Hersh Godse
Intel

Autoware

Why use ROS:

- Using ROS (middleware framework for robotic software development) because it simplifies complications from using various hardwares, OS, and programming languages
- ROS also provides several key tools:
    - Catkin build system
    - OpenCV image processing
    - ROSBAG for time-synchronized data logging and accessing
        - Data recorded and timestamped in .bag file
    - RViz to visualize data/software state
- Have several nodes that communicate with each other through message *topics*
    - Nodes can take input data, process it, and *publish* the results to other *receiver* nodes that in turn process their input
    - Allows for designing distribution systems
    - Each node run independently
    - ROS uses a .launch file to keep track of all the softare's nodes and parameters

Autoware Overview:

- Built on ROS
- Uses LIDAR (LIght Detection And Ranging) + on-vehicle cameras to localize a car's position
- Detects surrounding objects (pedestrians, traffic signals, vehicles, etc…) via LIDAR and GNSS (Global Navigation Satellite System) data
- Makes judgments on whether to drive/stop at lanes and intersections
- Provides 3D map generation, localization, object recognition, and vehicle control for autonomous driving

Autoware Mapping and Localization:

- Autoware has a high-definition 3D world map preloaded into the -computer that has information including roads, still objects around the roads, and locations of traffic signals and crosswalks. The world map is in PCD (Point Cloud Data) format
- Localization done by cross-matching this 3D world data to the 3D LIDAR data from the car's sensor, to determine the car's position
- This is done using the **NDT (Normal Distributions Tracking) algorithm** with world map and LIDAR
    - Algorithm determines best possible rigid transformation between two large point clouds, in this case car LIDAR data and the world map
- Error of localization: 10cm

Hersh Godse
Intel

Object Detection:

- **DPM (Deformable Part Models) algorithms** are used to detect cars/pedestrians/traffic lights, etc … from images
- Tracking using **Kalman filters** improves detection accuracy
- Incorporating 3D LIDAR data gives the distances to the detected objects
- Traffic Light Detection:

Path Generation:

- A route data generation app called AutowareRoute generates a path to the selected destination (path planning)
- The autonomous driving system then determines the lanes on the path, and what lanes to take
- The generated path includes appropriate speed info, used as a target speed by the autonomous driving system
- The rout also includes “waypoint” landmarks set at 1m intervals, and the autonomous driving system navigates to these successive waypoints (path following)

