2020.03

Version 1.0

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

Tier IV Intelligent Vehicle

Problem

- It’s diffcult 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

Tier IV Intelligent Vehicle

Example use cases that were considered during architecture design:

Module	Use cases
Sensing	● 360-degree sensing by the camera-LiDAR fusion
Perception	● Recognition of dynamic objects and traffic lights
Localization	● Robust Localization using multiple data sources
Planning	● Route planning, dynamic planning based on vector map (not only waypoint following) ● Automatic parking ● Object avoidance
Control	● High control performance on many kinds of vehicle-controllers

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

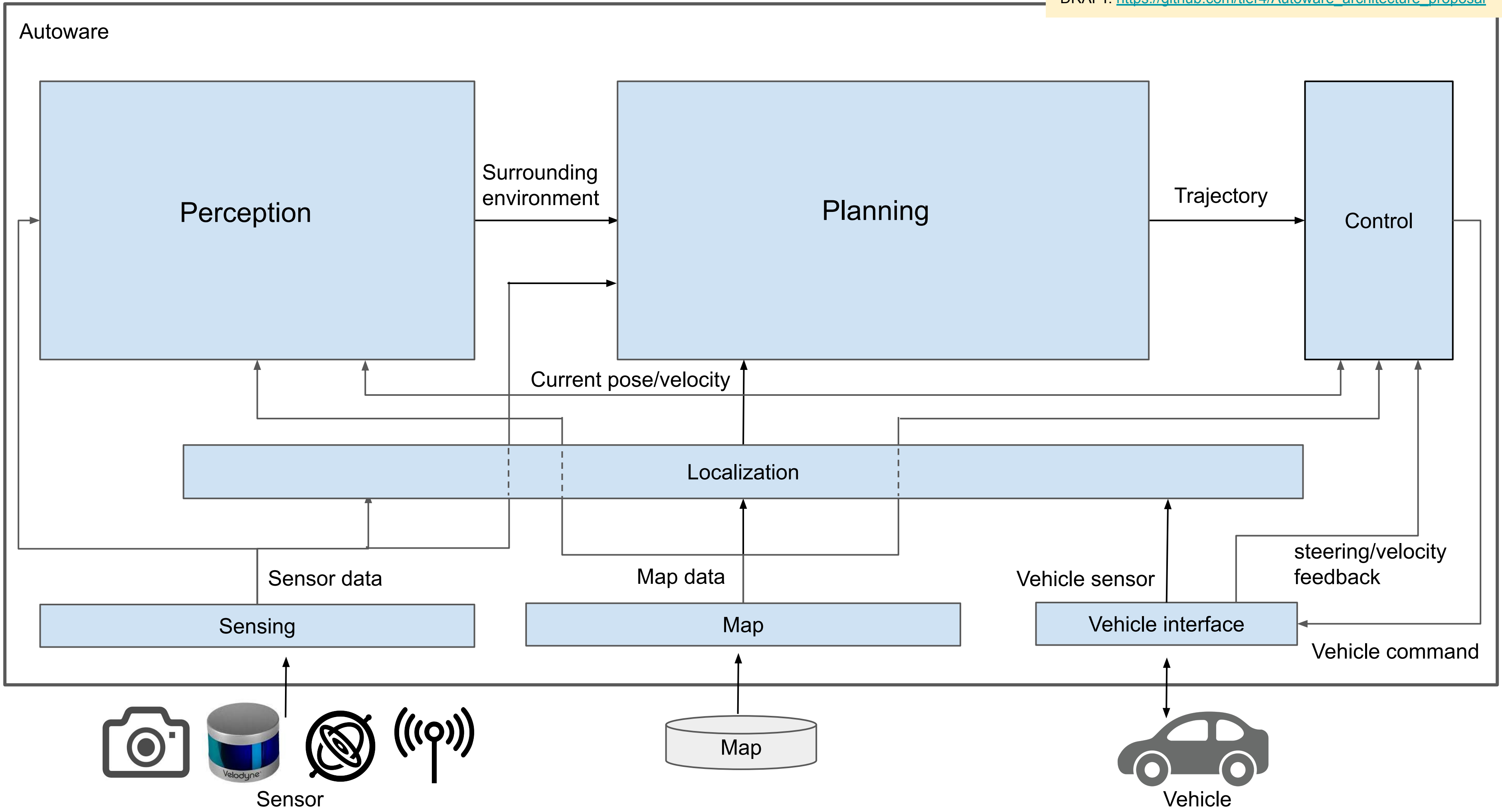
Demo video

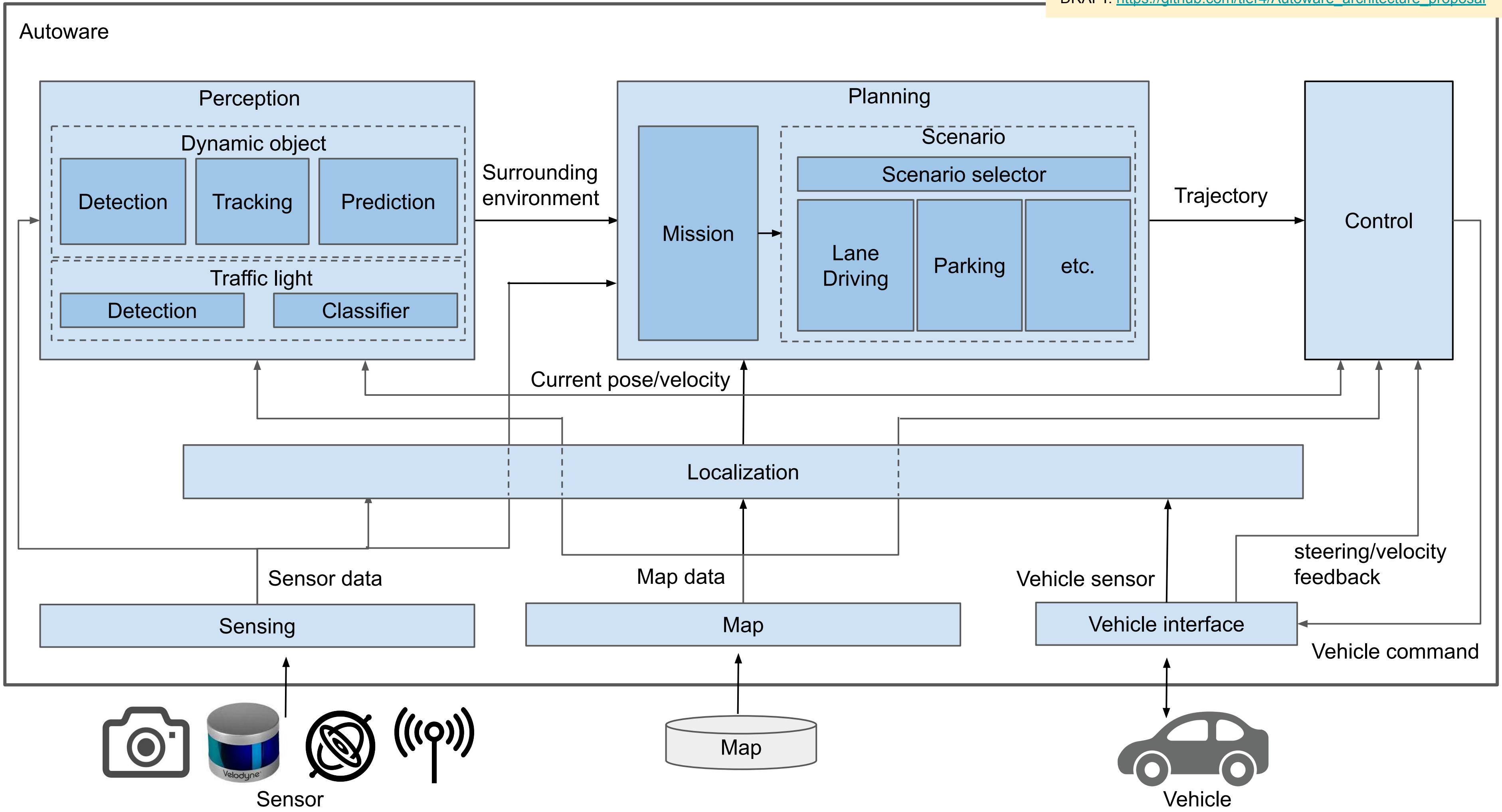
Tier IV Intelligent Vehicle

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

3 Architecture overview

Layered architecture
