Localisation multi-capteurs

1 Filtre de Kalman pour estimation angulaire

Un exemple de filtre de Kalman est donné dans le package **attitude_estimation** fourni (source DroMOOC, `https://github.com/dromooc/attitude_estimation` – à télécharger manuellement ou via git clone). Des éléments de description supplémentaires sont fournis en pages suivantes.

Le cas d'étude proposé est l'estimation de l'angle de tangage (pitch) à partir de mesures gyrométriques (vitesse angulaire) et accélérométriques (accélérations, combinées pour fournir une mesure d'inclinaison) issues d'une centrale inertielle réelle. Ces différentes mesures étant entachées de bruits et biais, elles ne peuvent fournir seules une estimation de l'angle d'un drone, il faut donc les combiner via un filtre.

1. Tester les différents launch fournis dans le package :
   — *imu_data_no_motion.launch* visualise les mesures sans mouvement, afin d'évaluer les bruits.
   — *imu_data_pitch_only.launch* visualise les mesures lors d'un mouvement de tangage.
   — *KF_pitch_imu_data_pitch_only.launch* lance le filtre de Kalman sur le jeu de données précédent. Modifier les paramètres de réglage du filtre (console dynamic_reconfigure lancée automatiquement) afin de donner plus d'importance à l'une ou l'autre des mesures et visualiser l'effet.
2. Modifier le script **KFPitchEstimation.py** afin d'estimer également l'angle de roulis (roll) – se référer aux slides de la suite du document pour le modèle de mesure fourni par les accéléromètres. Créer un nouveau launch à partir de l'existant afin de visualiser le résultat.

2 Robot mobile Husky

Le modèle dynamique d'évolution du robot mobile Husky (cf TPs précédents) est de la forme suivante :

$$\begin{cases} x_{k+1} = x_k + \Delta t \cdot v_k \cdot \cos(\theta_k) \\ y_{k+1} = y_k + \Delta t \cdot v_k \cdot \sin(\theta_k) \\ \theta_{k+1} = \theta_k + \Delta t \cdot \omega_k \end{cases} \tag{1}$$

L'objectif est de reconstruire l'état du robot $[x, y, \theta]$ à partir des mesures disponibles sur les topics suivants :
— **/imu/data**, qui regroupe à la fois des mesures brutes d'accéléromètres et gyromètres telles que celles étudiées dans la partie précédente « estimation angulaire », et un message « orientation » qui contient un quaternion déjà filtré dont on peut extraire directement une valeur fiable de l'angle θ (yaw).
— **/navsat/vel** : mesure de vitesse (GPS) en repère global, équivalente dans le plan (x, y) à $[v\cos\theta, v\sin\theta]^T$.
— **/husky_velocity_controller/odom** : vitesse linéaire v et vitesse angulaire ω (cette dernière pouvant être peu fiable en fonction de l'adhérence des roues au sol).

Les points ci-dessous sont à réaliser dans le package ayant servi à la navigation du robot Husky. Le topic **/odometry/filtered** utilisé dans le TP de navigation autonome fournit une estimation complète de l'état du robot, et doit être utilisé pour comparer les résultats obtenus. Le déplacement du robot peut être indifféremment en boucle ouverte (publication sur topic) ou boucle fermée (cf TP navigation autonome).

1. A partir de la mesure de vitesse GPS (/navsat/vel), estimer directement la position x, y par intégration.
2. Estimer par ailleurs x, y en utilisant la vitesse linéaire mesurée par odométrie et l'angle θ fourni par la centrale inertielle, comparer les différentes approches.

