Summary of related autoware pkgs

Visualization

1. Visualization of the map

corresponding packages or files:

> autoware/src/core/autoware_common/tmp/lanelet2_extension

three kinds of lanelet elements are conerted here:

- lanelet::Lanelet to **Triangle** Markers
- lanelet::LineString to **LineStrip** Markers
- TrafficLights to **Triangle** Markers

Each marker is defined in one function and inserted to the end of *map_marker_array* when the map is loaded. The way how the code is implemented can also be applied in the implementation of odd visualization. More details can be found here.

2. Visualization of the "ODD parameters"

corresponding packages or files:

> autoware/src/universe/autoware.universe/planning/behavior_path_planner/src/utilities.cpp & debug_utilities.cpp

Some of the visualizations are implemented for debugging in autoware. The following example shows the utility of it.

> drivable area boundary:
> This is defined as a ros topic and its type is "MarkerArray". The implementation is here.

Message definitions

corresponding packages or files:

> src/core/external/autoware_auto_msgs

All the messages and services files are stored in .idl files (*Interface Definition Language (IDL)*), which aims at "*achieving the CORBA goal of interoperability between different languages and platforms*".

The data structure of the map message is as following:

> Compact Message Definition of *MapPrimitive*
>
> ```
> int64 id
> string primitive_type
> ```
>
> Compact Message Definition of *HADMapSegment*
>
> ```
> sequence<autoware_auto_mapping_msgs::msg::MapPrimitive> primitives
> int64 preferred_primitive_id
> ```
>
> Compact Message Definition of *HADMapBin*
>
> ```
> std_msgs::msg::Header header
> uint8 map_format
> string format_version
> string map_version
> sequence < uint8 > data
> ```

There is still a problem not solved yet, the constants defined in idl file cannot be called in cpp files. An alternative way for this problem may be defining these constants manually.

Behavior path planner

corresponding packages or files:

> autoware/src/universe/autoware.universe/planning/behavior_path_planner

This package is responsible to generate

- **path** based on the traffic situation,
- **drivable area** that the vehicle can move (defined in the path msg),
- **turn signal** command to be sent to the vehicle - interface.

The basic algorithms for drivable area generation is explained here.

The drivable area generation flow:
generateDrivableArea(...)

-> getPathScope()

-> getNearestLaneId()

-> getClosestLanelet()(first search by distance then search by angle in the result)

-> calculate lane **boundary** coordinates

-> add lanes covers **initial and goal footprints**

(the drivabale area is basiclly generated from the planned path)

-> convert polygon to opencv type

-> create occupancygrid with opencv->convert opencv image to occupancygrid

Six different behaviors are implemented as separated modules in this package: Lane Following, Lane Change, Obstacle Avoidance, Pull Over, Pull Out and Side Shift.

A Design Tree is applied to manage which behavior should be applied in corresponding situations.