Behaviour Planner

How to launch Behaviour Planner separately

> `roslaunch agv behaviour_planner_node.launch`
>
> **Can just type "be" and (tab)(tab) to auto fill**

Nodes

1. Publishers and Subscribers

(1) behaviour_planner_node.cpp

Subscribe to	Publish to
/odometry/filtered	/nav_cmd_vel
/output_path	/front_left_obs_topic
/desired_steering_angle	/front_right_obs_topic
/present_steering_angle	/back_left_obs_topic
/collision_detector	/back_right_obs_topic
/special_waypoint	/stopwall_marker (rviz)
/obstacles (3D LiDAR)	
/obstacles2 (Front 2D LiDAR)	
/obstacles3 (Left & Right Back 2D LiDARs)	
/lane_info_topic	

(2) collision_detector.cpp

Subscribe to	Publish to
/odometry/filtered	/collision_detector
/obstacles (3D LiDAR)	/predicted_trajectory (rviz)
/obstacles2 (Front 2D LiDAR)	
/obstacles3 (Left & Right Back 2D LiDARs)	
/present_steering_angle	

(3) lane_publisher.cpp

Subscribe to	Publish to
/odometry/filtered	/lane_info_topic
/path_filename_topic	/left_boundary_xa_topic (rviz)
	/left_boundary_ax_topic (rviz)
	/right_boundary_topic (rviz)
	/centerline_topic (rviz)
	/waypoints_info_topic (rviz)
	/special_waypoint

2. Features

(1) behaviour_planner_node.cpp

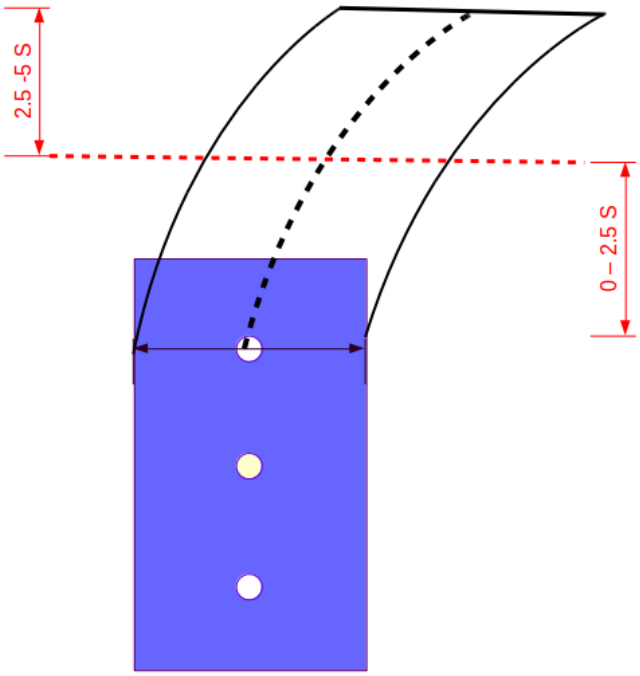
- **Gets desired modes from *collision_detector* and *special_waypoint*** (red words are the name of the file)
- **Publish final mode to /nav_cmd_vel** (choose the most conservative mode -- smaller of `collision_mode` and `waypoint_mode`)
- **6 Possible States**: (Capital names are the constants in code, the number in bracket is the default value of the constants)
  - *DYNAMIC_OBSTACLE_STOP*
    - When dynamic obstacle is detected to collide with buggy in 2.5 - 5 s, and moving faster than buggy's current speed (Currently, we didn't differentiate the directions of moving, this can be of the future improvement)
    - **Brake**: DYNAMIC_OBSTACLE_BRAKE (0.7)
    - **Desired speed**: 0 m/s
    - Stop for 3 seconds before executing new path
  - *STATIC_OBSTACLE_STOP*
    - When static obstacle is detected in buggy's projected path and will collide with buggy in COLLISION_TIME_THRESHOLD (2.5 s)
    - **Brake**: STATIC_OBSTACLE_BRAKE (0.7)
    - **Desired speed**: 0 m/s
  - *ENDPOINT_STOP*
    - Nearing endpoint
    - **Brake**: ENDPOINT_BRAKE (0.5)
    - **Desired speed**: 0 m/s
  - *SLOPE_SLOWDOWN* (Only for slope down)
    - Slope detected
    - **Brake**: SLOPE_BRAKE (0)
    - **Desired speed**: SLOPE_SLOWDOWN_SPEED (1.2 m/s)
  - *NORMAL_SLOWDOWN*
    - A static obstacle is detected in buggy's projected path but will not collide with buggy in COLLISION_TIME_THRESHOLD (2.5 s), or
    - Turning point detected
    - **Desired speed**: NORMAL_SLOWDOWN_SPEED (1.5 m/s)