DroMOOC

Sensor fusion and state estimation

Basic Level

Kalman filtering for attitude estimation

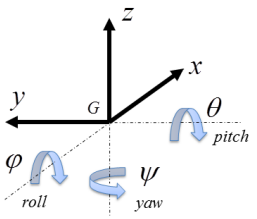
Sylvain BERTRAND

ONERA

Attitude estimation problem

Estimation of attitude angles (roll, pitch, yaw) is mandatory

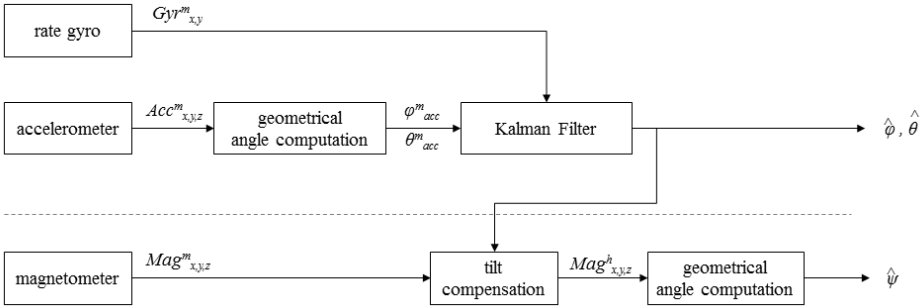
- for localization of the drone
- for control of the drone



In this lecture:

- Attitude estimation from measurements provided by an Inertial Measurement Unit (IMU) accelerometers, rate gyros + magnetometers
- Use of a Kalman Filter for estimation of roll and pitch from accelero and gyro measurements
- Estimation of yaw by geometrical computation from magnetometer measurements

Block diagram



Remarks

- Rate gyro bias estimated for roll and pitch
- Compensation by KF for possible initial error on pitch and roll estimates
- For yaw estimation: offset to be performed wrt to local reference frame (eg. zero yaw at take-off)

Roll and pitch angle estimation using KF

Model of rate gyro measurement: $Gyr_x^m = \dot{\phi} + b_x + n_x$

Assumption on the bias: $b_x \sim \text{cste} \Rightarrow \dot{b}_x = n_x^b$

Discrete-time model at sampling period T_s:

$$\begin{cases} \phi(k+1) = \phi(k) + T_s.Gyr_x^m(k) - T_s.b_x(k) - T_s.n_x(k) \\ b_x(k+1) = b_x(k) + T_s.n_x^b(k) \end{cases} \quad (1)$$

Roll and pitch angle estimation using KF

For roll and pitch:

$$\begin{cases} \phi(k+1) = \phi(k) & -T_s.b_x(k) + T_s.Gyr_x^m(k) & -T_s.n_x(k) \\ b_x(k+1) = b_x(k) & & +T_s.n_x^b(k) \end{cases} \quad (2)$$

$$\begin{cases} \theta(k+1) = \theta(k) & -T_s.b_y(k) + T_s.Gyr_y^m(k) & -T_s.n_y(k) \\ b_y(k+1) = b_y(k) & & +T_s.n_y^b(k) \end{cases} \quad (3)$$

Model can be written as a LTI system: $X(k+1) = AX(k) + BU(k) + V(k)$ with

$$A = \left[\begin{array}{cc|cc} 1 & -T_s & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & -T_s \\ 0 & 0 & 0 & 1 \end{array}\right] \qquad B = \left[\begin{array}{c|c} T_s & 0 \\ 0 & 0 \\ \hline 0 & T_s \\ 0 & 0 \end{array}\right]$$

Reminder on Kalman filtering

"Required information" to implement and run the filter

- model of the system (discrete-time LTI): A, B, C
- state and measurement noise characteristics: Q, R

  $V(k) \sim \mathcal{N}(0, Q) \qquad W(k) \sim \mathcal{N}(0, R)$
- initial state estimate and estimation error covariance matrix: $\hat{X}(0)$, $P(0)$
- online information: $U(k)$ (for prediction), $Y(k)$ (for update)

Output provided by the filter:

