Design of autonomous based software for Robotiiva rover

By: Mohamed Majdi Abdelsattar

Software overview

ROS is a software framework used for creating robotic applications. The main aim of the ROS framework is to provide the capabilities that you can use to create powerful robotics applications that can be reused for other robots. ROS has a collection of software tools, libraries, and a collection of packages that makes robot software development easy. The source code of ROS is open and it's completely free to use. The core part of ROS is licensed under a BSD license, and it can be reused in commercial and closed source products. ROS is a combination of plumbing (message passing), tools, capabilities, and ecosystem. There are powerful tools in ROS to debug and visualize the robot data. There are inbuilt robot capabilities in ROS, such as robot navigation, localization, mapping, manipulation, and so on. They help to create powerful robotics applications. Gazebo is a free and open-source robot simulator in which we can test our own algorithms, design robots, and test robots in different simulated environments. Gazebo can accurately and efficiently simulate complex robots in indoor and outdoor environments. Gazebo is built with a physics engine with which we can create high-quality graphics and rendering. Gazebo supports a wide range of sensors, including laser range finders, Kinect-style sensors, 2D/3D cameras, and so on. Gazebo provides models for popular robots, such as PR2, Pioneer 2 DX, iRobot Create, and TurtleBot. We can also build custom models of robots. We can run simulations on the cloud server using the CloudSim framework. Rviz (ROS Visualizer) is a GUI tool in ROS that is used to visualize different kinds of robot sensor data from robot hardware or a simulator, such as Gazebo.

System Overview

Robotiiva rover is a multifunctional robot with the ability to navigate outdoors in hard and unknown terrains with the help of its sensors' setup; GPS, IMU, and Kinect camera. The design of the autonomous software is done using the ROS and is simulated using Gazebo simulator. The main aim of deploying vision sensors in our robot is to detect objects and navigate the robot through an environment. SLAM is an algorithm that is used in mobile robots to build up a map of an unknown environment or update a map within a known environment by tracking the current location of the robot. Maps are used to plan the robot's trajectory and to navigate through this path. Using maps, the robot will get an idea about the environment. The two main challenges in mobile robot navigation are mapping and localization. Mapping involves generating a profile of obstacles around the robot. Through mapping, the robot will understand what the world looks like. Localization is the process of estimating the position of the robot relative to the map we build. The processes of mapping, localization, and navigation are done using specific ROS packages. Our system involves two main modules; one for indoors navigation and the other for outdoors navigation. The system that will be discussed here is the first version of our work, and there will be more clarification about the structure of the next versions in the future work section. In this version, indoors autonomous navigation is done using two main ROS packages; Gmapping and move_base. Gmapping provides laser-based SLAM (Simultaneous Localization and Mapping) to build the map and localize the robot at the same time. Once there is a map and the robot is localized on that map, move_base package can now do the path planning work and autonomously navigate the robot to the goal location. The outdoors navigation is different, we decided not to use SLAM approaches and do the navigation with an empty map. The method relies on localizing the robot in an empty map without obstacles using GPS and IMU mainly by the robot_localization package then using the move_base to do the rest of the work of navigation.

ROS Implementation Notes

Here we will state some important notes about some configurations inside the code in the files of the packages used and the whole code is shown in the appendix.

First, we will state the common files between the two modules, then we will show the specific files of each module separately.

Common files

(with_map_move_base.launch): A file for the Navigation Stack which will perform the path planning and navigation and it takes in configuration files of (.yaml) format that define everything from which planners are going to be used, to how maps and sensors are used.

(**view_robot.launch**): file inserted in other launch files to open rviz with pre-saved configurations immediately when launching the file.

(.yaml) files for the move_base:

(**costmap_common_params.yaml**): the obstacle_layer parameter is set to use the topic (/front/scan) to read the laser-type data for navigation. The laser range is set for 3 meters only for objects detection in raytrace_range parameter.

(**global_costmap_params.yaml**): the global_frame parameter is set to map to calculate all paths based on the map frame, and the added plugins generate costmaps.

(**local_costmap_params.yaml**): the global_frame parameter is set to map.

(**base_local_planner_params.yaml**): holonomic_robot is set to non-holonomic and yaw_goal_tolerance, xy_goal_tolerance parameters are relatively low.

