OpenPlanner Autoware proposed Packages/Nodes

By: Hatem Darweesh

Nodes Summary

1- Local Planner (dp_planner)

a) **Trajectory Generator ->** op_trajectory_genrator

b) **Trajectory Evaluator ->** op_trajectory_evaluator

c) **Behavior state planner ->** op_behavior_selector

d) **Contour KF Tracking ->** kf_contour_tracker

e) **Motion Predictor ->** op_motion_predictor (Behavior and trajectory prediction)

2- Global Planner (way_planner)

a) **Global Path Generator ->** op_global_planner

b) **Socket based HMI interface ->** op_hmi_interface (under development)

3- Multi vehicle simulator:

a) **Vehicle Simulator ->** op_car_simulator**:** simulated car using OpenPlanner

b) **Perception Simulator ->** op_perception_simulator: simulated object detection

c) **Traffic Sign Simulator ->** op_signs_simulator: simulated synchronized traffic lights

d) **Path Follower ->** op_waypoint_follower: simulated path following, localization, agent feedback + noise and delay (under development)

4- Testing Tools (OpenPlanner Utilities)

a) **Current pose TF generator from pose** -> op_pose2tf

b) **Playback ros bag with option to pause and step frame by frame** -> op_bag_player

1. Local Planner (dp_planner)

A) **Trajectory Generator ->** op_trajectory_genrator
B) **Trajectory Evaluator ->** op_trajectory_evaluator
C) **Behavior state planner ->** op_behavior_selector
D) **Contour KF Tracking ->** kf_contour_tracker
E) **Motion Predictor ->** op_motion_predictor

A) Trajectory Generator
op_trajectory_generator

A) Trajectory Generator: op_trajectory_generator

- **Intro:**
  - Generate smooth candidate trajectories (rollouts) for different driving scenarios (obstacle avoidance, lane change, branching)
- **Inputs :**
  - autoware_msgs::LaneArray: /lane_waypoints_array (global paths)
  - geometry_msgs::PoseStamped: /current_pose
  - geometry_msgs::TwistStamped: /current_velocity
  - autoware_msgs::CanInfo: /can_info
- **Outputs:**
  - Roll outs (local trajectory candidates)
    - autoware_msgs::LaneArray: /trajectory_candidates

op_trajectory_generator Node

- **Node Params:**
  - Car model (width, length, …)
  - Enable smoothing
  - Smoothing parameters
  - roll_in, roll_out, car_tip
  - Trajectory resolution
  - Max local planning distance
- **Requirements to successful run:**
  - Global path is generated and send by the topic "/lane_waypoints_array"
  - op_global_planner is the recommended route planner
  - Localization
  - map (server or loader or kml), for global plan to be generated

Visualization:

- Generated trajectory candidates. "/local_trajectories"

op_trajectory_generator Testing

1- Run dp_planner from runtime_manager/computing/motion planning/

- Global path data should be available.
- current _pose should be available either by live localization or simulation.
- Results could be observed on rviz.

2- generated trajectories will be saved to log path “/media/user/SimuLogs/TrajectoriesLogs/LocalPath_… .csv”

B) Trajectory Evaluator
op_trajectory_evaluator

B) Trajectory Evaluator: op_trajectory_evaluator

- **Intro:**
  - Calculate the collision cost for each roll out candidate generated by dp_planner and return best candidate trajectory information.
- **Inputs :**
  - autoware_msgs::LaneArray: /lane_waypoints_array (global paths)
  - autoware_msgs::LaneArray: /trajectory_candidates
  - autoware_msgs::BehaviorStateArray: /predicted_behaviors (optional)
  - autoware_msgs::DetectedObjectArray: /detected_objects
  - geometry_msgs::PoseStamped: /current_pose
  - geometry_msgs::TwistStamped: /current_velocity
  - autoware_msgs::CanInfo: /can_info
- **Outputs:**
  - Trajectory cost for the selected candidate (trajectory index, lane index, distance to collision, velocity of collision, actual trajectory id exists)
    - autoware_msgs::TrajectoryCost: /trajectory_cost *(message not implemented yet)*
    - autoware_msgs::lane: /final_waypoints
    - autoware_msgs::lane: /base_waypoints
    - std_msgs::Int32: /closest_waypoint

op_trajectory_evaluator Node

- **Node Params:**
  - Car model (width, length, …)
  - Planning horizon
  - Safety margins
  - Min following distance
- **Requirements to successful run:**
  - Localization
  - Global and local trajectories are generated, dp_planner should be working.
  - Detected objects with calculated contour is available.
  - With the availability of predicted behavior and trajectories, cost should be calculated accurately and smoothly.