- state estimate $\hat{X}(k)$
- estimation error covariance $P(k)$

Roll and pitch angle estimation using KF

Geometrical computation of roll and pitch from accelerometer measurements

$$\tan(\phi_{Acc}^m) = \frac{Acc_y^m}{Acc_z^m} \qquad \tan(\theta_{Acc}^m) = \frac{-Acc_x^m}{\sqrt{\left(Acc_y^m\right)^2 + \left(Acc_z^m\right)^2}} \quad (4)$$

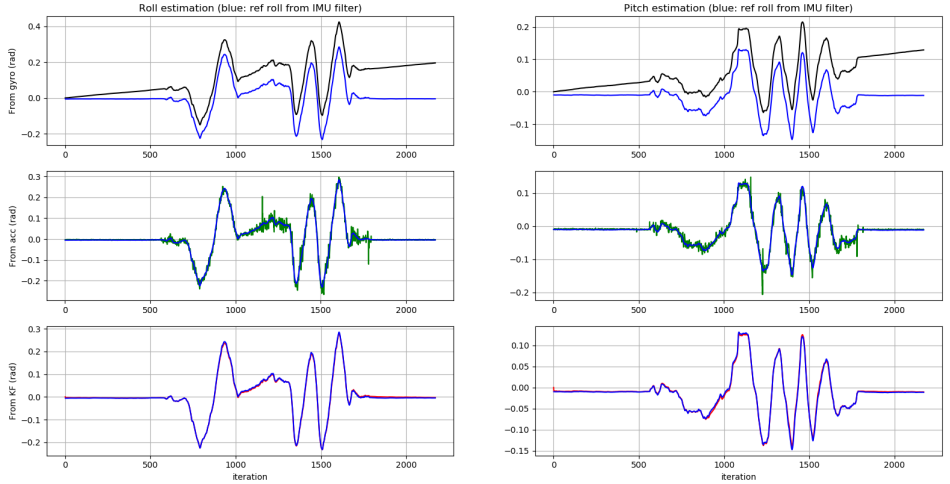
Taking

$$Y = \begin{bmatrix} \phi_{Acc}^m \\ \theta_{Acc}^m \end{bmatrix} = \begin{bmatrix} \operatorname{atan2}\left(Acc_y^m, Acc_z^m\right) \\ \operatorname{atan2}\left(-Acc_x^m, \sqrt{\left(Acc_y^m\right)^2 + \left(Acc_z^m\right)^2}\right) \end{bmatrix}$$

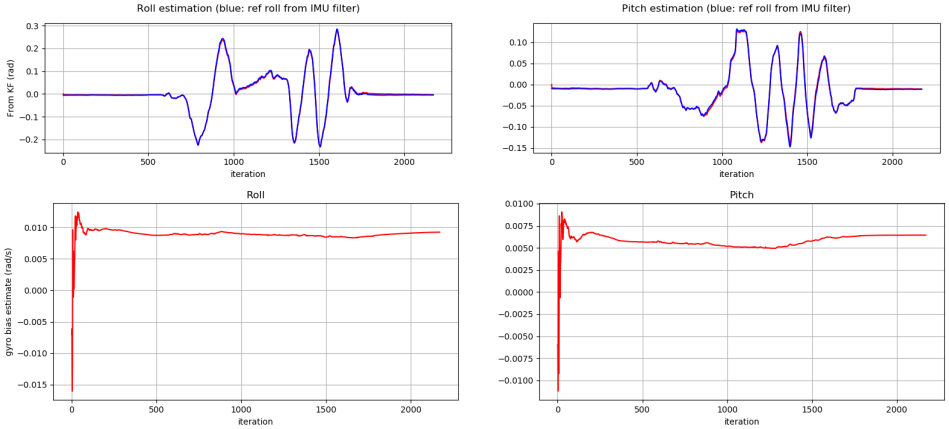
one has the measurement equation

$$Y(k) = CX(k) + W(k) \qquad \text{with} \qquad C = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right]$$