(**move_base_params.yaml**): clearing_rotation_allowed parameter is set to true to make the robot turn around to clear space if for some reason got lost to recover itself and is helpful in situations when a moving obstacle was detected in the path because when turning, this gives time to the obstacle to move out the way and thus clear the local cost map.

For localplanner: TrajectoryPlannerRos is used.

For globalplanner: navfn/NavfnRos is used.

Indoors Navigation module

(**gmapping.launch**): here all parameters related to map generation are defined. To improve performance, maxUrange parameters is set to 5.0m and maxRange parameter is set to 10m.

(**start_mapping.launch**): this file is launched to create a map, then rviz is launched to see the map creation process, then saving it as two files using the command (rosrun map_server map_saver -f name_of_the_map).

(**amcl.launch**): include the localization parameters and does the transform from the map frame to the odometry frame.

(**start_navigation_with_map.launch**): this file is launched after the map creation to start autonomous navigation and it launches the map server, amcl, and move_base nodes.

Outdoors Navigation module

The main difference in the two modules is that the normal indoors localization is done using existing built map and the outdoors localization is done without map (empty map) and by using GPS.

In this configuration, we will use the robot_localization package in the localization step instead of Adaptive Monto Carlo Localization. Using the robot_localization package, our sensors represented in GPS, IMU, and Odometry sensors -which computes how much a robot is moved from its previous location- , would be fused to perform the transformation from the map frame to the odometry frame to make the work of the move_base package possible and hence perform the autonomous navigation.

(**start_map_server.launch**): this file is used to load an empty map to be used as a reference to be able to use rosnodes that use maps for navigating.

(**start_navsat.launch**): this file launches the robot_localization package with Navsat transfer node enabled to be able to work with GPS data. The node converts the GPS coordinate data to UTM coordinate system, then publish the transform from UTM to the map. The changes in this file include remapping topics of the sensors to work with different systems properly.

(**start_navigation_with_gps_ekf.launch**): this file launches all the necessary launch files for different packages to start autonomous navigation.

(**robot_localization_with_gps.yaml**): this file is used to configure the localization node to work specifically with our model. Changes include indicating the odometry topics.

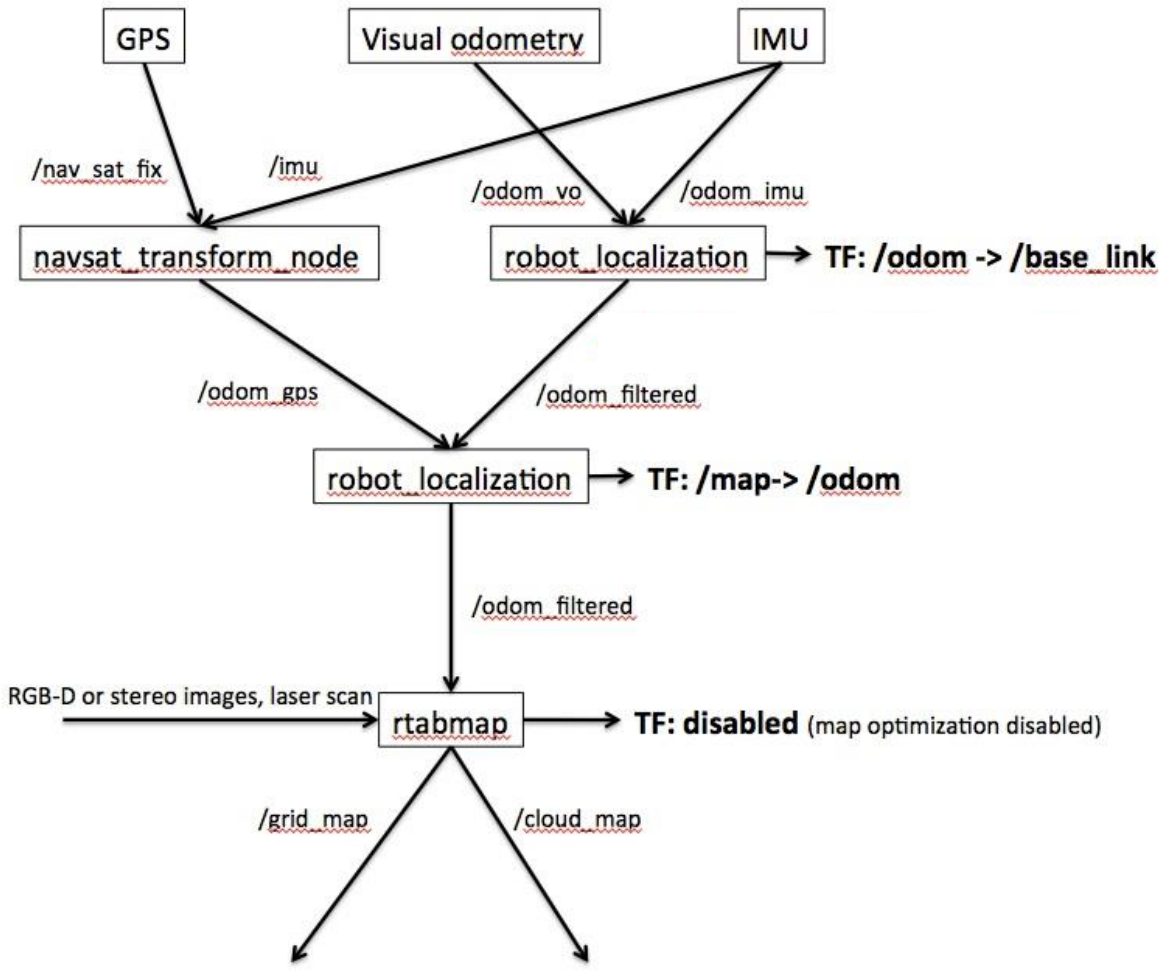
Note that the outdoors navigation module can be used in the indoors environments and vise verse, but the performance would decrease in such cases for sure.