High level introduction of each module


Autoware
Perception
Surrounding
environment
Planning
Trajectory
Control
Current pose/velocity
Localization
Sensor data
Map data
Vehicle sensor
steering/velocity
feedback
Sensing
Map
Vehicle interface
Vehicle command
Velodyne
Map
Sensor
Vehicle

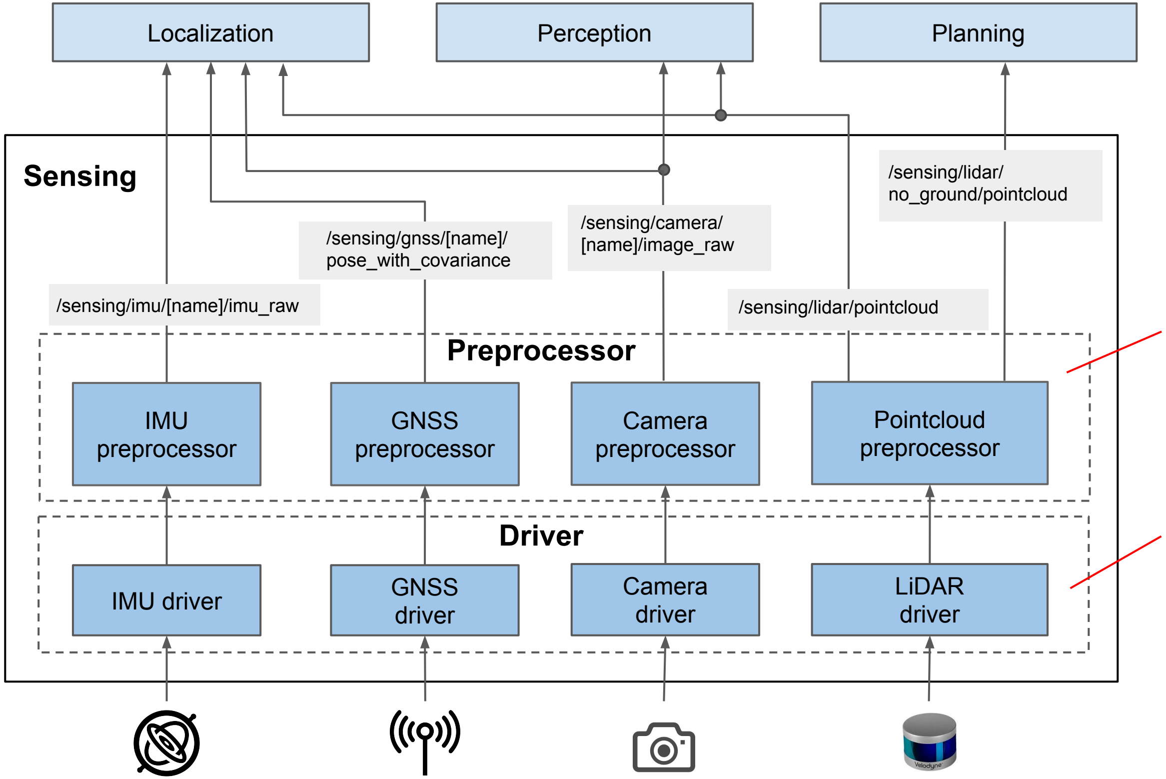

Autoware
Perception
Dynamic object
Detection
Tracking
Prediction
Traffic light
Detection
Classifier
Surrounding environment
Planning
Mission
Scenario
Scenario selector
Lane Driving
Parking
etc.
Trajectory
Control
Current pose/velocity
Localization
Sensor data
Map data
Vehicle sensor
steering/velocity feedback
Sensing
Map
Vehicle interface
Vehicle command
Velodyne
Sensor
Map
Vehicle

[Architecture] Sensing

Design Doc

Tier IV
Intelligent Vehicle

Role : Conversion of sensing data to ROS message



Examples of preprocessors

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

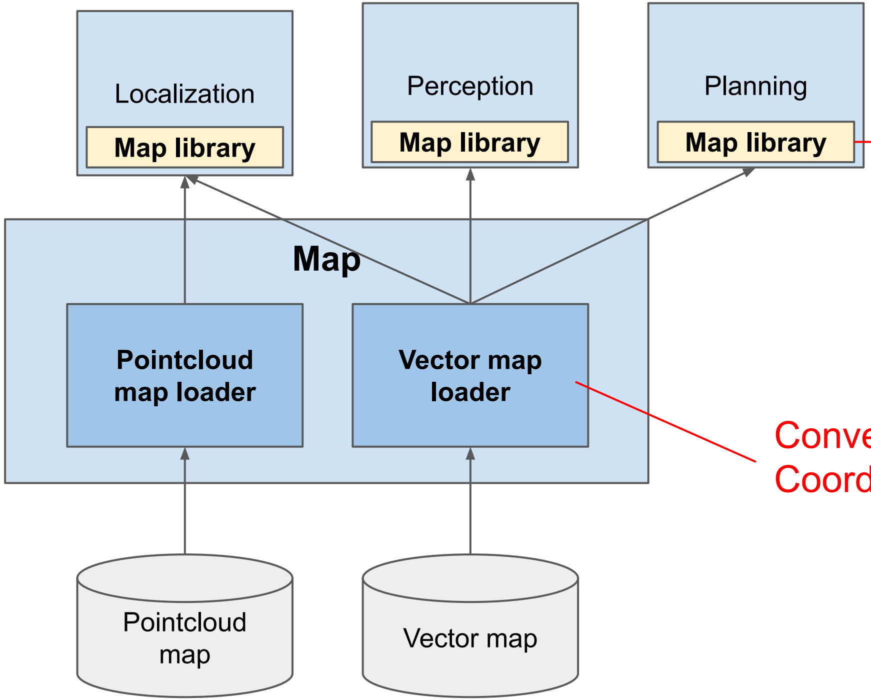
ROS drivers (mainly from upstream)