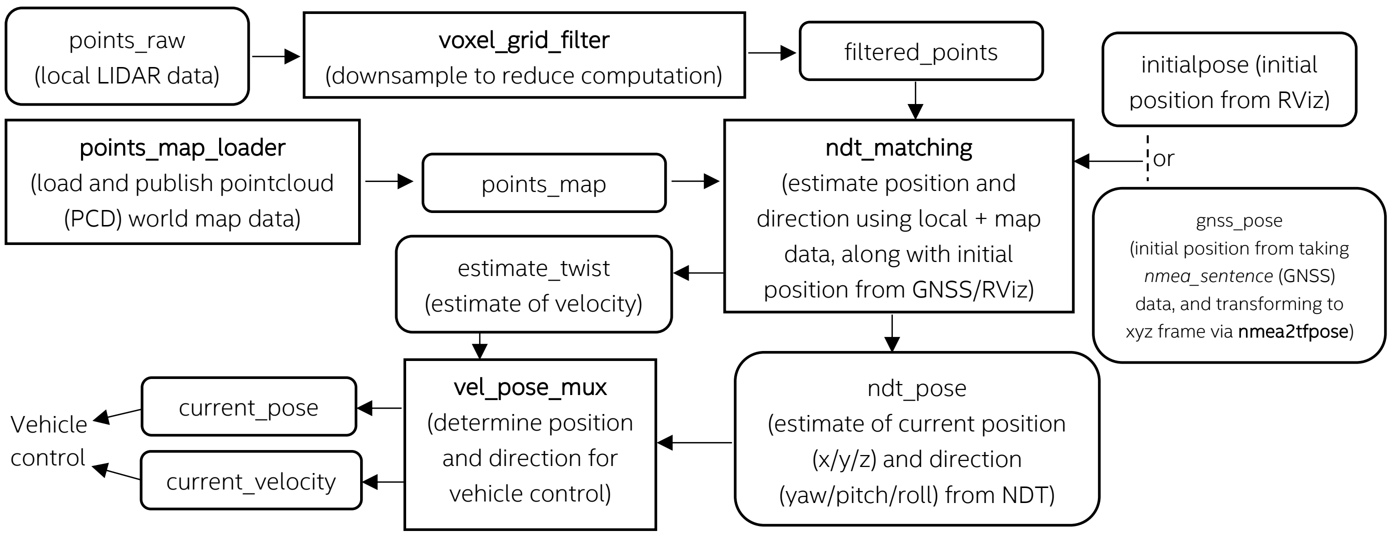
Hersh Godse
Intel

Sensor Calibration:

- /ros/src/sensing/fusion/packages/calibration_camera_lidar/
  - Calculates relative position of camera and LIDAR sensor on the car (x/y/z position, roll/pitch/yaw)
  - This information is used to project LIDAR sensor data onto camera images, and thus fuse the data

Localization:

- From LIDAR scan data, generate 3D map
- Localize vehicle position with **NDT (Normal Distributions Tracking) scan matching**
  - Matching local LIDAR scan data and 3D world map
  - Used to estimate position and moving direction of vehicle
  - Initial localization position obtained from GNSS data/known map coordinates/manual entry with RViz (2D Pose Estimate button in GUI)
- Workflow:


points_raw (local LIDAR data)
voxel_grid_filter (downsample to reduce computation)
filtered_points
initialpose (initial position from RViz)
or
points_map_loader (load and publish pointcloud (PCD) world map data)
points_map
ndt_matching (estimate position and direction using local + map data, along with initial position from GNSS/RViz)
gnss_pose (initial position from taking nmea_sentence (GNSS) data, and transforming to xyz frame via nmea2tfpose)
estimate_twist (estimate of velocity)
vel_pose_mux (determine position and direction for vehicle control)
ndt_pose (estimate of current position (x/y/z) and direction (yaw/pitch/roll) from NDT)
current_pose
current_velocity
Vehicle control

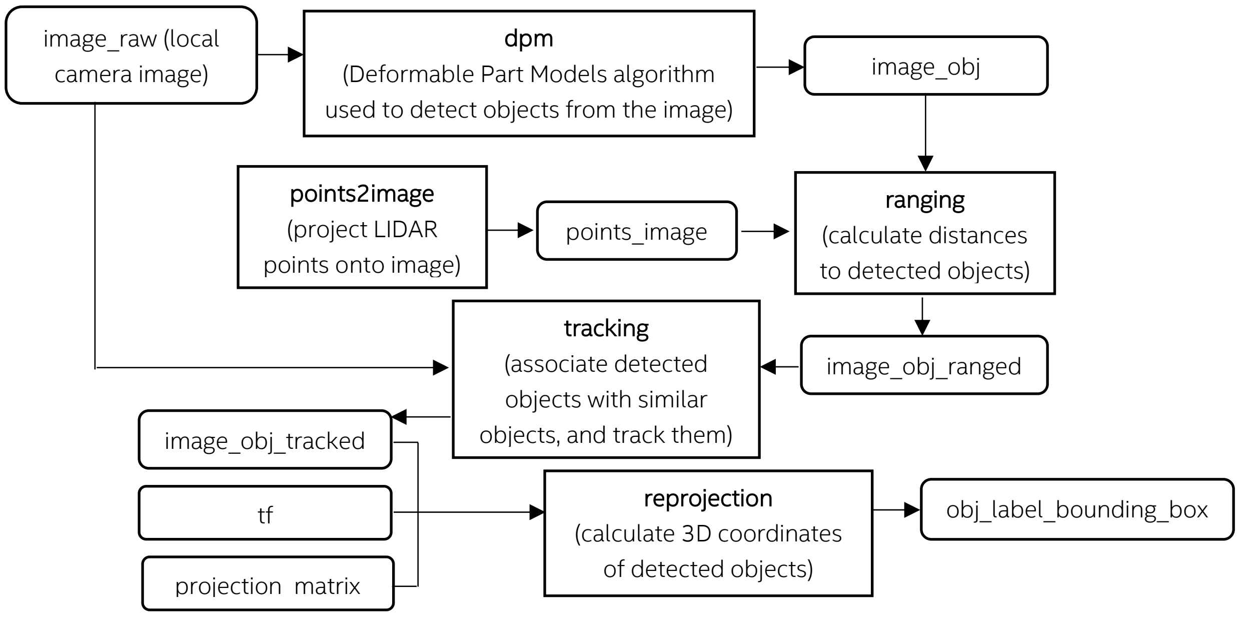
- /ros/src/computing/perception/localization/packages/ndt_localizer/nodes/ndt_matching
- /ros/src/computing/perception/localization/packages/gnss_localizer/nodes/nmea2tfpose
- /ros/src/data/packages/map_file/nodes/points_map_loader
- /ros/src/sensing/filters/packages/points_downsampler/nodes/voxel_grid_filter

