monitors a status of a whole localization module (Temporary implementation)
- ※ Only NDT is implemented in Pose Estimator now.  
In future, other estimators like a white line recognition based one will be added.

# [Perception] Components



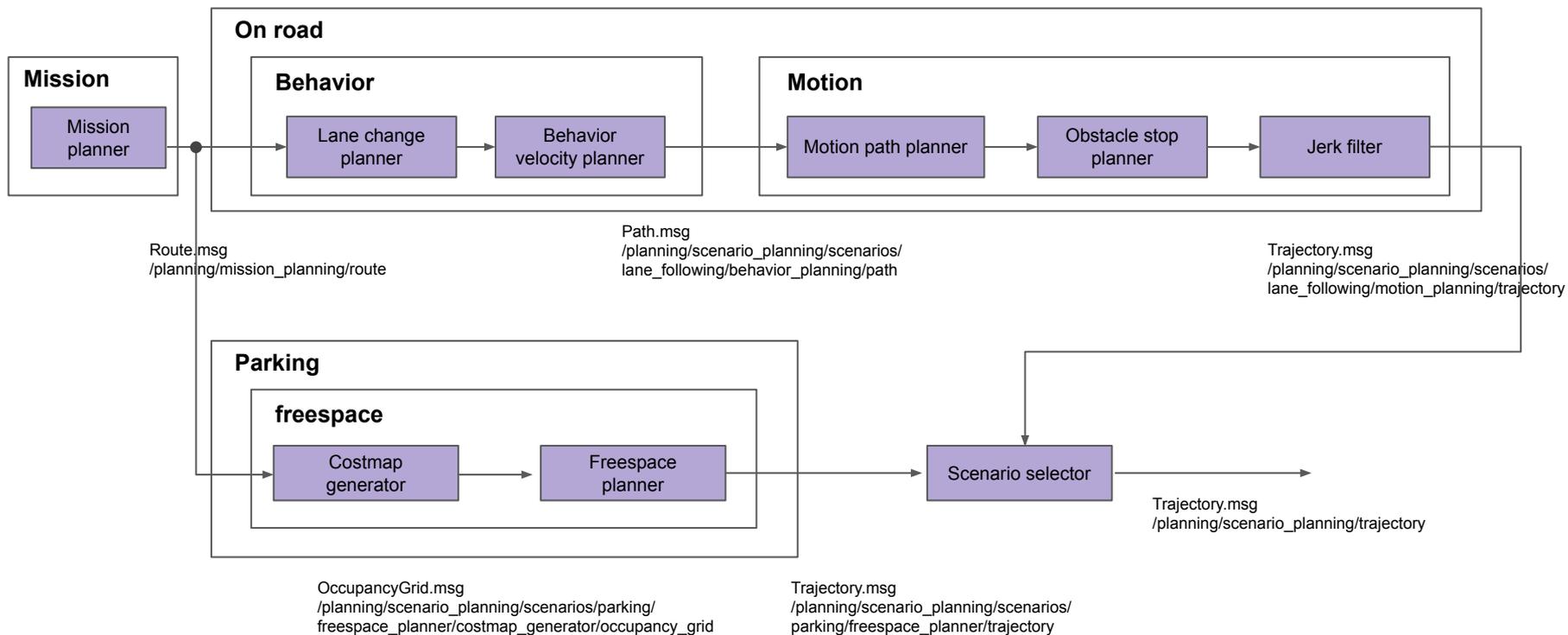
## Dynamic Object

- **Point Cloud Segmentation** : Splits pointclouds into object clusters
  - Becomes faster by algorithm improvement (It can work in real-time on CPU)
- **Image Detection** : Detects objects on an image as ROI
  - Becomes to work on 40FPS on Jetson AGX by Tensor RT and int8 quantization
- **Fusion** : Matches the result of Pointclouds Segmentation and the ROI of the image detection result
  - Matching accuracy is improved by using IoU
- **Shape Estimation** : Geometrically approximates the whole size of the objects by Point Cloud Segmentation
  - Shape estimations by each object class
  - Accuracy improvement by changing fitting algorithm of Bounding Box
- **Multi-Object Tracking** : Assigns IDs based on time series data, estimates the velocity and the acceleration, and removes outliers
  - Performance improvement by changing a tracking model by class labeling
  - Data association considering the class label and the size
- **Map Based Prediction** : Predicts moving path of the objects with the lane information in a map
  - Infers objects' intent of the behavior and estimates each reliability of the predicted moving paths.