[Architecture] Map

Design Doc

Tier IV Intelligent Vehicle

Role : Distribute static environment information to other modules



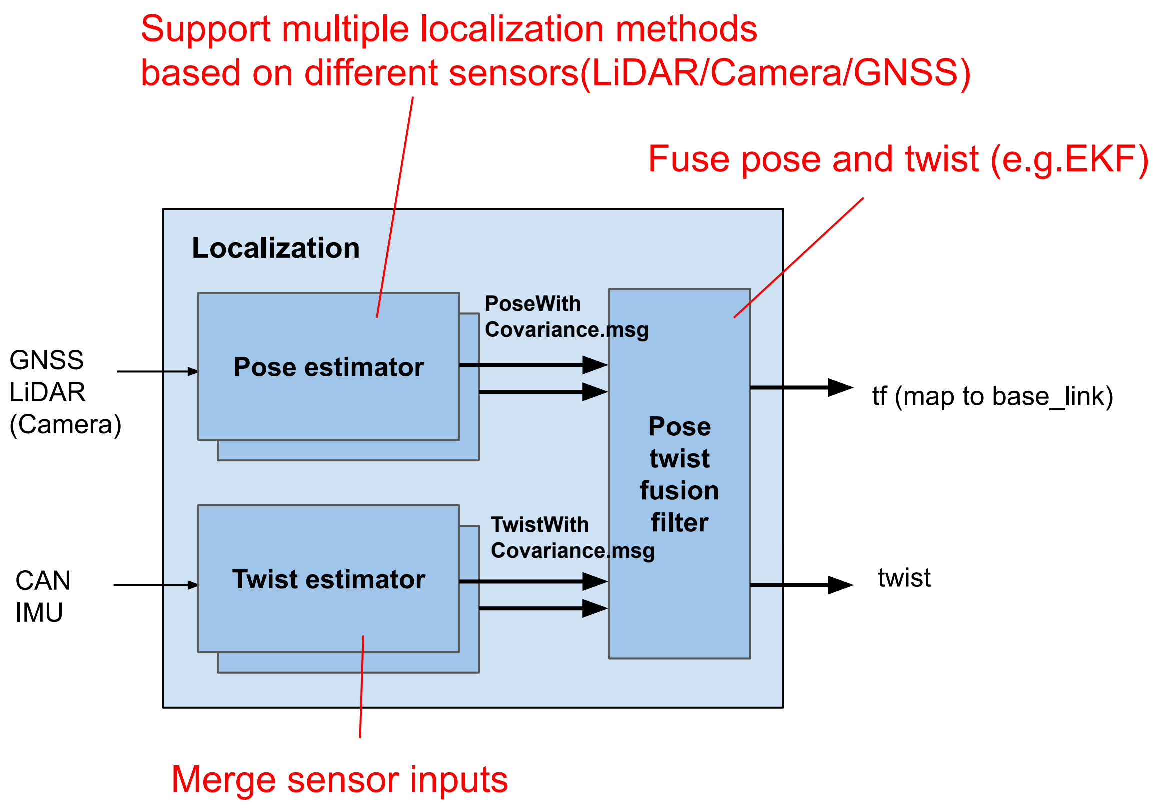
Map is distributed as a single message, and accessed through library API

Conversion to on-memory data structure
Coordinate transformation

[Architecture] Localization

Design Doc

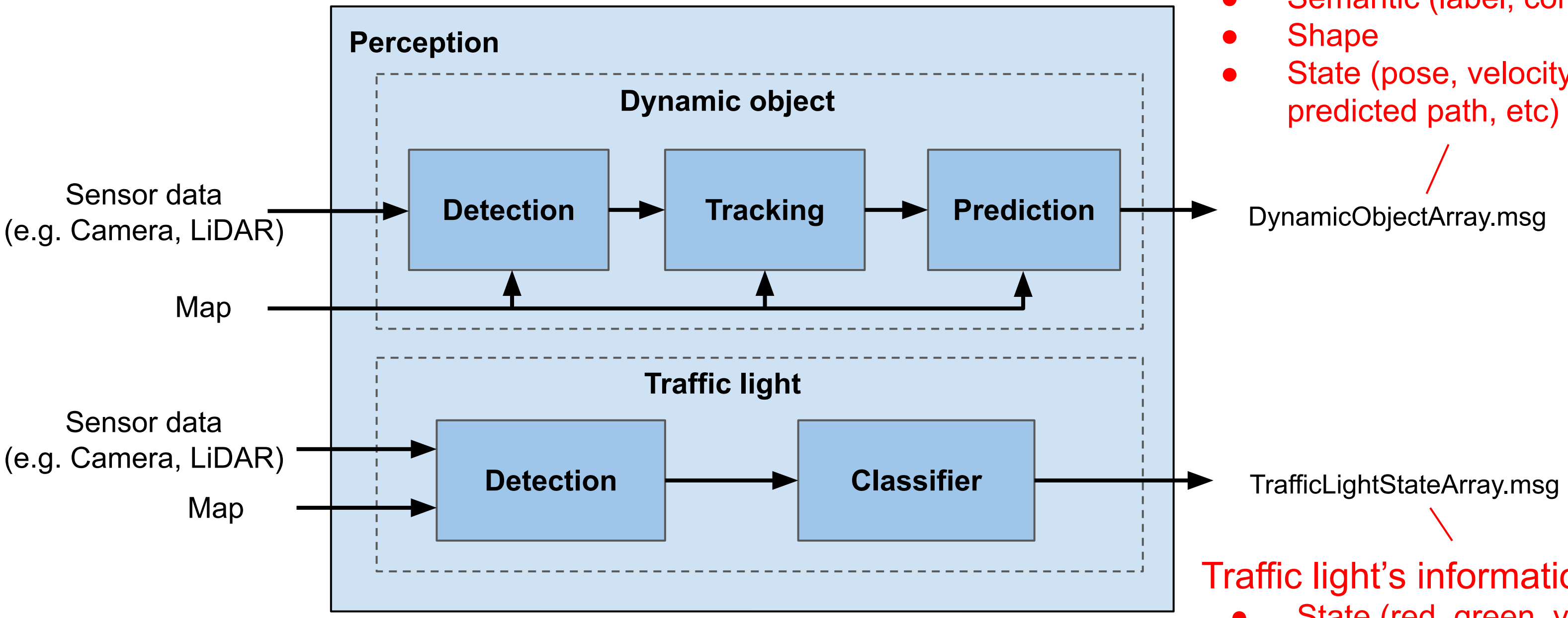
Role : Integration of each sensor data and estimation of self-pose and self-twist