- *NORMAL_SPEED*
  - **Desired speed**: NORMAL_CRUISE_SPEED (2.5 m/s)

(2) collision_detector.cpp

Buggy's projected path:

- Position of the buggy for the next 5 seconds is predicted using `(current_x, current_y, current_speed, current_steering_angle)`
-



  - The `desired_mode` is initialized as `NORMAL_SPEED`
  - Coordinates of the buggy's center of gravity (COG) and frontlink are determined by transforming the baselink coordinates `(current_x, current_y)`
  - Project the path 5 seconds forward from the frontlink.
  - Use front 2D and 3D LiDARs to detect obstacles and calculate the distance between each predicted point on the projected path with obstacles (considering a safety margin). If their distance is smaller than a set threshold, the `collision_flag` is set to *true*.
  - If `collision_flag` is *true* and obstacles are static, `desired_mode` is updated to `NORMAL_SLOWDOWN` or `STATIC_OBSTACLE_STOP` based on the estimated time to collision (In the above image, 0 - 2.5 s is the "stopping" zone, 2.5 - 5 s is the "slowdown" zone).
  - If the `collision_flag` is *true* and the maximum speed of the obstacles (speed detected by 2D and 3D LiDARs) is faster than buggy's `current_speed` and the time to collision is less than 2.5 s, `desired_mode` is updated to `DYNAMIC_OBSTACLE_STOP`.
  - `desired_mode` is published to /collision_detector topic.

Back Left and Right 2D LiDARs

- The other two 2D LiDARs at the back are specially used for detecting "dynamic obstacles".
- The distance between the dynamic obstacles with the baselink of the buggy and center of gravity including a back safety margin) is calculated.
- If the `collision_flag` is *true* and the dynamic obstacle speed is faster than the buggy's `current _speed`, the `desired_mode` is updated to `DYNAMIC_OBSTACLE_STOP`.

Dynamic obstacle predicted path

- Assume the detected obstacle is moving in a straight line.
- Predict the path of the obstacle for the next 5 seconds and display on rviz.
- Future Improvement Part: This path can be used to improve the algorithm of dynamic obstacle detection and behaviour.

(3) lane_publisher.cpp

- **Display lane boundaries on rviz**
  - Read coordinates from lane files (in lanes folder):
    - left_boundary_xa
    - left_boundary_ax
    - right_boundary
- **Publish lane info to** /lane_info_topic
  - Lane info includes *{x, y, dx, dy, s, left_width, right_width, far_right_width, special_point}* for each waypoint
  - Refer to Lanes Documentation for details
- **Display waypoints on rviz**
  - The red arrows indicate target heading of buggy
- **Check for special waypoints**
  - Detect if buggy has entered the radius of endpoint, turning or slope points (determine the distance between buggy's baselink and the special waypoint)
    - If endpoint detected, `desired_mode` is set to `ENDPOINT_STOP`
    - If turning point detected, `desired_mode` is set to `NORMAL_SLOWDOWN`
    - If slope point detected, `desired_mode` is set to `SLOPE_SLOWDOWN`
    - else, `desired_mode` is set to `NORMAL_SPEED`
  - `desired_mode` is published to /special_waypoint topic