Example from real IMU measurements



Example from real IMU measurements



Yaw angle estimation

Geometrical computation of yaw angle from magnetometer measurements

- If no roll and pitch:

$$\tan(\psi) = \frac{-Mag_y^m}{Mag_x^m}$$

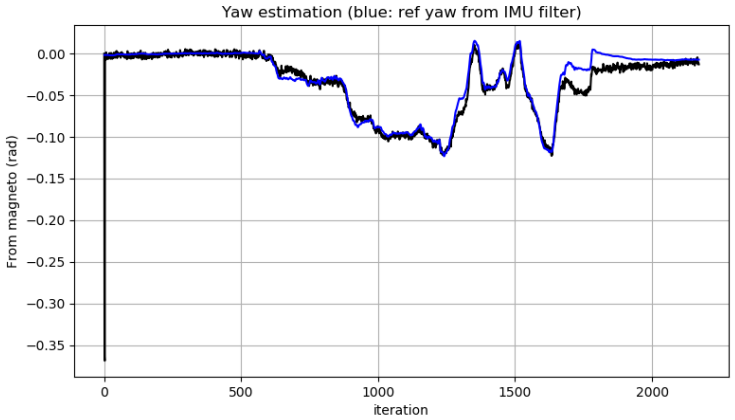
- Tilt compensation due to non zero roll and pitch:

$$Mag^h = R_{\theta,\phi} Mag^m \qquad \psi = \operatorname{atan2}\left(-Mag_y^h, Mag_x^h\right) \tag{5}$$

with

$$R_{\theta,\phi} = \begin{bmatrix} c_\theta & s_\phi . s_\theta & c_\phi . s_\theta \\ 0 & c_\phi & -s_\phi \\ -s_\theta & s_\phi . c_\theta & c_\phi . c_\theta \end{bmatrix}$$

Example from real IMU measurements



Attitude estimation: conclusions

- Simple approach for attitude estimation using IMU measurements
  - Assumption: small angles and small accelerations
  - Not robust to magnetic perturbations
- Improvements and more adavanced methods
  - Use additional sensors (eg. GPS, vision, etc.)
  - Advanced Kalman Filters (EKF, UKF)
  - Complementary filters, nonlinear observers
  - Other attitude representations (eg. quaternions)

Kalman Filter for attitude estimation
DroMOOC data set

The *attitude_estimation* ROS package provides different data set recorded from an Inertial Measurement Unit (IMU) and implements a Kalman Filter (KF) for estimation of attitude angles, as well as graphical tools for parameter tuning and visualization.

Detailed instructions on how to install and run the Virtual Machine can be found in the document "Importing and starting the DroMOOC Virtual Machine".

The *attitude_estimation* package is located in the `~/catkin_ws/src/attitude_estimation` directory of the Virtual Machine and in the DroMOOC Github repository. Before first use, it is recommended to update the package located on your Virtual Machine by entering the following command lines in a terminal (internet connection is required):

```
roscd attitude_estimation
git pull
```

If you need some help, use the contact us link on the website.

1. Understanding the data set

Measurements provided by an Inertial Measurement Unit (IMU) have been recorded in the following conditions to provided different data sets:

- still horizontal IMU (no motion): no_motion.bag
- small rotations around pitch axis[1]: pitch_only.bag
- small rotations around roll, pitch and yaw axis: roll_pitch_yaw.bag

Data sets have been recorded and can be played using the rosbag tools. They are located in the directory *bags* of the *attitude_estimation* package.

Measurements are recorded as Imu and MagneticField messages defined by the ROS sensor_msgs package and hence contain the following time-stamped information:

- 3 axis accelerometer measurements (**linear accelerations**)
- 3 axis rate-gyro measurements (**angular velocities**)
- 3 axis magnetometer measurements (**magnetic field**)
- Quaternion computed by the internal processing of the IMU

The **attitude estimation problem** consists in computing **roll, pitch and yaw angles** (see Figure 1) from the sensor measurements. Estimated angles will be compared to angles deduced from the quaternion computed by the internal processing of the IMU, which will be assumed accurate enough to be considered as ground truth.