[Architecture] Perception

Design Doc

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)

Traffic light's information including:

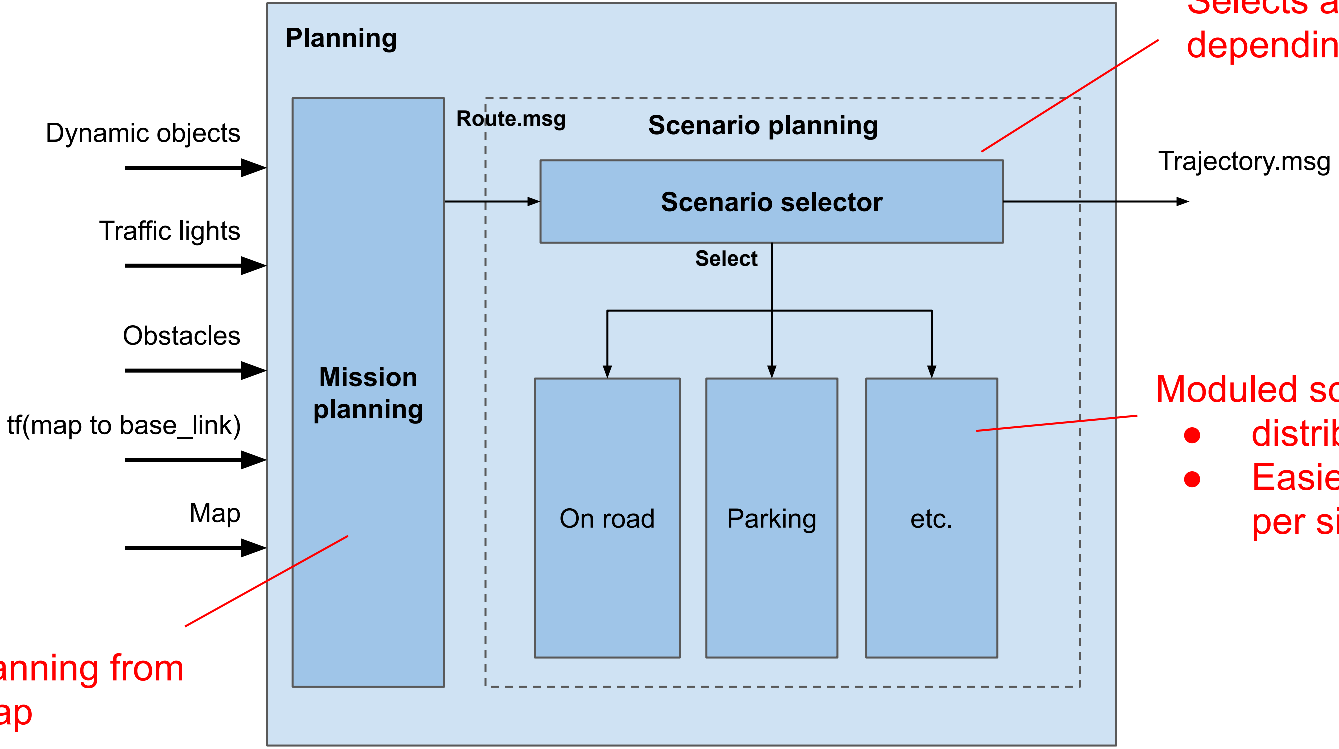
- State (red, green, yellow, etc)

Could be customized for unique traffic light types.

[Architecture] Planning

Design Doc

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



Selects appropriate module depending on situation

Moduled scenarios allow:

- distributed development
- Easier parameter tuning per situation

Route planning from vector map

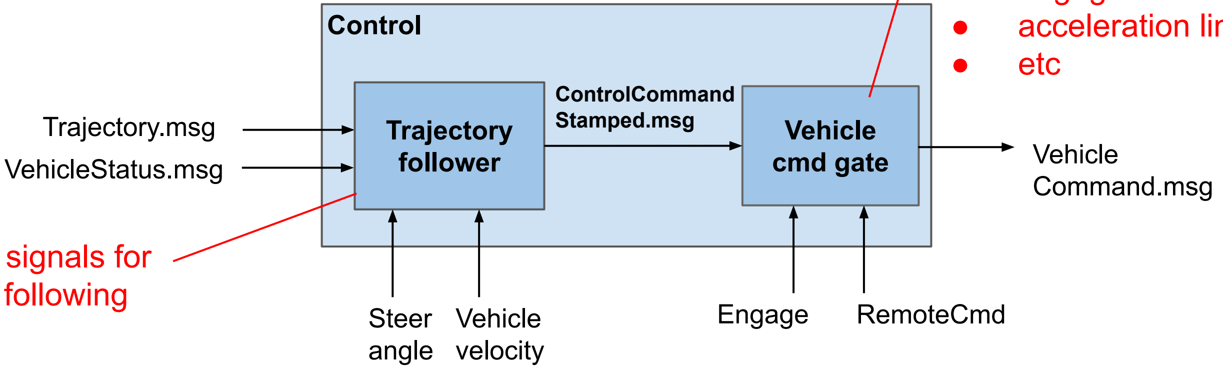
[Architecture] Control

Design Doc

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

Final gateway for commands to vehicle:

- engage
- acceleration limits
- etc



Generate signals for trajectory following

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)