- /ros/src/computing/perception/localization/packages/ndt_localizer
- /ros/src/computing/perception/localization/packages/icp_localizer
- /ros/src/computing/perception/localization/packages/orb_localizer
- /ros/src/computing/perception/localization/packages/autoware_connector
- /ros/src/computing/perception/localization/packages/gnss_localizer/nodes/fix2tfpose

Hersh Godse
Intel

Detection:

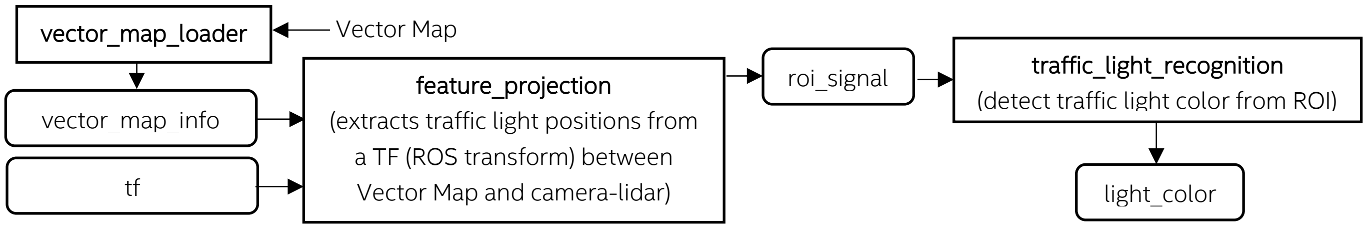
- Object detection:
    - Detect vehicles and pedestrians from camera images
    - Calculate distance to detected objects via LIDAR point information
    - Calculate 3D coordinates of objects via reprojection
    - Workflow:


image_raw (local camera image)
dpm
(Deformable Part Models algorithm used to detect objects from the image)
image_obj
points2image
(project LIDAR points onto image)
points_image
ranging
(calculate distances to detected objects)
tracking
(associate detected objects with similar objects, and track them)
image_obj_ranged
image_obj_tracked
tf
projection matrix
reprojection
(calculate 3D coordinates of detected objects)
obj_label_bounding_box

- /ros/src/computing/perception/detection/packages/cv_tracker/nodes/dpm_ocv
- /ros/src/computing/perception/detection/packages/cv_tracker/nodes/dpm_ttic
- /ros/src/computing/perception/detection/lib/image/dpm_ocv
- /ros/src/computing/perception/detection/lib/image/dpm_ttic
- /ros/src/sensing/fusion/packages/points2image
- /ros/src/computing/perception/detection/packages/cv_tracker/nodes/range_fusion
- /ros/src/computing/perception/detection/packages/lidar_tracker
- /ros/src/computing/perception/detection/packages/cv_tracker
- /ros/src/computing/perception/detection/lib/image/dpm_ttic/cpu/tracking.cpp
- /ros/src/computing/perception/detection/lib/image/dpm_ttic/gpu/tracking.cpp
- /ros/src/computing/perception/detection/lib/image/dpm_ttic/cpu/tracking.hpp
- /ros/src/computing/perception/detection/lib/image/dpm_ttic/gpu/tracking.hpp
- /ros/src/computing/perception/detection/packages/cv_tracker/nodes/obj_reproj