[1] Data sets are also provided for small rotations around roll axis (roll_only.bag) and yaw axis (yaw_only.bag) only, and for rotations around the three axes (roll_pitch_yaw.bag).

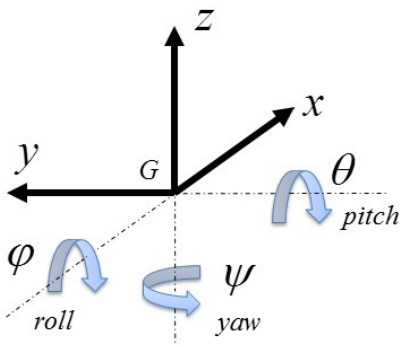


Figure 1. Simplified definition of Roll-Pitch-Yaw

2. Characteristics of the sensors

The first data set (*no_motion.bag*) can be used to analyze some characteristics of the sensors, namely their bias and noise variance.

To play this record and visualize the data, open a new terminal and enter the command line:

```
roslaunch attitude_estimation imu_data_no_motion.launch
```

It will open two graphical interfaces with time plots of the acceleration, angular velocity and magnetic field measurements (see Figure 2). A RViz window also presents a 3D arrow which corresponds to the normal (local z-axis) of the IMU. This arrow will remain aligned to the "vertical" for the data set with no motion.

The data recorded in this bag are also made available in a CSV file for more convenience.

You can use this CSV file directly to compute the mean and variance of the acceleration and angular velocity measurements over time. It will provide information on the bias of the sensors and variance of their noise. Note that acceleration due to gravity must be substracted from the acceleration signal along the z-axis.