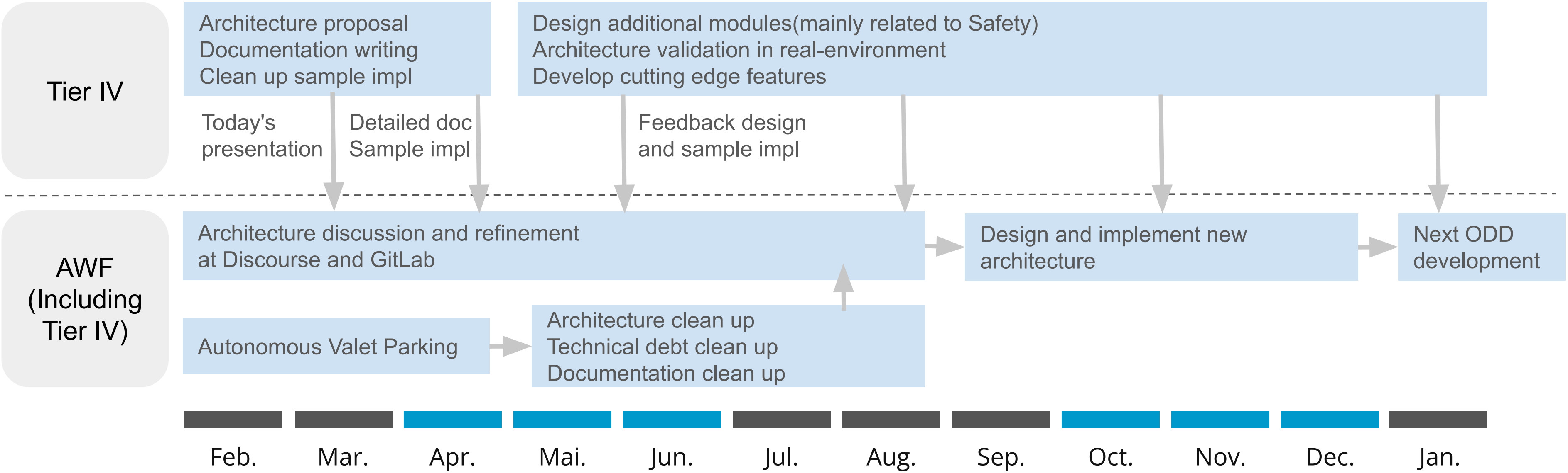
----

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

Conclusion

Tier IV Intelligent Vehicle

- 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

Appendix

Appendix - Table of Contents

Appendix - Table of Contents

- Design alternative
- Module implementation
  - Implementation details
  - Achievements

[Design alternative] Planning (1/2)

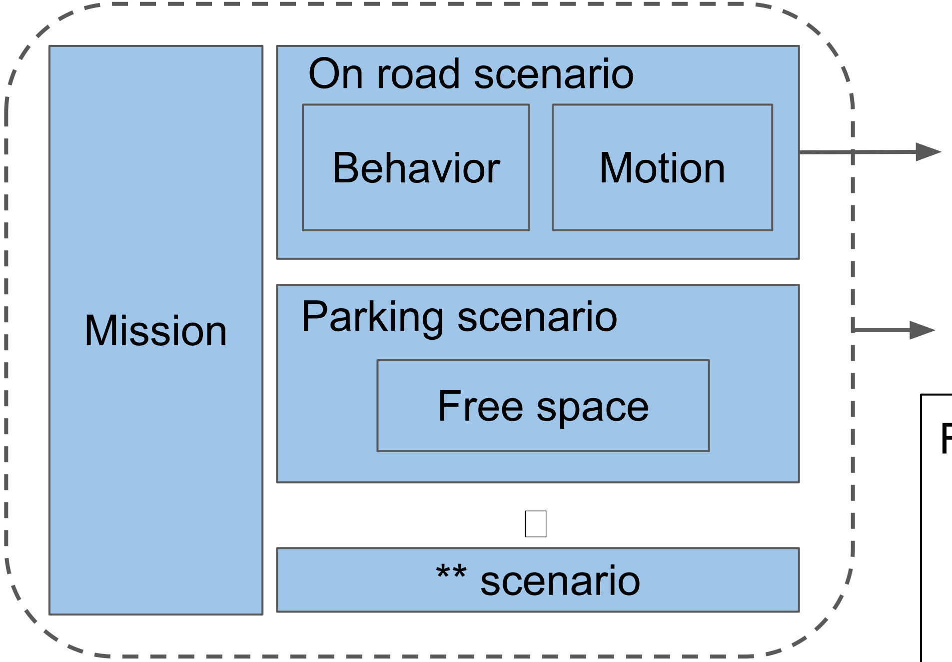
Tier IV Intelligent Vehicle

Pre-study of the planning architecture

Type	Structure	Pros.	Cons.
BOSS type (Junior type)	Mission / Behavior / Motion	・Structure is intuitive and easy to understand. ・Easy to handle separately.	・Behavior has to make a conservative decision in order to reduce discrepancies with a motion's decision.
CMU Dr. paper type	Mission / Trajectory	・It solves and optimizes both behavior and motion simultaneously to overcome a cons. of the BOSS type.	・Difficult to improve if any issue was found. ・Optimization of the whole autonomous driving functions is not realistic..
Victor Tango type	Mission / Behavior → Motion	・Behavior and motion are tightly coupled to overcome a cons. of the BOSS type. ・Easy to improve if any issue was found	・Difficult to take a new research result into the modules.
Apollo type	Mission / Scenario / Scenario	・Scenario is one layer upper concept. ・We can adopt prefered types of the planning structure according to each scenario.	・In the scenario, cons. such as things described above are still remained. ・If scenarios increased, switching mechanism can be complicated.

[Design alternative] Planning (2/2)