For some reasons, we will use a Kinect camera instead of a laser sensor, and we will convert the depth data to laser scan data to be able to run our system-laser-based properly. The depthimage_to_laserscan ROS package would be used to do so.

Future work

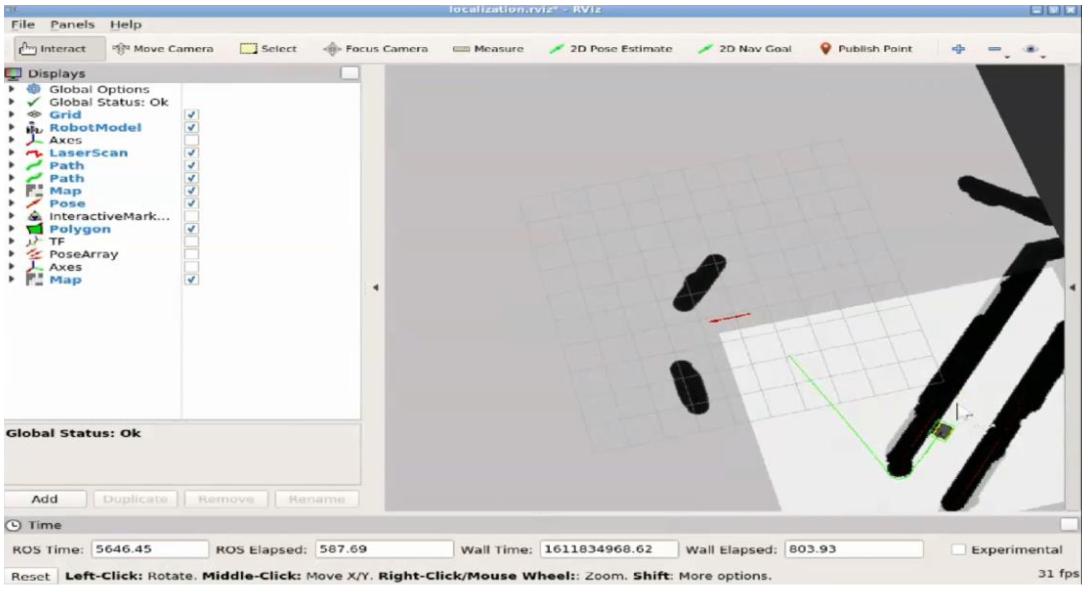
The next version of our system will be robust and for all cases. The main changes would be in adding an RGB-D SLAM approach by using the RTAB-MAP ROS package to the autonomous system, then integrating it with the robot_localization package and the navigation stack.

