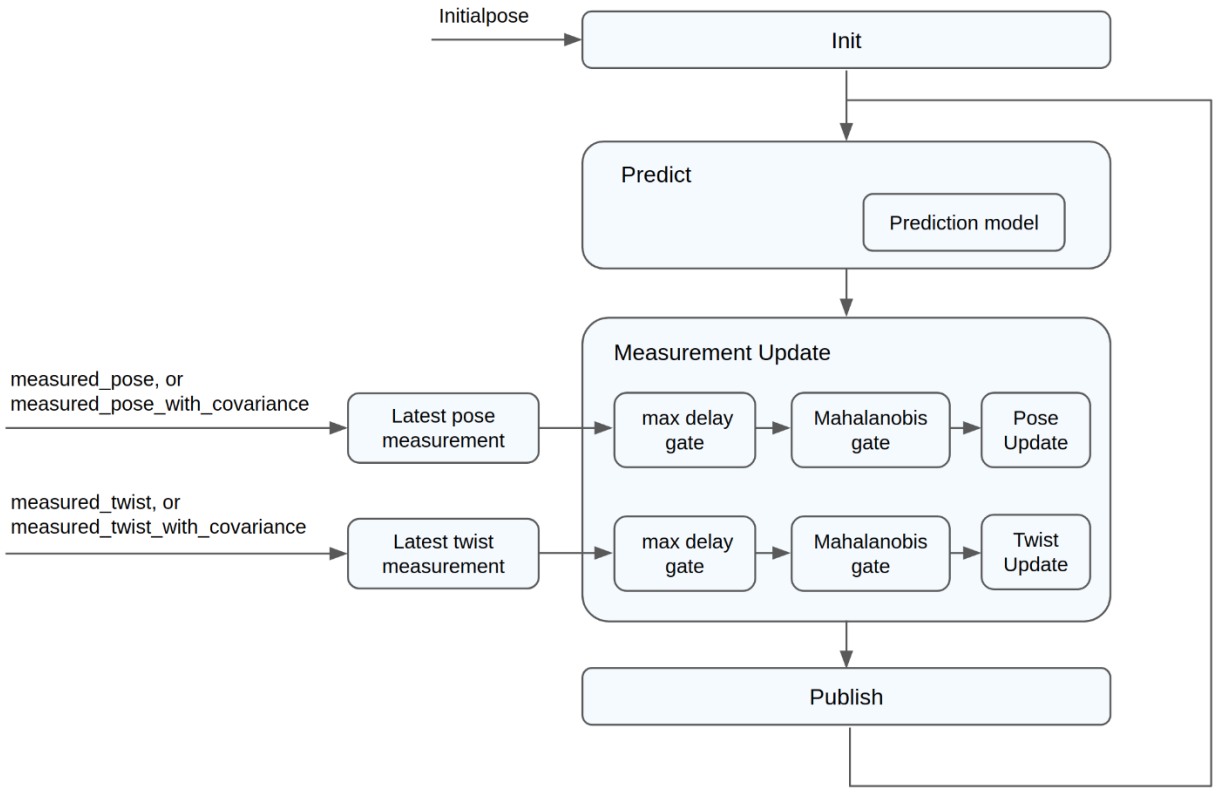
Autoware Hands-on

Autoware ekf_localizer



https://gitlab.com/autowarefoundation/autoware.ai/core_perception/tree/master/ekf_localizer

Input:
/devbot/twist … twist from Devbot (velocity, yaw_rate)
/ndt_pose … position from localization (lidar or noisy GPS data)

Output:
/ekf_pose_with_covariance … output from the EKF for localization

Ground truth: /gps_local/pose

Start localization

Terminal 1:
roscore -p $ROS_PORT

Terminal 2:
rosparam set use_sim_time true
rosbag play ~/aa274_autoware_ws/src/aa274_data/devbot_lap0.bag --clock /tf:=/tf_old /ndt_pose:=/ndt_pose_old

Terminal 3:
source ~/autoware.sh
roslaunch vifware_launch Devbot_localization.launch

Operations for localization evaluation:

1) GPS based localization with noisy gps data:
lidar_localization_active: false

localization_pose: /ndt_pose (gps_pose + noise)

2) Lidar based localization (localization running online)
lidar_localization_active: true

localization_pose: /ndt_pose (ndt_localization)

Definition of the mode in: **vifware_launch/launch/localization_devbot/Devbot_localization.launch**

The EKF localizer can be defined in: **vifware_launch/launch/localization_devbot/ekf_localizer.launch**

After every change in a launch file you need to rebuild the source!