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



Lane following scenario : BOSS type

- It's clear and convenient to develop as OSS

Whole module : Apollo type

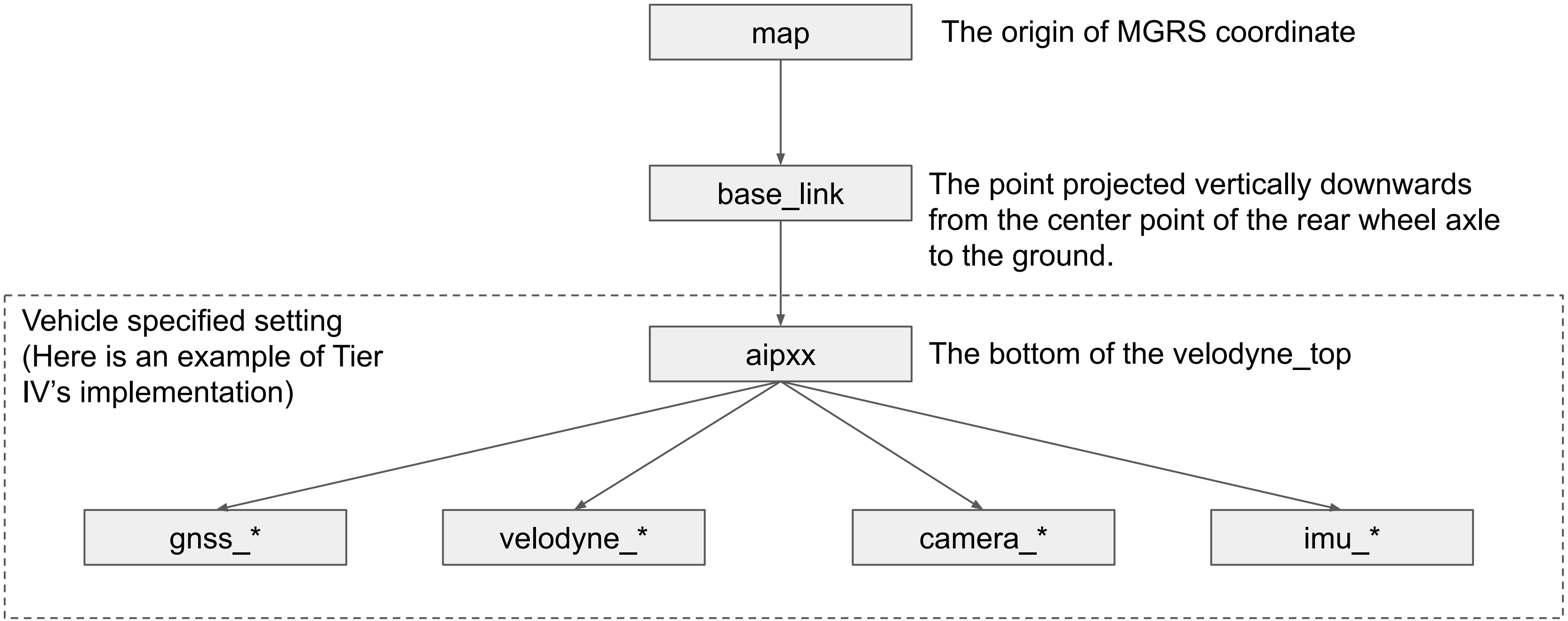
- New scenario can be added as needed

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.

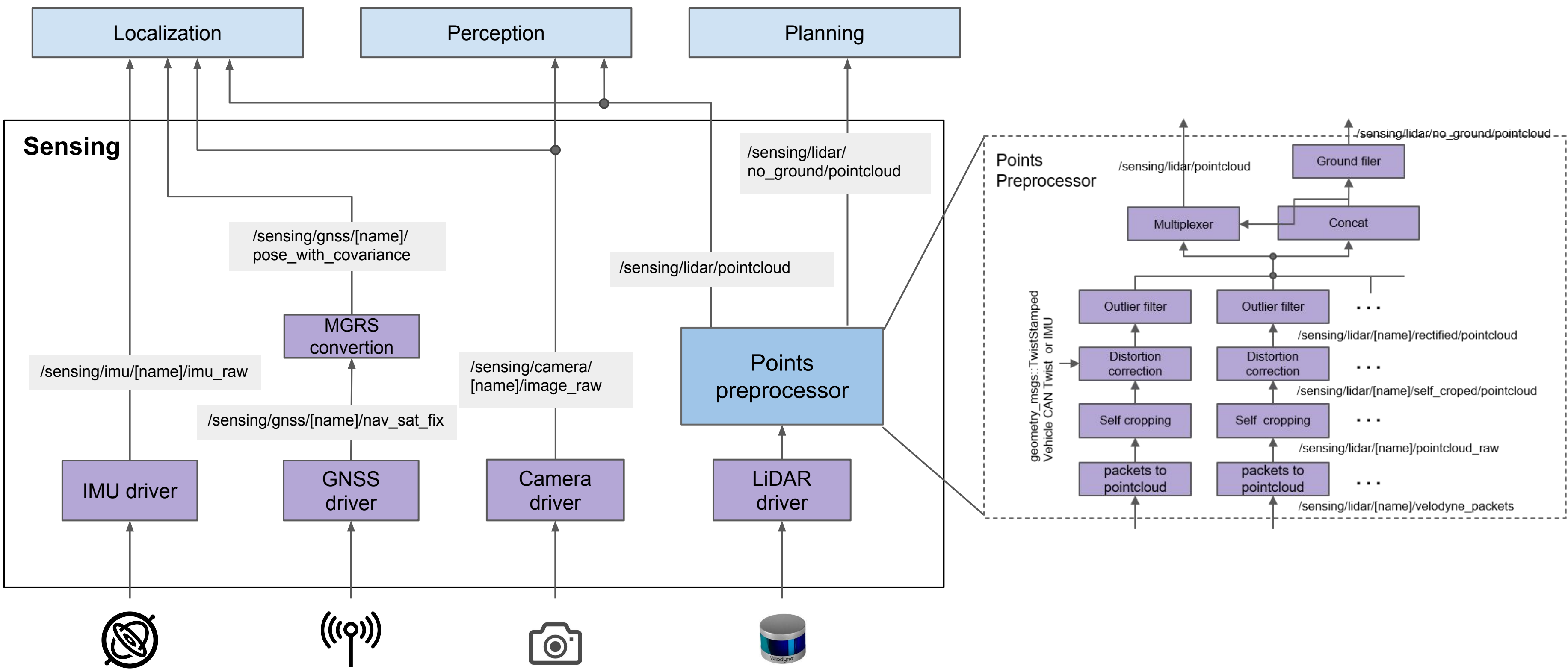
Appendix - Table of Contents

TF Tree



map
The origin of MGRS coordinate
base_link
The point projected vertically downwards from the center point of the rear wheel axle to the ground.
Vehicle specified setting
(Here is an example of Tier IV's implementation)
aipxx
The bottom of the velodyne_top
gnss_*
velodyne_*
camera_*
imu_*