Hersh Godse
Intel

- Traffic Light Detection:
    - Detect traffic light color from camera images
    - Calculate traffic light coordinates from current position + VectorMap info from localization
        - Vector map is GIS (Global Information System) data representing geometric information of the roads
    - Traffic light position region of interest (ROI) is projected onto the camera image via sensor fusion
    - Image processing is used to determine the traffic light color in the ROI
    - In Autoware, can link light color detection result with path planning node to have starting/stopping at traffic light


vector_map_loader
Vector Map
vector_map_info
tf
feature_projection
(extracts traffic light positions from a TF (ROS transform) between Vector Map and camera-lidar)
roi_signal
traffic_light_recognition
(detect traffic light color from ROI)
light_color

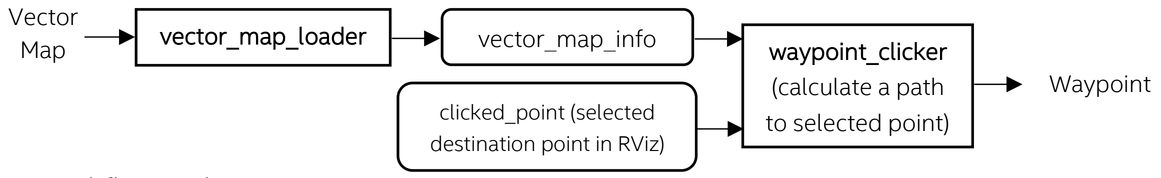
- /ros/src/computing/perception/detection/lib/libvectormap/
- /ros/src/computing/perception/detection/packages/road_wizard
- /ros/src/computing/perception/detection/packages/road_wizard/feat_proj
- /ros/src/computing/perception/detection/packages/road_wizard/region_tlr
- /ros/src/computing/perception/detection/packages/road_wizard/tlr_tuner
- /ros/src/computing/perception/detection/packages/integrated_viewer
- /ros/src/computing/perception/detection/packages/viewer

- Lane Detection (done with OpenCV):
    - Get an image frame from camera
    - Create an OpenCV image
    - Scale down image to reduce computation
    - Crop image – remove top part of image because only care about the bottom part where road is
    - Convert cropped image to grayscale
    - Perform a Gaussian blur + smooth edges: cvSmooth
    - Detect edges with Canny edge detection: cvCanny
    - Do Hough transform to find lanes: cvHoughLines2
    - Classify lanes to left/right side of vehicle, based on distance between line midpoint and horizontal center of image (assume vanishing point here)
        - reject poor lane candidates (reject if lane angle less than a threshold)
    - Publish to ROS topic *lane_pos_xy*
- /ros/src/computing/perception/detection/packages/lane_detector/nodes/lane_detector

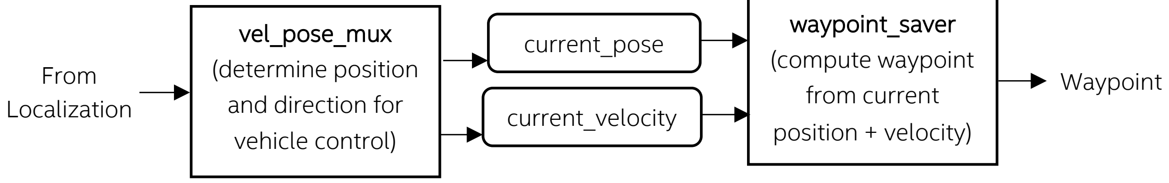
Hersh Godse
Intel

Path Generation:

- Generate path from Vector Map and position + velocity information (fom localization)
    - Vector map is GIS (Global Information System) data representing geometric information of the roads
    - If no vector map provided, log data used instead (see below)
- Path output is a data string of a "waypoint" containing coordinate, direction, and velocity info
- Workflow when using Vector Map:


Vector Map
vector_map_loader
vector_map_info
clicked_point (selected destination point in RViz)
waypoint_clicker (calculate a path to selected point)
Waypoint

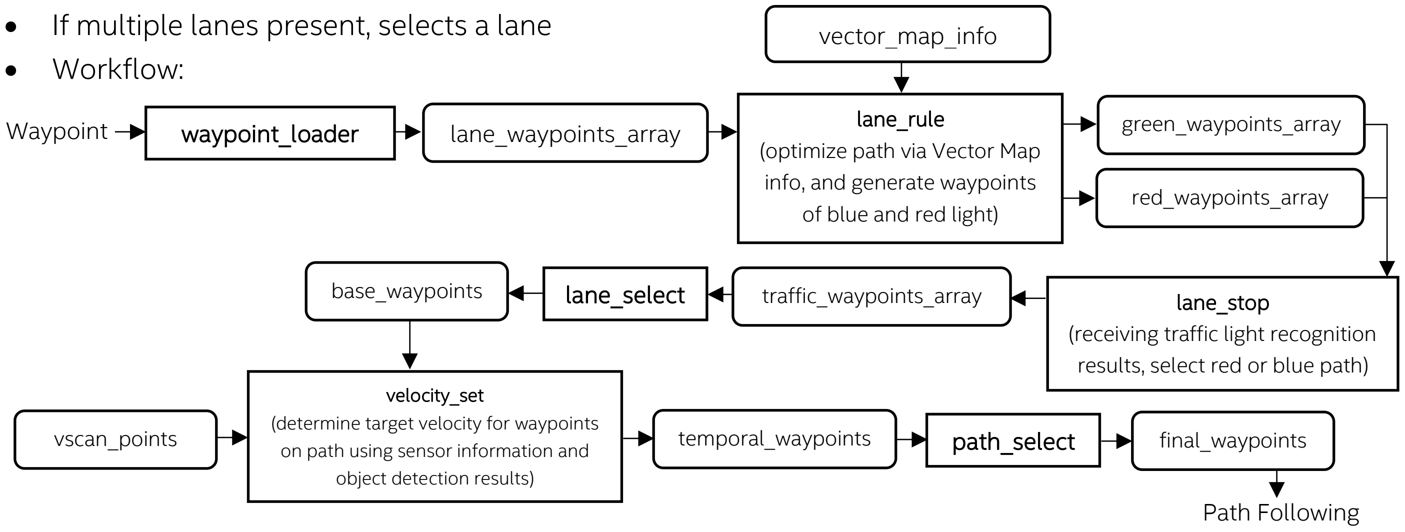
- Workflow without Vector Map:


From Localization
vel_pose_mux (determine position and direction for vehicle control)
current_pose
current_velocity
waypoint_saver (compute waypoint from current position + velocity)
Waypoint

- Creates a route to destination (set of sequential waypoints with x/y/z position, yaw, and target velocity information)
- /ros/src/computing/planning/motion/packages/waypoint_maker/nodes/waypoint_saver
- /ros/src/computing/planning/motion/packages/waypoint_maker/nodes/waypoint_clicker
- /ros/src/computing/planning/motion/packages/waypoint_maker/nodes/waypoint_marker_publisher

Path Planning:

- Happens after path generation (above), more global focus
- Load a previously saved path file
- Optimize velocities on the path using world map info from vector map
- If multiple lanes present, selects a lane
- Workflow:


Waypoint
waypoint_loader
lane_waypoints_array
vector_map_info
lane_rule (optimize path via Vector Map info, and generate waypoints of blue and red light)
green_waypoints_array
red_waypoints_array
lane_stop (receiving traffic light recognition results, select red or blue path)
traffic_waypoints_array
lane_select
base_waypoints
vscan_points
velocity_set (determine target velocity for waypoints on path using sensor information and object detection results)
temporal_waypoints
path_select
final_waypoints
Path Following

