**Swag\_gazebo package:**

**Imu\_to\_rpy**

This script subscribes the IMU message information in the format of ROS/IMU message format

i.e, header, frame id and orientation………

o/p:

Converts into simple vector format

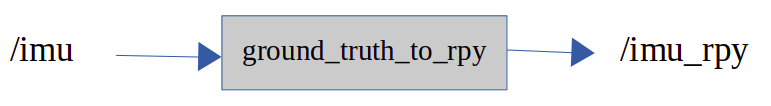
Vector.x = r

Vector.y = p

Vector.z = y

Subscribes : /imu

Publish : /imu\_rpy

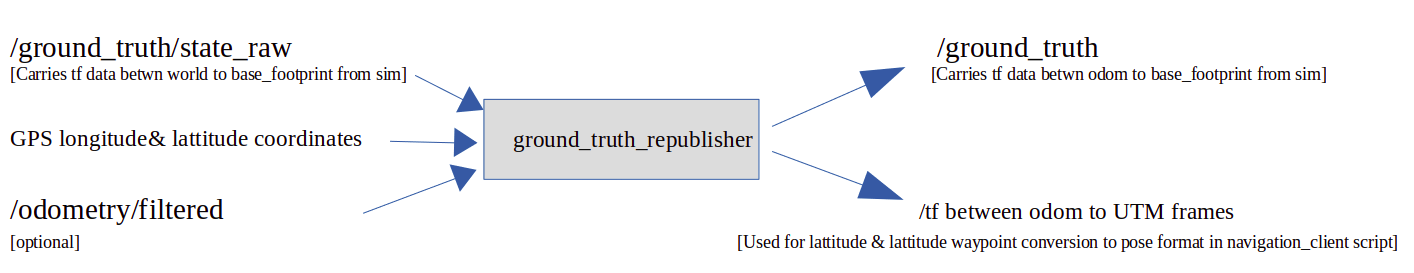


**Ground\_truth\_republishe**r

Uses robot odometry and GPS longitude-latitude, and performs conversion of UTM coordinates to normal position xyz and quaternion conversions.

Subs: /ground\_truth/state\_raw, /odometry\_filterd

Pub: /ground\_truth



Broadcasting tf trames data between

odom → Base\_footprint

odom → utm or utm → odom

**Quaternion\_to\_rpy**

This script subscribes the ground\_truth message information in the format of ROS/odometry message format

i.e, header, frame id and orientation………

o/p:

Converts into simple vector format

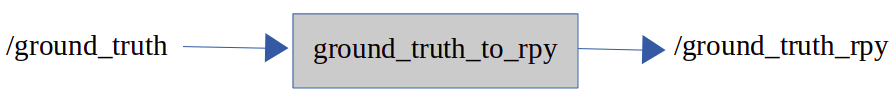
Vector.x = r

Vector.y = p

Vector.z = y

Subscribes : /ground\_truth

Publish :/ground\_truth\_rpy

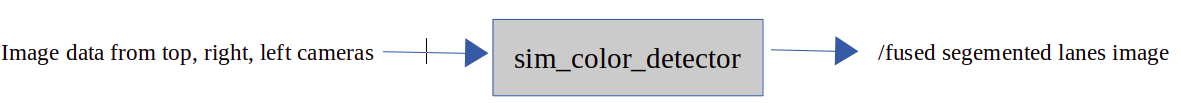


**controls (Only for simulation)**

Uses advanced simulated control architecture for commanding simulated motors.

**sim\_color\_detector**

This script fuses right, left and front cameras images into segmented lane information/image.



**Note:** If you do not find “ground\_truth/state\_raw” in the rostopic list used below plugin, it will publish tf between world frame to base\_footprint. This information can be used for global gps position adjustments.

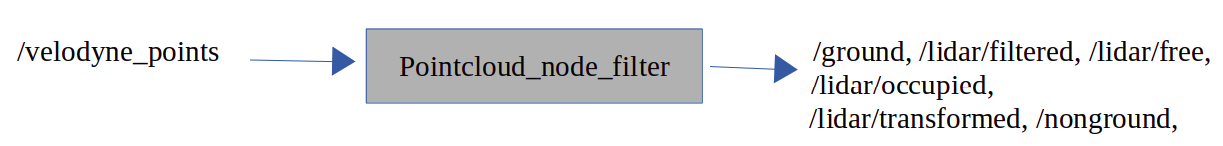
<plugin name="p3d\_base\_controller" filename="libgazebo\_ros\_p3d.so">

If this plugin does not work in real time , use static state publisher.

**Swag\_perception:**

**Pointcloud\_node\_filter**

Uses point cloud filter library(PCL) and applies various filter approaches and provides obstacle detection information. Later this can be used for obstacle avoidance use case.



# 

# **GPS based navigation using ROS navigation stack**

# **navsat\_transform\_node**

This node converts the gps coordinates (longitude and latitude) information into a world coordinate frame.

**Inputs:** GPS coordinate messages, imu heading information and odometry source information(optional).

**Outputs:**  ROS Odometry message(linear xyz & angular xyz) format converted gps coordinate data.

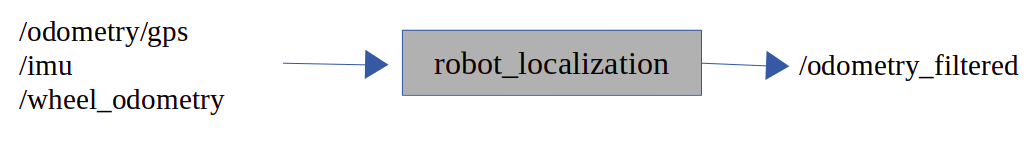
**Overview**

navsat\_transform\_node takes as input a [nav\_msgs/Odometry](http://docs.ros.org/api/nav_msgs/html/msg/Odometry.html) message (usually the output of ekf\_localization\_node or ukf\_localization\_node), a [sensor\_msgs/Imu.html](http://docs.ros.org/api/sensor_msgs/html/msg/Imu.html) containing an accurate estimate of your robot’s heading, and a [sensor\_msgs/NavSatFix.html](http://docs.ros.org/api/sensor_msgs/html/msg/NavSatFix.html) message containing GPS data. It produces an odometry message in coordinates that are consistent with your robot’s world frame. This value can be directly fused into your state estimate.

**robot\_localization**

Provides nonlinear state estimation through sensor fusion of an arbitrary number of sensors.

Fuses the different odometry sources(GPS(navsat\_transform conversion data), Wheel encoder, visual odom, Imu data and provides estimated odometry filtered data.



**Note:** robot\_localization node reads ekf\_localization\_node\_params.yaml. This contains all the required covariance matrices and topics information to process EKF localisation algorithm.

**Navigation**