## Traffic Light

- **Map Based Detection** : Extracts an ROI of the traffic light in camera image based on the self position and map data
  - Takes errors of a self position, a calibration and a hardware vibration into consideration
- **Fine Detection**
  - Extracts the ROI of the traffic light more precisely with a learner
- **Classifier** : Recognizes a state of the traffic light by a color information in the image
  - Reduces detection errors by fine noise removal process

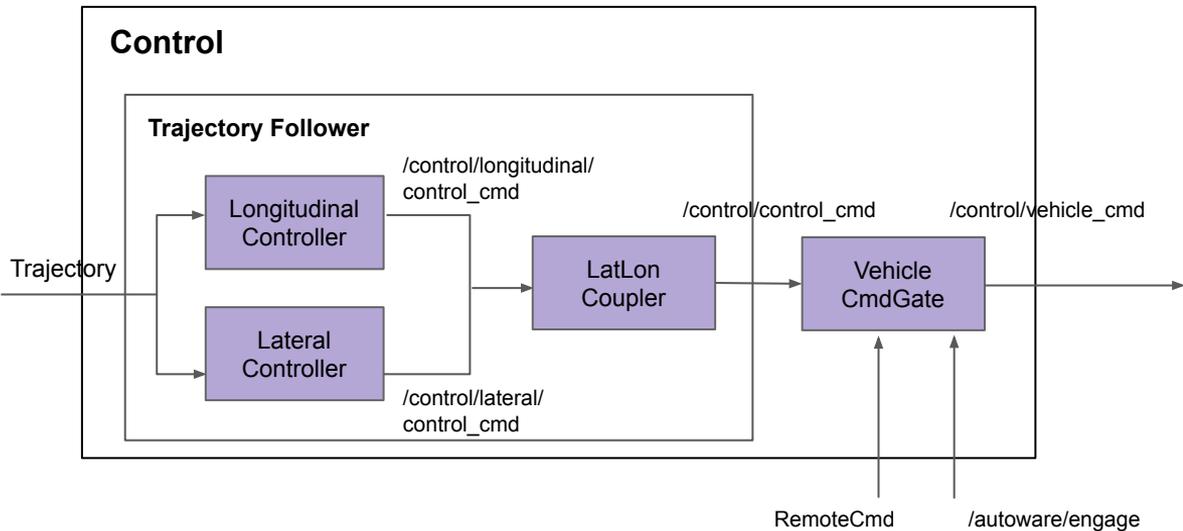
# [Planning] Components



- **Scenario selector**
  - Chooses and executes “lane\_following” or “parking” scenario according to the information of the mission
- **Mission planner**
  - Automatically searches the route from a self position to a goal via certain pass through points (by using lanelet function)
  - Automatically generates a route by using a map information
- **Lane change**
  - Judges a situation which needs lane change (Drive route limitation / street parking avoidance)
  - Judges whether a collision with a dynamic will happen obstacle and if a lane change is possible, it will be executed
- **Behavior Velocity Planner**
  - Plans a velocity based on traffic rules
  - Supports intersections, crosswalks, back-side check when turning right/left, stop with traffic lights and temporary stop line
  - Uses results of the perception (dynamic object information)
- **Obstacle avoidance (motion path planner)**
  - Plans a path to avoid the obstacle while driving ( $A^* + QP$ )
  - Generate a smooth path considering a drivable area and the obstacle
- **Obstacle stop (obstacle stop planner)**
  - Judges whether a collision with obstacle pointcoulds considering the vehicle shape, and fill a stop velocity
- **Jerk filter**
  - Smoothens the velocity under the limitation of max speed, acceleration and lateral acceleration (QP)
  - Enables to switch rapid/slow acceleration and deceleration plans