This picture shows an example of possible but not final future integrations between the existing system and the R-TAB MAP ROS package:



Appendix

This picture shows the actual software in action and the RVIZ configurations:



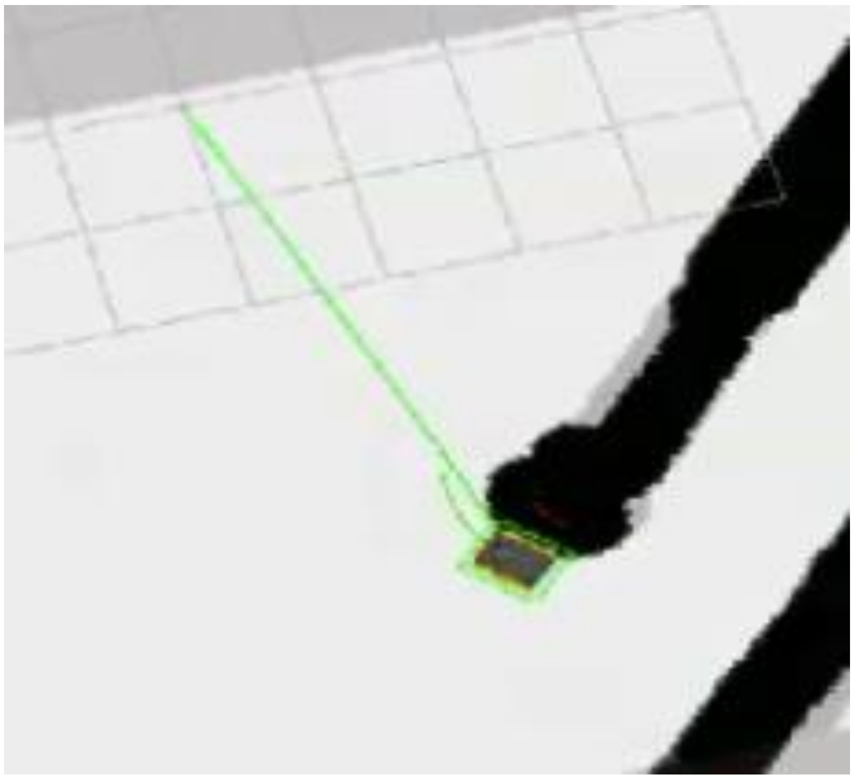
Global Options
Global Status: Ok
Grid ✓
RobotModel ✓
Axes
LaserScan ✓
Path ✓
Path ✓
Map ✓
Pose ✓
InteractiveMark...
Polygon ✓
TF
PoseArray
Axes
Map ✓

Global Status: Ok

ROS Time: 5646.45 ROS Elapsed: 587.69 Wall Time: 1611834968.62 Wall Elapsed: 803.93 Experimental

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options. 31 fps

This picture shows the global and cost map :



In this part, the whole code is illustrated:

(With_map_move_base.launch):

```
<launch>

  <node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">

    <rosparam file="$(find my_jackal_tools)/params/costmap_common_params.yaml" command="load" ns="global_costmap" />

    <rosparam file="$(find my_jackal_tools)/params/costmap_common_params.yaml" command="load" ns="local_costmap" />

    <rosparam file="$(find my_jackal_tools)/params/map_nav_params/local_costmap_params.yaml" command="load" />

    <rosparam file="$(find my_jackal_tools)/params/map_nav_params/global_costmap_params.yaml" command="load" />

    <rosparam file="$(find my_jackal_tools)/params/base_local_planner_params.yaml" command="load" />

    <rosparam file="$(find my_jackal_tools)/params/move_base_params.yaml" command="load" />

    <param name="base_global_planner" type="string" value="navfn/NavfnROS" />

    <param name="base_local_planner" value="base_local_planner/TrajectoryPlannerROS"/>

    <remap from="odom" to="odometry/filtered" />

  </node>

</launch>
```

(view_robot.launch):

```
<launch>

  <param name="use_gui" value="true"/>

  <arg name="config" default="complete" />

  <node name="rviz" pkg="rviz" type="rviz" args="-d $(find my_jackal_tools)/rviz/$(arg config).rviz" />

</launch>
```