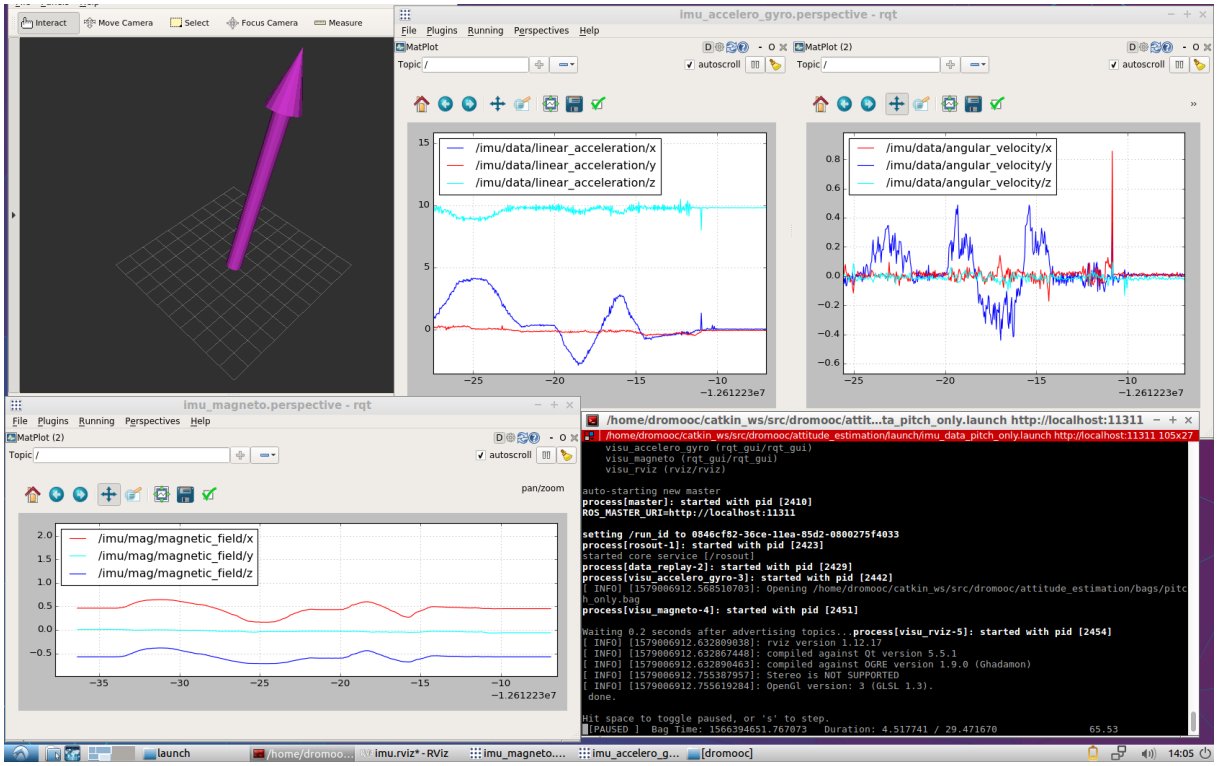


Figure 2. Graphical interfaces for visualization of data sets
(example on the *pitch_only.bag* data set)

3. Pitch estimation using Kalman Filter

3.1. Data set

You can then visualize the second data set related to pitch motion by opening a new terminal and entering the command line:

```
roslaunch attitude_estimation imu_data_pitch_only.launch
```

3.2. Python implementation of a Kalman Filter

As presented in the video lectures, a Kalman Filter will be used to estimate online the pitch angle from the measurements of the data set. A Python implementation of a linear Kalman Filter is provided in the script *KalmanFilter.py* located in the *script* directory of the *attitude_estimation* package. Take some time to understand this implementation and the way it can be used by looking at the example at the end of this script. You can run the script to visualize some graphical results on this example.

Note that this implementation is generic in the sense that you can use it for other purposes by simply importing the KalmaFilter module in any other Python code (*import KalmanFilter*).

3.3. Use of the Kalman Filter for pitch estimation

The Python file KFPitchEstimation.py implements a ROS node with the Kalman Filter for pitch estimation. You can try the code by opening a new terminal and entering the command line:

```
roslaunch attitude_estimation KF_pitch_imu_data_pitch_only.launch
```

It will play the pitch_only.bag data set, run the node implemented in KFPitchEstimation.py and open a graphical user interface that allows online tuning of the parameters

This command line starts the replay of the *pitch_only.bag* data set, runs the node implemented in *KFPitchEstimation.py* and also starts a graphical user interface based on RQT (Figure 3). This interface enables to tune parameters of the filter (Figure 3 – top), and to plot the time evolutions of the "true" pitch angle to be estimated (ground truth – in red) and of the pitch angle estimated by the Kalman filter (in blue).