Hersh Godse
Intel

- /ros/src/computing/planning/motion/packages/waypoint_maker/nodes/waypoint_loader
    - For each path, load waypoint sequence (position, orientation, target velocity)
    - Calculate/plan out target velocities for waypoint sequences using physics deceleration formula
- /ros/src/computing/planning/mission/packages/lane_planner/nodes/lane_rule
    - From vector map info of world, gets number of lanes at a particular waypoint
    - Create 'stop points' at crosswalks/intersections where the car should be decelerating
    - Compute waypoint plans for both stopping and continuing at intersections. Use acceleration and deceleration functions to set/update target velocities before and after intersections/crosswalks (referred to as *'crossroads'* and *'stoplines'*). Apply functions when waypoints are within some radius of a stop point
        - Stop point proximity radius changes depending on if lane contains a curve/crossroad
    - Not sure, but possibly creating a different route for each of the lanes at a road
- /ros/src/computing/planning/mission/packages/lane_planner/nodes/lane_stop
    - Use info from traffic light detection topic to choose red (stop) vs green (continue) path
- /ros/src/computing/planning/mission/packages/lane_planner/nodes/lane_select
    - Receives multiple routes from the waypoint_maker package (Path Generation)
    - For all routes, calculate the nearest neighbor point relative to current position
    - From currently running route, use current location to compute current lane, then classify the other lines as to the left or right, and calculate the closest left lane and closest right lane (it's possible for these to not exist)
    - The current route's nearest neighbor will have a lane change flag (straight/left/right). This becomes the current lane change flag of the route
    - If route's lane change flag indicates a lane change, generate a **Hermitian** interpolated curve route. For this curve, target point is lane to be changed into (left/right lane).
    - Create a new lane change route that consists of this Hermite curve to make the lane change, joined together with the route of the changed lane after the target point (the route from the target point on left/right lane and beyond)
    - When we DO make a lane change, publish this lane change route, it's nearest neighbor point, and it's lane change flag
    - When we DON'T make a lane change, publish the current route, it's nearest neighbor point, and it's lane change flag
    - https://github.com/CPFL/Autoware/blob/fad2d17234e77e393e263aac861b7a835f10903a/ros/src/computing/planning/mission/packages/lane_planner/README.md (Right click, translate to English)

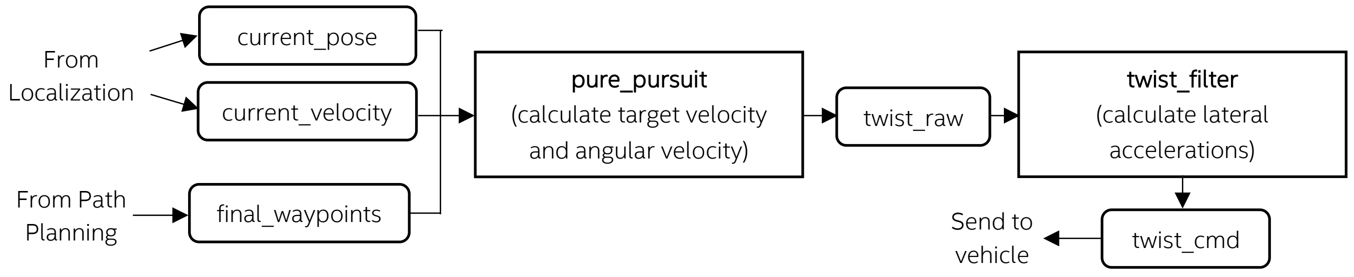
Hersh Godse
Intel

- /ros/src/computing/planning/mission/packages/lattice_planner/nodes/path_select
- /ros/src/computing/planning/mission/packages/lattice_planner/nodes/lattice_velocity_set
- /ros/src/computing/planning/mission/packages/astar_planner/nodes/velocity_set