(start_mapping.launch):

```
<launch>

  <!--- Run gmapping -->

  <include file="$(find my_jackal_tools)/launch/gmapping.launch" />

  <!--- Run Move Base -->

  <include file="$(find my_jackal_tools)/launch/with_map_move_base.launch" />

</launch>
```

(start_map_server.launch):

```
<launch>

    <node name="map_server" pkg="map_server" args="$(find jackal_tools/maps/mymap_empty.yaml)"/>

</launch>
```

(start_navsat.launch):

```
<launch>

 <!-- -->

  <node pkg="robot_localization" type="navsat_transform_node" name="navsat_transform_node" respawn="true">

    <param name="magnetic_declination_radians" value="0"/>

    <param name="yaw_offset" value="0"/>

    <param name="zero_altitude" value="true"/>

    <param name="broadcast_utm_transform" value="false"/>

    <param name="publish_filtered_gps" value="false"/>

    <param name="use_odometry_yaw" value="false"/>

    <param name="wait_for_datum" value="false"/>

    <remap from="/imu/data" to="/imu/data" />

    <remap from="/gps/fix" to="/navsat/fix" />

    <remap from="/odometry/filtered" to="/odom" />

  </node>

</launch>
```

(start_navigation_with_gps_ekf.launch):

```
<launch>

    <!--run navsat gps to odometry conversion-->

    <include file="$(find my_jackal_tools)/launch/start_navsat.launch" />

    <!--run the ekf for map to odom config -->

    <node pkg="robot_localization" type="ekf_localization_node" name="ekf_localization_with_gps">

    <rosparam command="load" file="$(find my_jackal_tools)/params/robot_localization_with_gps.yaml" />

    </node>

    <!-- Run the map server -->

   <arg name="map_file" default="$(find my_jackal_tools)/maps/mymap_empty.yaml" />

   <node name="map_server" pkg="map_server" type="map_server" args="$(arg map_file)" />

    <!--- Run Move Base -->

    <include file="$(find my_jackal_tools)/launch/with_map_move_base.launch" />

    <!-- run RVIZ for loclization-->

    <include file="$(find my_jackal_tools)/launch/view_robot.launch" />

        <arg name="config" value="localization" />

</launch>
```

(start_navigation_with_map.launch):

```
<launch>

    <!-- Run the map server -->

    <arg name="map_file" default="$(find jackal_tools)/maps/mymap.yaml"/>

    <node name="map_server" pkg="map_server" type="map_server" args="$(arg map_file)" />

    <!--- Run AMCL -->

    <include file="$(find jackal_tools)/launch/amcl.launch" />

    <!--- Run Move Base -->

    <include file="$(find jackal_tools)/launch/with_map_move_base.launch" />

</launch>
```

(gmapping.launch):

```
<launch>

  <arg name="scan_topic" default="front/scan" />

  <node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output="screen">

    <param name="odom_frame" value="odom"/>

    <param name="base_frame" value="base_link"/>

    <param name="map_frame" value="map"/>

    <!-- Process 1 out of every this many scans (set it to a higher number to skip more scans)  -->

    <param name="throttle_scans" value="1"/>

    <param name="map_update_interval" value="5.0"/> <!-- default: 5.0 -->

    <!-- The maximum usable range of the laser. A beam is cropped to this value.  -->

    <param name="maxUrange" value="5.0"/>

    <!-- The maximum range of the sensor. If regions with no obstacles within the range of the sensor should appear as free space in the map, set maxUrange < maximum range of the real sensor <= maxRange -->

    <param name="maxRange" value="10.0"/>

    <param name="sigma" value="0.05"/>

    <param name="kernelSize" value="1"/>

    <param name="lstep" value="0.05"/>

    <param name="astep" value="0.05"/>

    <param name="iterations" value="5"/>

    <param name="lsigma" value="0.075"/>

    <param name="ogain" value="3.0"/>

    <param name="minimumScore" value="0.0"/>

    <!-- Number of beams to skip in each scan. -->

    <param name="lskip" value="0"/>

    <param name="srr" value="0.01"/>

    <param name="srt" value="0.02"/>

    <param name="str" value="0.01"/>
    <param name="stt" value="0.02"/>

    <!-- Process a scan each time the robot translates this far  -->

    <param name="linearUpdate" value="0.1"/>

    <!-- Process a scan each time the robot rotates this far  -->

    <param name="angularUpdate" value="0.05"/>

    <param name="temporalUpdate" value="-1.0"/>

    <param name="resampleThreshold" value="0.5"/>

    <!-- Number of particles in the filter. default 30        -->

    <param name="particles" value="10"/>

<!-- Initial map size  -->

    <param name="xmin" value="-10.0"/>

    <param name="ymin" value="-10.0"/>

    <param name="xmax" value="10.0"/>

    <param name="ymax" value="10.0"/>

    <!-- Processing parameters (resolution of the map)  -->

    <param name="delta" value="0.02"/>

    <param name="llsamplerange" value="0.01"/>

    <param name="llsamplestep" value="0.01"/>

    <param name="lasamplerange" value="0.005"/>

    <param name="lasamplestep" value="0.005"/>

    <remap from="scan" to="$(arg scan_topic)"/>

  </node>

</launch>
```