Tasks

1) Localization only with Odometry (invalid input_pose_name)

2) Localization with GPS without noise
vifware_launch/launch/localization_devbot/gps_to_ndt_pose.launch
stddev_x_y: 0
mu_x_y: 0

3) Localization with GPS with noise
vifware_launch/launch/localization_devbot/gps_to_ndt_pose.launch
stddev_x_y: 1
mu_x_y: 0

4) Localization with GPS with noise incl. bias
vifware_launch/launch/localization_devbot/gps_to_ndt_pose.launch
stddev_x_y: 1
mu_x_y: 1

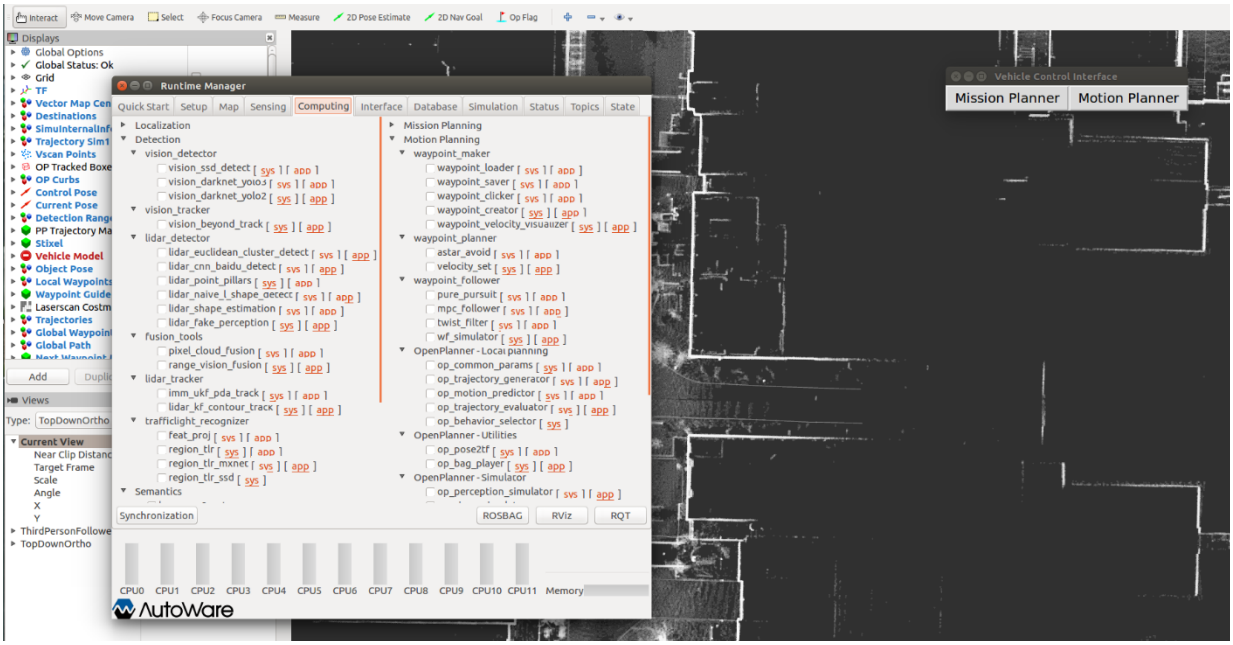
5) Localization with lidar
→parameter tuning (lidar pose has an unknown time delay and unknown noise)

Goal: the ekf_pose should match the gps_local/pose

Autoware simulator / Path planning

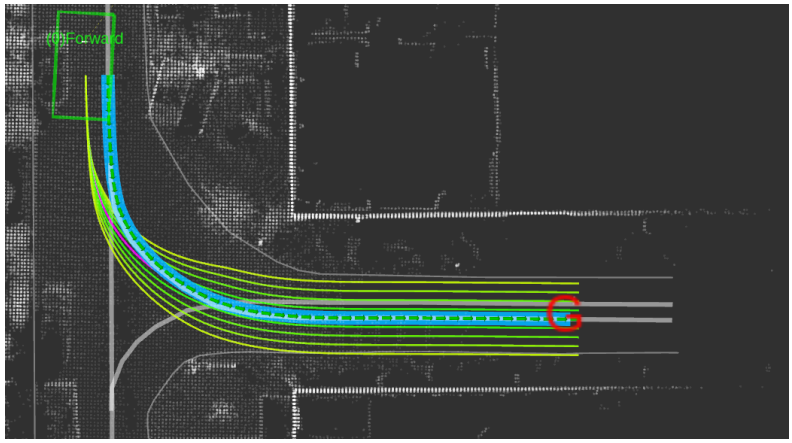
Terminal 1:
roscore -p $ROS_PORT

Terminal 2:
source ~/autoware.sh
roslaunch vifware_launch simulation.launch

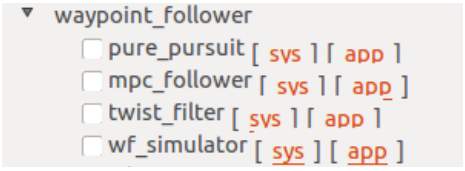


1) Start in the “Vehicle Control Interface” the “Mission Planner” and set a start/stop position of the ego vehicle in RVIZ with 2D Pose Estimate/2D Nav Goal.

2) Start the “Motion Planner”



3) Start the waypoint follower to start the vehicle movement with the Autoware GUI under the page “Computing”. You can choose between the Pure Pursuit and the MPC Follower trajectory tracker.



→ The vehicle starts driving!