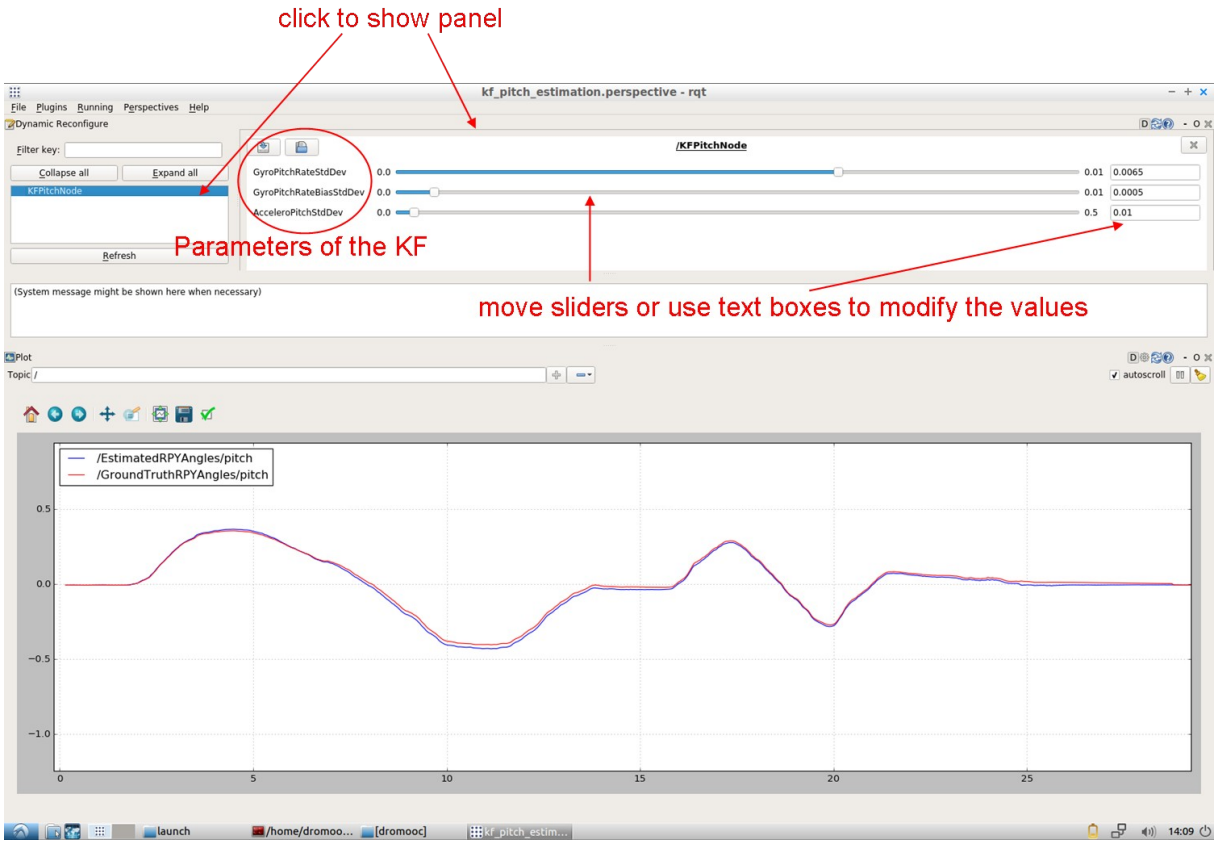


Figure 3 – Graphical user interface associated to the Kalman Filter for pitch estimation

3.4. Understanding the code

The simulator package is composed of several directories:

- `scripts`: Python files for Kalman filter and pitch estimation node
- `bags`: bags file with data sets

- `msg`: custom message formats used by the code
- `launch`: launch files that are used to run several nodes
- `doc`: this document

The node for KF pitch estimation (/KFPitchNode) requires acceleration and angular velocity measurements provided by the IMU. They are given by the /imu/data topic generated by the replay of the bag file (/data_replay node).

The node for KF pitch estimation publishes the estimated pitch angle (/EstimatedRPYAngles/ pitch topic) and is also responsible for the publication of the ground truth value of the pitch angle (/GroundTruthRPYAngles/pitch topic) computed from the quaternion provided by the internal processing of the IMU.

The main structure of the code is summarized by the following Node Graph automatically generated by ROS:

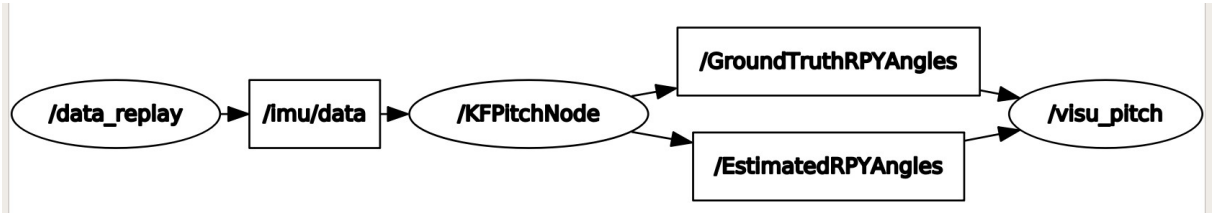


Figure 4 – Node Graph of the pitch estimation implementation