- /ros/src/computing/planning/mission/packages/lane_planner/lane_navi
- /ros/src/computing/planning/mission/packages/freespace_planner/astar_navi
- /ros/src/computing/planning/motion/packages/astar_planner/obstacle_avoid
- /ros/src/computing/planning/motion/packages/lattice_planner/lattice_trajectory_gen
- /ros/src/computing/planning/motion/packages/lattice_planner/lattice_twist_convert

Path Following:

- Happens after path planning (above), more local focus
- Calculate vehicle control signals for vehicle to follow (velocity, angular velocity)
- Two nodes: *pure_pursuit* and *twist_filter*
- Pure_pursuit:
    - Calculate curve of circle passing through current position and path target point
    - Compute target angular velocities from calculated curvature and current velocity
- Twist_filter:
    - Calculate lateral accelerations from target velocity/angular velocity
    - If lateral acceleration is over a threshold value, slow down target velocities
- These computations are transmitted to the vehicle for control
- Workflow:



- /ros/src/computing/planning/motion/packages/waypoint_follower/pure_pursuit
- /ros/src/computing/planning/motion/packages/waypoint_follower/twist_filter
- /ros/src/computing/planning/motion/packages/waypoint_follower/wf_simulator
    - Simulates ideal vehicle position and velocity (in closed loop feedback)

Note:

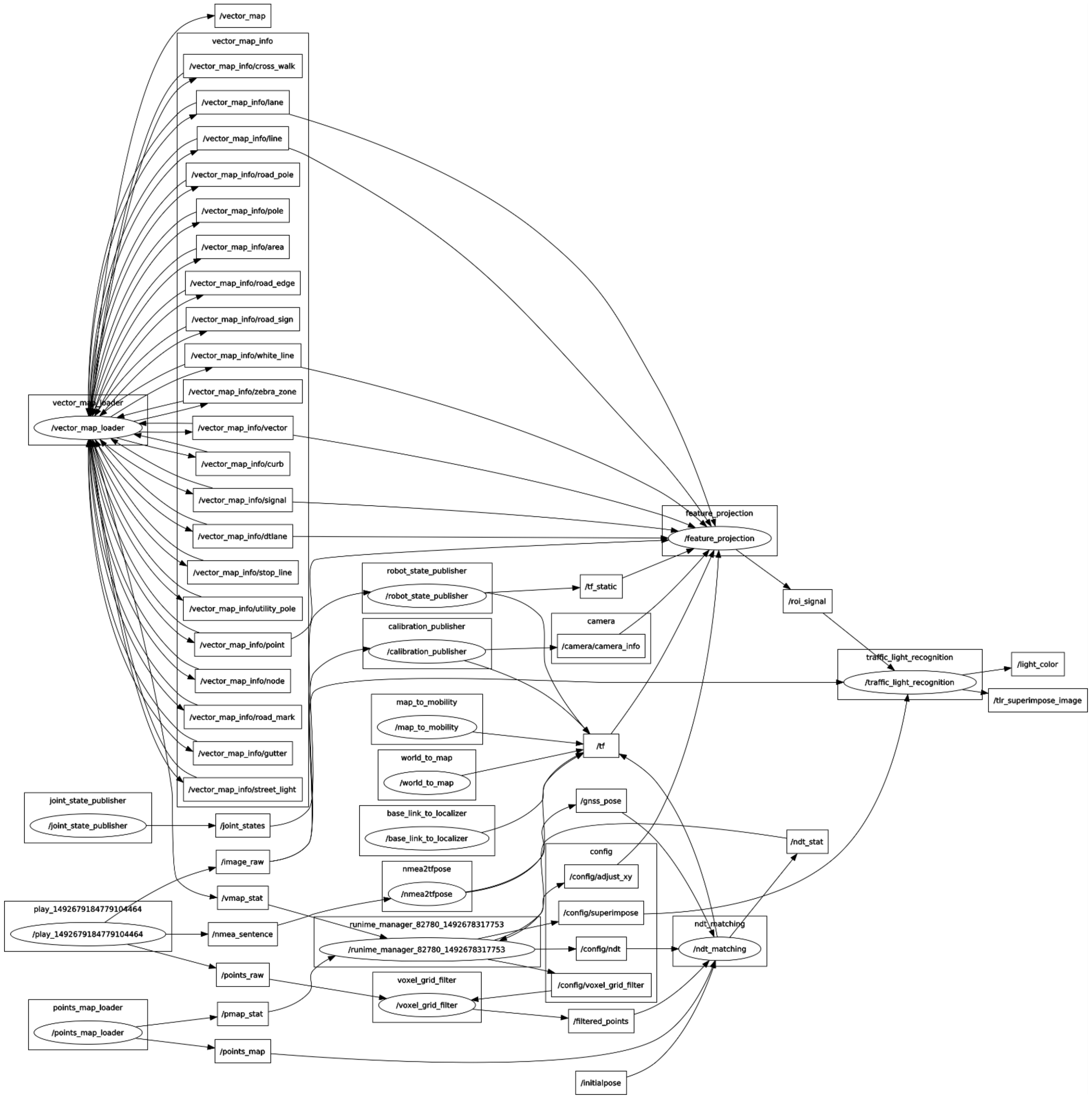
- The package **vel_pose_mux** has been replaced by the **autoware_connector** package