[Sensing] Components



[Sensing] Feature

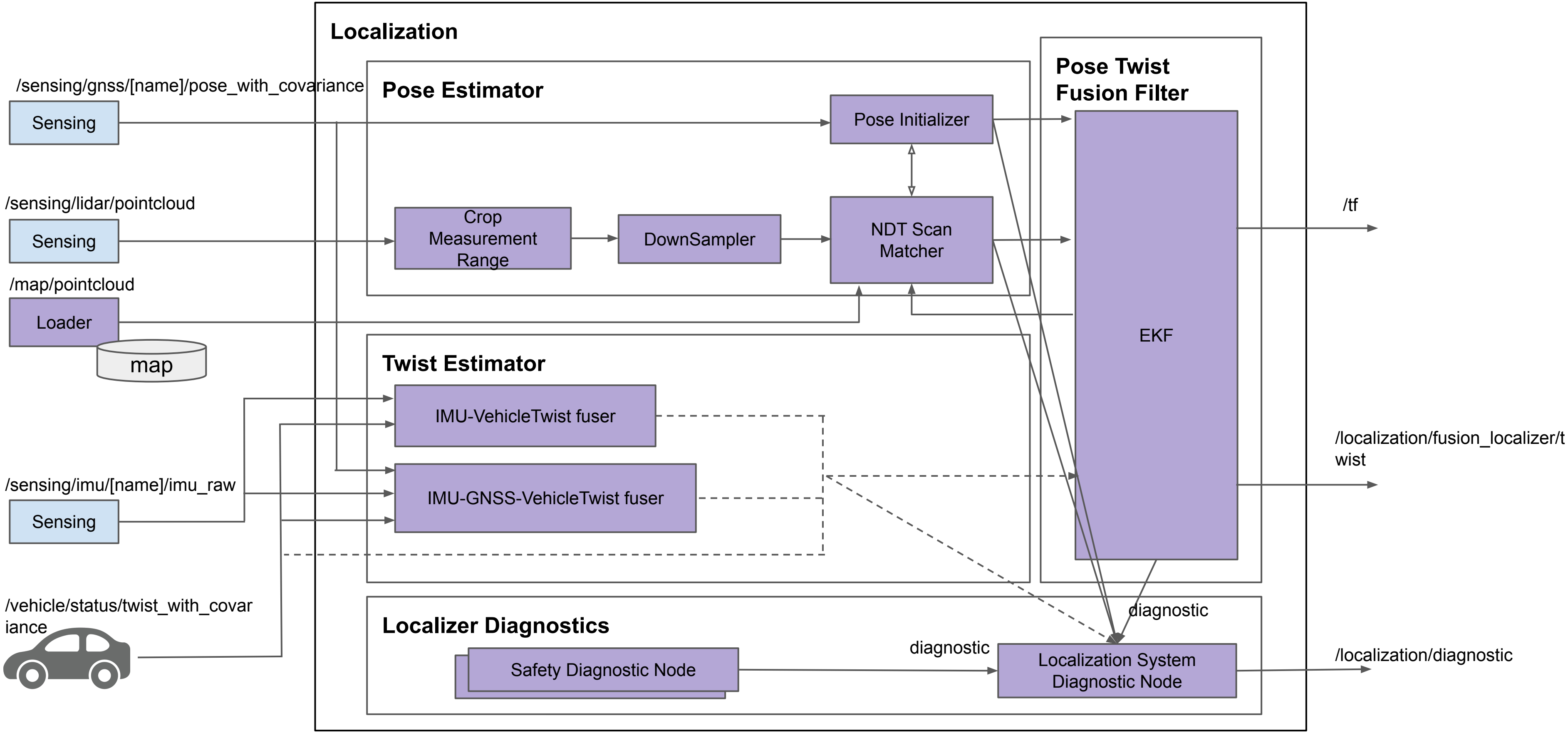
- **Distortion correction**
  - Corrects a distortion of the pointclouds due to an observation time gap ➪ Accuracy of the localization is improved