Visualization:

- Selected trajectory. “/current_trajectory”

op_trajectory_evaluator Testing

1- Run op_trajectory_evaluator from runtime_manager/computing/motion planning/

- dp_planner should be working and candidate trajectories are available
- Results could be observed on rviz. (selected trajectory will be visualized with different color)

2- selected trajectory will be saved to log path "/media/user/SimuLogs/TrajectoriesLogs/CurrentPath_… .csv"

C) Behavior state Planner

op_behavior_selector

C) Behavior state Planner
op_behavior_selector

- **Intro:**
  - Decision making node which generates ego vehicle behavior state (forward, follow, stop, wait, slow down, swerve, yield, lane change, emergency stop ..).
- **Inputs :**
  - autoware_msgs::LaneArray: /lane_waypoints_array (global paths)
  - autoware_msgs::TrajectoryCost: /trajectory_cost
  - geometry_msgs::PoseStamped: /current_pose
  - geometry_msgs::TwistStamped: /current_velocity
  - autoware_msgs::CanInfo: /can_info
  - Road network map: from (autoware, .kml, .csv files)
- **Outputs:**
  - Behavior state plus information about stopping distance and stop line/traffic light ids, target general velocity.
    - autoware_msgs::BehaviorState: /behavior_state *(message not implemented yet)*

op_behavior_selector Node

- **Node Params:**
  - Car model (width, length, …)
  - Enables behavior states
  - Min distance to stop
  - Safety margins
- **Requirements to successful run:**
  - Localization
  - Trajectory evaluator
  - map (server or loader or kml), for global plan to be generated

Visualization:

- Behavior State for Ego car (text floating on top of the car)

op_behavior_selector Testing

1- Run op_behavior_selector from runtime_manager/computing/motion planning/

- Results could be observed on rviz

2- generated behavior and other information will be saved to log path “/home/hatem/SimuLogs/BehaviorsLogs/MainLog… .csv”

D) Contour Kalman Filter Tracking

kf_contour_tracker

D) Contour KF Tracking: kf_contour_tracker

- **Intro:**
  - Finds detected objects contours.
  - Track detected objects using kf, using priority based association circles.
  - Should be included in the perception packages.
- **Inputs :**
  - autoware_msgs::CloudClusterArray: /cloud_clusters
  - geometry_msgs::PoseStamped: /current_pose
  - geometry_msgs::TwistStamped: /current_velocity
  - autoware_msgs::CanInfo: /can_info
- **Outputs:**
  - Tracked Objects with enough information for Local Planning
    - autoware_msgs::DetectedObjectArray: /detected_objects

kf_contour_tracker Node

- **Node Params:**
  - Max number of contour points
  - Priority circles numbers
  - Enable never forget.
  - Max association distance
- **Requirements to successful run:**
  - Localization
  - Object Detection (euclidean_cluster)
  - Ego vehicle current pose/velocity

Visualization:

- Moving objects (ID, Velocity, color of priority region)
- Contours for all objects
- Priority regions circles

kf_contour_tracker Testing

1- Run kf_contour_tracker from runtime_manager/computing/detection/lidar_detector

- Lidar data should be available from rosbag playback or live sensor data.
- Tracked objects and contours should be consistent with the detected obstacles from euclidean clustering node.
- Result could be observed visually on rviz.