(amcl.launch):

```
<launch>

  <arg name="use_map_topic" default="false"/>

  <arg name="scan_topic" default="front/scan" />

  <node pkg="amcl" type="amcl" name="amcl">

    <param name="use_map_topic" value="$(arg use_map_topic)"/>

    <!-- Publish scans from best pose at a max of 10 Hz -->

    <param name="odom_model_type" value="diff"/>

    <param name="odom_alpha5" value="0.1"/>

    <param name="gui_publish_rate" value="10.0"/>

    <param name="laser_max_beams" value="720"/>

    <param name="laser_min_range" value="0.1"/>

    <param name="laser_max_range" value="30.0"/>

    <param name="min_particles" value="500"/>

    <param name="max_particles" value="2000"/>
```

```
<!-- Maximum error between the true distribution and the estimated distribution. -->

<param name="kld_err" value="0.05"/>

<param name="kld_z" value="0.99"/>

<param name="odom_alpha1" value="0.2"/>

<param name="odom_alpha2" value="0.2"/>

<!-- translation std dev, m -->

<param name="odom_alpha3" value="0.2"/>

<param name="odom_alpha4" value="0.2"/>

<param name="laser_z_hit" value="0.5"/>

<param name="laser_z_short" value="0.05"/>

<param name="laser_z_max" value="0.05"/>

<param name="laser_z_rand" value="0.5"/>

<param name="laser_sigma_hit" value="0.2"/>

<param name="laser_lambda_short" value="0.1"/>

<param name="laser_model_type" value="likelihood_field"/>

<!-- Maximum distance to do obstacle inflation on map, for use in likelihood_field model. -->

<param name="laser_likelihood_max_dist" value="2.0"/>

<!-- Translational movement required before performing a filter update.  -->

<param name="update_min_d" value="0.1"/>

<!--Rotational movement required before performing a filter update. -->

<param name="update_min_a" value="0.314"/>

<param name="odom_frame_id" value="odom"/>

<param name="base_frame_id" value="base_link"/>

<param name="global_frame_id" value="map"/>

<!-- Number of filter updates required before resampling. -->

<param name="resample_interval" value="1"/>

<!-- Increase tolerance because the computer can get quite busy -->

<param name="transform_tolerance" value="1.0"/>

<!-- Exponential decay rate for the slow average weight filter, used in deciding when to recover by adding random poses. A good value might be 0.001. -->

<param name="recovery_alpha_slow" value="0.0"/>

<!--Exponential decay rate for the fast average weight filter, used in deciding when to recover by adding random poses. A good value might be 0.1. -->

<param name="recovery_alpha_fast" value="0.1"/>

<!-- Initial pose mean -->

<param name="initial_pose_x" value="0.0" />

<param name="initial_pose_y" value="0.0" />

<param name="initial_pose_a" value="0.0" />

<!-- When set to true, AMCL will subscribe to the map topic rather than making a service call to receive its map.-->

<param name="receive_map_topic" value="true"/>

<!--  When set to true, AMCL will only use the first map it subscribes to, rather than updating each time a new one is received. -->
```

```
    <param name="first_map_only" value="false"/>
    <remap from="scan" to="$(arg scan_topic)"/>
  </node>
</launch>
```