- **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



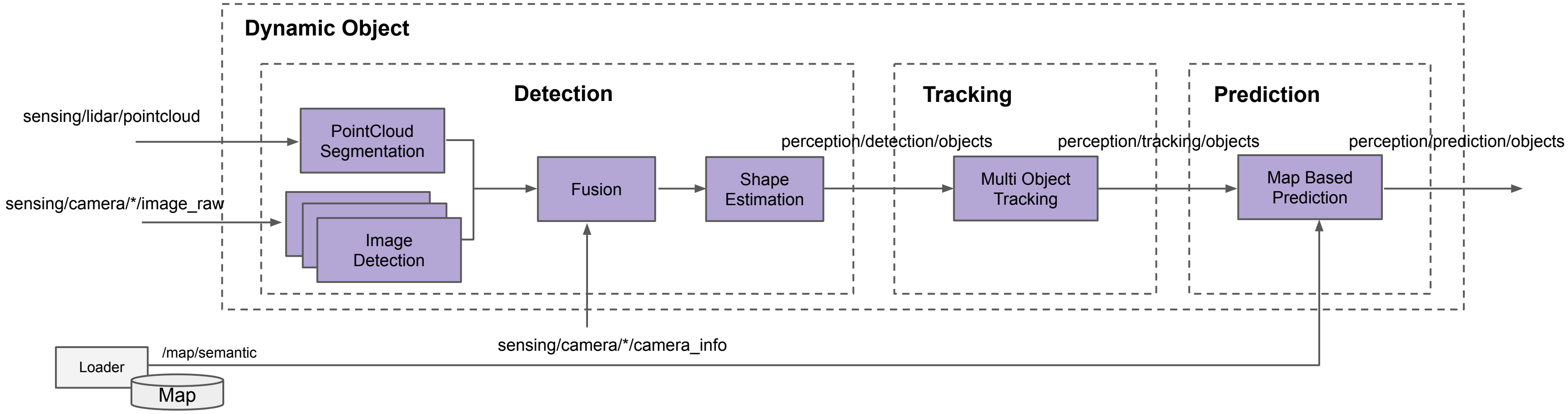
[Localization] Feature

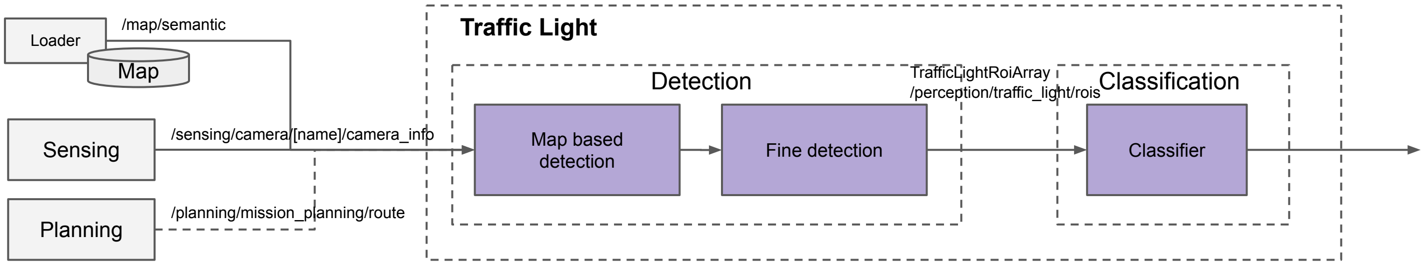
Tier IV Intelligent Vehicle

- **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





[Perception] Feature

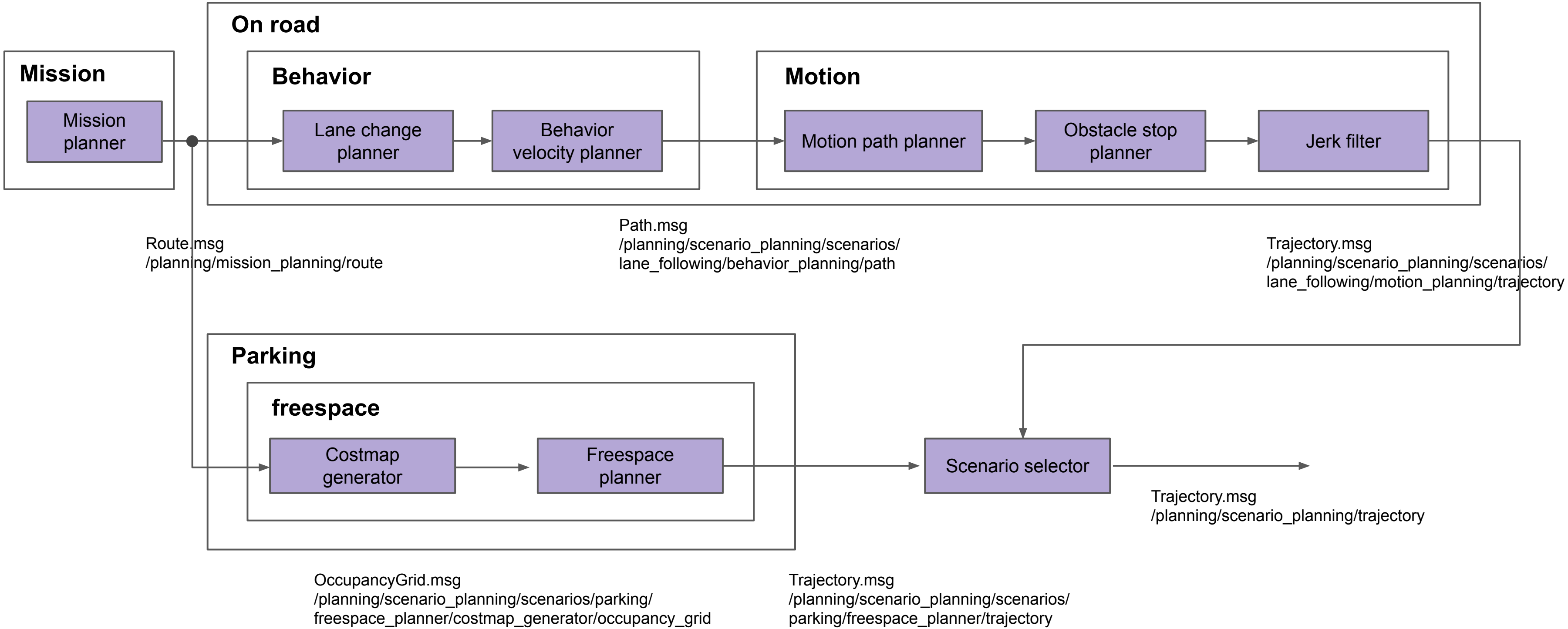
Dynamic Object

- **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 removement process

[Planning] Components

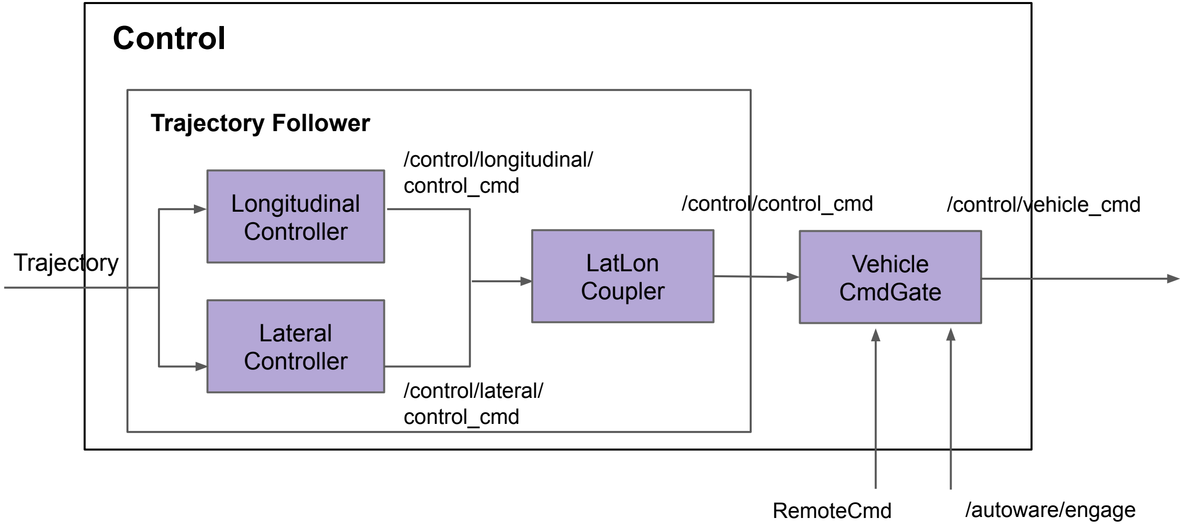


[Planning] Feature

- **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

[Control] Components

- **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

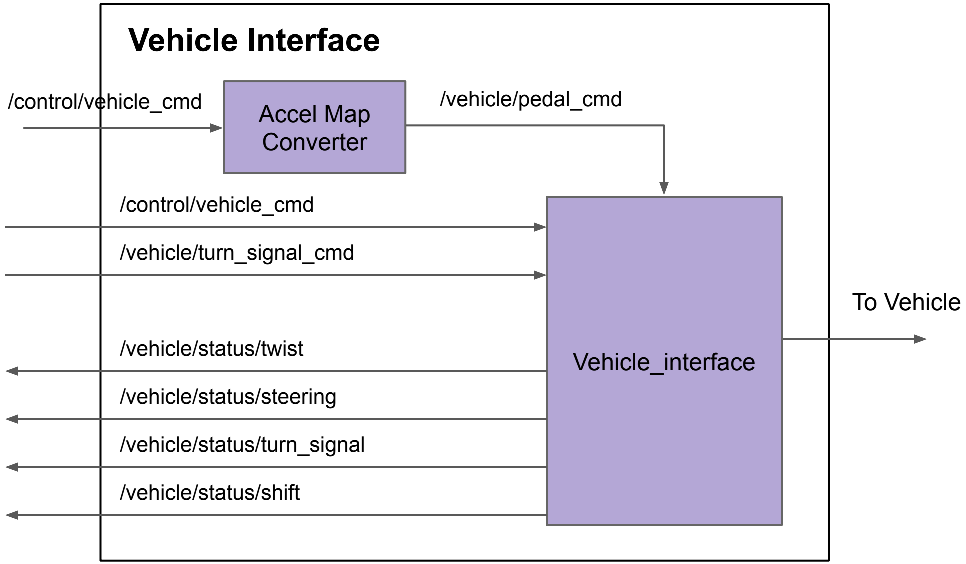


・In this implementation, longitudinal and lateral controls are separately calculated
・In future implementation, these might be calculated collectively

[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



Other features

Tier IV Intelligent Vehicle

- **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 / …)

Appendix - Table of Contents

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