- **Longitudinal Controller**
  - Calculates a target velocity and a target acceleration by PID
  - Supports a gradient correction and start/stop at a slope
  - Supports a smoothly stop
- **Lateral Controller**
  - Calculates a steering angle and an angular velocity (pure pursuit or MPC)
- **LatLon Coupler**
  - Integrates the longitudinal and lateral control command values
- **Vehicle Cmd Gate**
  - Switches the some command values like “remote” and “auto”
  - Attaches a limitation for the control command
    - longitudinal/lateral acceleration, longitudinal/lateral jerk

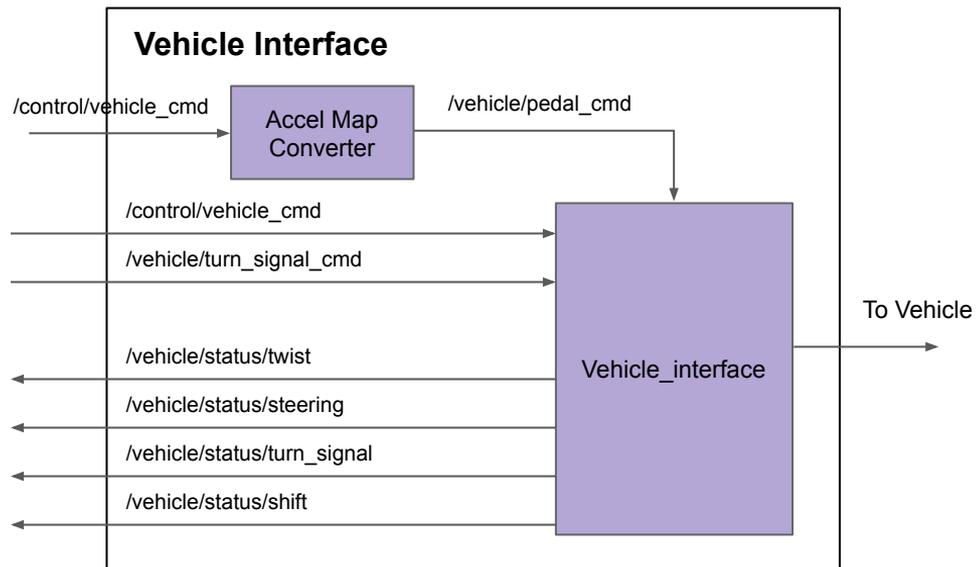


- In this implementation, longitudinal and lateral controls are separately calculated
- In future implementation, these might be calculated collectively

# [Vehicle Interface] Components



- **Accel Map Converter**
  - Converts a target acceleration to a vehicle specific accel/brake pedal value by Accel Map
  - If the vehicle\_interface supports a velocity/acceleration command, this converter isn't necessary
- **Vehicle Interface (vehicle specific)**
  - Interface between Autoware and a vehicle
  - Communicates control signals
  - Obtains the vehicle information



- **Web Controller**
  - Through this controller, we can engage and specify the max velocity and also use “Go Home” button, “lane change OK” button and sensor Hz monitor
- **Autoware State Monitor**
  - State monitoring (Initializing / WaitForEngage / Driving / ...)

# Backup agenda

---

- Design alternative
- **Module implementation**
  - Implementation details
  - **Achevements**

# Achievements (1/2)

- Automatic initial self pose estimation
- Robustness of self pose estimation improvement
  - CAN / imu fusion, EKF feedback, Gradient method algorithm improvement, pointclouds distortion correction, estimation speed improvement by Open-MP implementation
- 3D objects class recognition and shape estimation
- Performance improvement of dynamic objects tracking
- Moving path prediction of dynamic objects by using map information
- 360-degree sensing by the camera-LiDAR fusion
- Automatic lane change planning & decision
- Object avoidance (with lane change/ in the same lane)
- Intersection support (consider the oncoming vehicles and crossing vehicles)
- Person recognition at pedestrian crossing

## Achievements (2/2)

---

- Performance improvement of the traffic light recognition
- Blind spot check when turning right/left
- Route generation from current position to a destination (via specified passing points)
  - With this feature, “Go Home button” can be implemented
- Automatic parking
- Slope support
- Slow / rapid brake planning support
- Performance improvement of the vehicle control
- Automatic deceleration at curve
- Collision judgement considering a vehicle shape