(global_costmap_params.yaml):

```
global_costmap:
   global_frame: map
   robot_base_frame: base_link
   update_frequency: 20.0
   publish_frequency: 5.0
   width: 40.0
   height: 40.0
   resolution: 0.05
   origin_x: -20.0
   origin_y: -20.0
   static_map: true
   rolling_window: false
   plugins:
   - {name: static_layer, type: "costmap_2d::StaticLayer"}
   - {name: obstacles_layer, type: "costmap_2d::ObstacleLayer"}
   - {name: inflater_layer, type: "costmap_2d::InflationLayer"}
```

(local_costmap_params.yaml):

```
local_costmap:
   global_frame: map
   robot_base_frame: base_link
   update_frequency: 20.0
   publish_frequency: 5.0
   width: 10.0
   height: 10.0
   resolution: 0.05
   static_map: false
   rolling_window: true
```

(move_base_params.yaml):

```
shutdown_costmaps: false

controller_frequency: 20.0

controller_patience: 15.0

planner_frequency: 20.0

planner_patience: 5.0

oscillation_timeout: 0.0

oscillation_distance: 0.5

recovery_behavior_enabled: true

clearing_rotation_allowed: true
```

(base_local_planner_params.yaml):

```
TrajectoryPlannerROS:

  # Robot Configuration Parameters

  acc_lim_x: 10.0

  acc_lim_theta:  20.0

  max_vel_x: 0.5

  min_vel_x: 0.1

  max_vel_theta: 1.57

  min_vel_theta: -1.57

  min_in_place_vel_theta: 0.314

  holonomic_robot: false

  escape_vel: -0.5

  # Goal Tolerance Parameters

  yaw_goal_tolerance: 0.157

  xy_goal_tolerance: 0.25

  latch_xy_goal_tolerance: false

  # Forward Simulation Parameters

  sim_time: 2.0

  sim_granularity: 0.02

  angular_sim_granularity: 0.02

  vx_samples: 6

  vtheta_samples: 20

  controller_frequency: 20.0

  # Trajectory scoring parameters

  meter_scoring: true # Whether the gdist_scale and pdist_scale parameters should assume that goal_distance and path_distance are expressed in units of meters or cells. Cells are assumed by default (false).

  occdist_scale:  0.1 #The weighting for how much the controller should attempt to avoid obstacles. default 0.01

  pdist_scale: 0.75  #     The weighting for how much the controller should stay close to the path it was given . default 0.6
  gdist_scale: 1.0 #     The weighting for how much the controller should attempt to reach its local goal, also controls speed  default 0.8

  heading_lookahead: 0.325  #How far to look ahead in meters when scoring different in-place-rotation trajectories

  heading_scoring: false  #Whether to score based on the robot's heading to the path or its distance from the path. default false

  heading_scoring_timestep: 0.8   #How far to look ahead in time in seconds along the simulated trajectory when using heading scoring (double, default: 0.8)

  dwa: true #Whether to use the Dynamic Window Approach (DWA)_ or whether to use Trajectory Rollout

  simple_attractor: false

  publish_cost_grid_pc: true

  #Oscillation Prevention Parameters

  oscillation_reset_dist: 0.05 #How far the robot must travel in meters before oscillation flags are reset (double, default: 0.05)

  escape_reset_dist: 0.1

  escape_reset_theta: 0.1
```

(costmap_common_params.yaml):

```
map_type: costmap

origin_z: 0.0

z_resolution: 1

z_voxels: 2

obstacle_range: 2.5

raytrace_range: 3.0

publish_voxel_map: false

transform_tolerance: 0.5

meter_scoring: true

footprint: [[-0.21, -0.165], [-0.21, 0.165], [0.21, 0.165], [0.21, -0.165]]

footprint_padding: 0.1

plugins:

- {name: obstacles_layer, type: "costmap_2d::ObstacleLayer"}

- {name: inflater_layer, type: "costmap_2d::InflationLayer"}

obstacles_layer:

  observation_sources: scan

  scan: {sensor_frame: front_laser, data_type: LaserScan, topic: front/scan, marking: true, clearing: true, min_obstacle_height: -2.0, max_obstacle_height: 2.0, obstacle_range: 2.5, raytrace_range: 3.0}

inflater_layer:

 inflation_radius: 0.30
```