Other Files in Autoware src:

- /ros/src/.config/
    - Contains simulation data, configuration files, and quick_start demo setup
- OpenPlanner:
    - /ros/src/computing/planning/motion/packages/op_simulator
    - /ros/src/computing/planning/motion/packages/op_simulator_perception
    - /ros/src/computing/planning/motion/packages/ff_waypoint_follower
    - /ros/src/computing/planning/common/lib/openplanner
- /ros/src/computing/planning/state/state_machine
- /ros/src/computing/perception/semantics/packages/object_map
- /ros/src/sensing/fusion/packages
- /ros/src/sensing/filters/packages
- /ros/src/sensing/polygon/packages
- /ros/src/sensing/drivers
- /ros/src/data/packages
- /ros/src/socket/packages
- /ros/src/system/
- /ros/src/util/

Hersh Godse
Intel

Diagram of all Autoware ROS Topics and Overall Workflow:


/vector_map
vector_map_info
/vector_map_info/cross_walk
/vector_map_info/lane
/vector_map_info/line
/vector_map_info/road_pole
/vector_map_info/pole
/vector_map_info/area
/vector_map_info/road_edge
/vector_map_info/road_sign
/vector_map_info/white_line
/vector_map_info/zebra_zone
/vector_map_info/vector
/vector_map_info/curb
/vector_map_info/signal
/vector_map_info/dtlane
/vector_map_info/stop_line
/vector_map_info/utility_pole
/vector_map_info/point
/vector_map_info/node
/vector_map_info/road_mark
/vector_map_info/gutter
/vector_map_info/street_light
vector_map_loader
/vector_map_loader
feature_projection
/feature_projection
robot_state_publisher
/robot_state_publisher
/tf_static
/roi_signal
camera
/camera/camera_info
calibration_publisher
/calibration_publisher
traffic_light_recognition
/traffic_light_recognition
/light_color
/tlr_superimpose_image
map_to_mobility
/map_to_mobility
/tf
world_to_map
/world_to_map
joint_state_publisher
/joint_state_publisher
/joint_states
base_link_to_localizer
/base_link_to_localizer
/gnss_pose
/ndt_stat
/image_raw
config
/config/adjust_xy
nmea2tfpose
/nmea2tfpose
/vmap_stat
/config/superimpose
play_1492679184779104464
/play_1492679184779104464
/nmea_sentence
runime_manager_82780_1492678317753
/runime_manager_82780_1492678317753
/config/ndt
ndt_matching
/ndt_matching
/points_raw
voxel_grid_filter
/voxel_grid_filter
/config/voxel_grid_filter
points_map_loader
/points_map_loader
/pmap_stat
/filtered_points
/points_map
/initialpose