E) Motion Predictor
op_motion_predictor

E) Motion Predictor: op_motion_predictor

- **Intro:**
  - Predict moving obstacles local **trajectory**(s)
  - Predict moving obstacles **behavior** state (follow, stop, overtake, yield, branch, …)
- **Inputs :**
  - autoware_msgs::DetectedObjectArray: /detected_objects
  - Road network map: from (autoware, .kml, .csv files)
  - Prediction time (published from trajectory evaluator node)
- **Outputs:**
  - autoware_msgs::BehaviorStateArray: /predicted_behaviors *(message not implemented yet)*

op_motion_predictor Node

- **Node Params:**
  - Max prediction distance
  - Default prediction time
  - use vector map
- **Requirements to successful run:**
  - Object tracking (euclidean_lidar_track, kf_lidar_track, pf_lidar_track, kf_contour_track)
  - map (server or loader or kml), if use_vector_map parameter is enabled.

Visualization:

  - Object behavior (text floating on object)
  - Predicted local trajectory for moving objects

op_motion_predictor Testing

1- Run motion_Predictor from runtime_manager/computing/prediction/

- Lidar data should be available from rosbag playback or live sensor data.
- Tracked objects should be available with relatively accurate velocity and orientation.
- Prediction should find one or more expected moving trajectory for detected objects.
- Results could be observed on rviz.

2- Results could be simulated by using simulated cars (op_simulator) and simulated lidar based detection (op_simulator_perception).

- Actual velocity and planned trajectory could be compared to predicted ones to calculate prediction accuracy.

2. Global Planner (way_planner)

A) **Global Path Generator ->** op_global_planner
B) **Socker based HMI interface** -> op_hmi_interface (under development)

A) op_global_planner

- **Intro:**
  - Global Planner that receives start goal position and general a global reference paths using the lanes centerlines from vector map. the plan include lane change information and upcoming traffic information (stop line , traffic light, stop line, stop sign, …)
- **Inputs :**
  - **Start Pose: (x,y,z,theta)**
    - If current_pose doesn’t exist, it uses rviz “2d pose estimate”.
  - **Goal Pose (s): (x,y,z,theta)**
    - If enableRvizInput is true, it uses rviz “2d nav goal” to select one or more goal points.
    - If enableRvizInput is false, op_global_planner try to find .csv file in SimuLog folder containing destination points.
    - In case of enableRvizInput is true .csv file with destinations is saved after shutting down op_global_planner node.
  - **Road Network Map:**
    - Construct map from Vector Map loaded by Autoware
    - Construct map from Vector Map .csv files
    - Construct map from Vector Map .kml file
- **Outputs:**
  - **Reference Trajectories** (autoware_msgs::LaneArray) “lane_waypoints_array”
    - Waypoint (pose,orientation, laneID, ref_v, lane change cost, stop line id, left lane id, right lane id, branching direction)

op_global_planner Node

- **Node Params:**
  - reference path resolutions - path Density- (distance between each two waypoints)
  - enable smoothing
  - enable lane change
  - mapping source (map_loader, map_server, kml)
  - kml map path
  - enable rviz input
- **Requirements to successful run:**
  - the availability of tf
  - map (server or loader or kml)
  - start point and goal point
  - start point and nodes list.

Visualization:

- Map to visualize “vector_map_center_lines”
- global paths to visualize “global_waypoints_mark”
- start point “Global_StartPoint”
- goal points “Global_GoalPoints”
- nodes list “Goal_Nodes_Points”

op_global_planner Testing

1- Run op_global_planner from runtime_manager/computing

- In rviz "vector_map_center_lines" will show the map lane centers in orange color, if not map loading is failed , check map path and loaded type from parameters.

2- If current_pose message is published (due to simulation or live localization). Only goal position needed to be set. Otherwise user need to select start position from rviz "2d pose estimate". User can check for current_pose and current_velocity values, also make sure to use runtime_manager/computing/localization/autoware_connector/vel_pose_connect.

3- There are two options to select goal(s) position.

- From destination log file, it is saved automatically after turn of op_global_planner node. Its default path is "/home/user/SimuLogs/SimulationData/EgoCar.csv". User needs to disable parameter "Rviz Goals".
- By selection one or more goal destinations using rviz "2d nav goal". User needs to enable parameter "Rviz Goals".
- In both cases global path for the first goal point showd appear on rviz, by visualizing the topic "global_waypoints_mark". If not user can check error message in the terminal window for more information.

4- generated global path(s) will be saved to log path "/home/user/SimuLogs/GlobalPathLogs/GlobalPath … .csv"