(robot_localization_with_gps.yaml):

```
#Configuation for robot odometry EKF

frequency: 50



odom0: /jackal_velocity_controller/odom

odom0_config: [false, false, false,

               false, false, false,

               true, true, true,

               false, false, false,

               false, false, false]

odom0_differential : true



imu0: /imu/data

imu0_config: [false, false, false,

              true, true, true,

              false, false, false,

              true, true, true,

              false, false, false]

imu0_differential : false



odom1: /odometry/gps

odom1_config: [false, false, false,

               false, false, false,

               true, true, true,

               false, false, true,

               false, false, false]

odom1_differential : false



odom_frame: odom

base_link_frame: base_link

world_frame: map

map_frame: map







process_noise_covariance: [0.05, 0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    0,

                            0,    0.05, 0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                        0,    0,    0.06, 0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                        0,    0,    0,    0.03, 0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                        0,    0,    0,    0,    0.03, 0,    0,     0,     0,    0,    0,    0,    0,    0,    0,
                        0,    0,    0,    0,    0,    0.06, 0,     0,     0,    0,    0,    0,    0,    0,    0,
                        0,    0,    0,    0,    0,    0,    0.025, 0,     0,    0,    0,    0,    0,    0,    0,
                        0,    0,    0,    0,    0,    0,    0,     0.025, 0,    0,    0,    0,    0,    0,    0,
                        0,    0,    0,    0,    0,    0,    0,     0,     0.04, 0,    0,    0,    0,    0,    0,
                        0,    0,    0,    0,    0,    0,    0,     0,     0,    0.01, 0,    0,    0,    0,    0,
                        0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0.01, 0,    0,    0,    0,
                        0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0.02, 0,    0,    0,
                        0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    0.01, 0,    0,
                        0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    0.01, 0,
                        0,    0,    0,    0,    0,    0,    0,     0,     0,    0,    0,    0,    0,    0,    0.015]


initial_estimate_covariance: [1e-9, 0,    0,    0,    0,    0,    0,    0,    0,    0,     0,     0,     0,    0,    0,
                              0,    1e-9, 0,    0,    0,    0,    0,    0,    0,    0,     0,     0,     0,    0,    0,
                              0,    0,    1e-9, 0,    0,    0,    0,    0,    0,    0,     0,     0,     0,    0,    0,
                              0,    0,    0,    1e-9, 0,    0,    0,    0,    0,    0,     0,     0,     0,    0,    0,
                              0,    0,    0,    0,    1e-9, 0,    0,    0,    0,    0,     0,     0,     0,    0,    0,
                              0,    0,    0,    0,    0,    1e-9, 0,    0,    0,    0,     0,     0,     0,    0,    0,
                              0,    0,    0,    0,    0,    0,    1e-9, 0,    0,    0,     0,     0,     0,    0,    0,
                              0,    0,    0,    0,    0,    0,    0,    1e-9, 0,    0,     0,     0,     0,    0,    0,
                              0,    0,    0,    0,    0,    0,    0,    0,    1e-9, 0,     0,     0,     0,    0,    0,
                              0,    0,    0,    0,    0,    0,    0,    0,    0,    1e-9,  0,     0,     0,    0,    0,
                              0,    0,    0,    0,    0,    0,    0,    0,    0,    0,     1e-9,  0,     0,    0,    0,
                              0,    0,    0,    0,    0,    0,    0,    0,    0,    0,     0,     1e-9,  0,    0,    0,
                              0,    0,    0,    0,    0,    0,    0,    0,    0,    0,     0,     0,     1e-9, 0,    0,
                              0,    0,    0,    0,    0,    0,    0,    0,    0,    0,     0,     0,     0,    1e-9, 0,
                              0,    0,    0,    0,    0,    0,    0,    0,    0,    0,     0,     0,     0,    0,    1e-9]
```

(mymap_empty.yaml):

```
image: mymap_empty.pgm

resolution: 0.020000

origin: [-10.000000, -20.240000, 0.000000]

negate: 0

occupied_thresh: 0.65

free_thresh: 0.196
```