MTRN4010.2025

Project2, part B7 (EKF implementation)

Part B7 (7 marks) Implement all the previous items (except B6), put them together to estimate the attitude, in real-time. You will adapt the structure of your program in Proj1.

The process will operate in the following realistic condition.

a) We do not accurately know the initial attitude.
b) There is no initial calibration time (so that there is no estimation of gyroscopes biases).

It is because those adverse conditions that we need to exploit additional sources of information for obtaining adequate estimates of the attitude. Those extra sources of information can be provided by the 3D camera, which allows us to detect the floor and walls, flat surfaces whose orientations, in global coordinate frame (GCF) are well known for us.

For that purpose, we will need to implement an EKF state estimator, which will involve prediction steps and update steps.

For the sake of simplicity, we will only exploit two known surfaces: the floor and the wall YZ (when one of those is detected by the camera's ROI). We will not use surfaces aligned with plane XZ.

To simulate a condition in which we do not know the initial attitude, you will run your EKF only if the event time, t, is ***t>15*** seconds. This means the EKF starts to operate when the platform is already flying, far from the initial attitude when the platform was initially resting before taking off (and far from the friendly initial calibration period). You will assume expected attitude =[0;0;0] and standard deviation (associated each of its components) std =30 (if expressed indegrees) at the starting time of the EKF (t0~15 seconds).

Prediction steps

After the EKF process is started, prediction steps must always be applied, at ***every event***, as you did (or as you should have done) in Proj1 for running the attitude predictor.

Your prediction step will use the last measured **raw** gyroscope measurements, the current expected value of the attitude, and its associated covariance matrix and few additional parameters. For example,

[Xe,P]=MyPredictionStep(Xe, P, W,dt,...)

The function is called using the following input arguments:

Xe,P are the expected value and covariance matrix of the attitude estimates at time ***t-dt. W*** are the last received gyroscopes' measurements (That vector "gxyz" in our example program for Proj1.). The variable ***dt*** has the same meaning that it had in Proj1. You may also include extra parameters, depending on your particular way of implementing the function.

The function's output will be the expected value and covariance matrix of the attitude estimates, after the prediction step. In the example those do overwrite the previous content.

The Jacobian matrix $\mathbf{J} = \left[\dfrac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}} \right]$ is provided via an API function.

We assume that the noises that pollute the gyroscopes measurements are **WGN** having ***std = 3*** (if expressed in in degrees/second).

You may use the provided API function that does calculate $\mathbf{J}$

[Jx, Ju] = API.k.MkJxJu(X, W, dt);

In which ***X*** is the current attitude expected value (in radians), ***W*** is a 3x1 vector for the last gyroscopes' measurements (in radians/second), and **dt**, the time step (in seconds). The function returns: The numeric values of $\mathbf{J} = \left[\dfrac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}} \right]$ and $\mathbf{J}_u = \left[\dfrac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}} \right]$ matrixes. You will use $\mathbf{J}$. For $\mathbf{J}_u$ you will use your version.

Update steps.

Each time the predefined ROI does detect a flat patch, and it is inferred that it corresponds to the floor or to the wall, we will exploit its calculated normal vector, to perform an EKF update. You will implement what you proposed in part B4 (if the patch is parallel to global XY plane, the floor) or that of B5 (if it is parallel to YZ, the wall). The inference will be performed by the module you implemented for item B1. You will use the current expected value of the attitude for setting the attitude required by module B1.

We will use datasets of longer duration (than of those we used in Proj1). The dataset **HH04** has a duration close to 300 seconds.

You will run simultaneously three modules. One is the EKF one, which will run only for times **t>1**5 seconds. You will also run a calibrated predictor, which will run from t=0. This predictor will run from initial attitude *=[-1.5;-0.5;-2.5]* (if expressed in degrees). It will perform its calibration up to time=5 seconds (as you did in Proj1, item 3). This predictor will start from an relatively accurate initial attitude, it will also have the opportunity to estimate the gyroscopes' biases. That calibration will be valid and accurate, during the trip of 300 seconds. We will consider the attitude that is generated by this calibrated predictor to be our **ground truth**. Our EKF estimates must not drift away of this ground truth (even that it uses biased gyroscopes measurements and also uses a bad initial expected attitude). You will run an uncalibrated predictor, also for t>15 seconds, to appreciate how wrong it turns to be.

You will use the API predictor, for implementing these two instances of predictor.
You will plot the three versions of attitude estimates in a 3-channel oscilloscope. You will use these colours:

1) Uncalibrated predictor: green.
2) EKF's attitude estimates: red
3) Ground truth: blue.

In the oscilloscope, the attitude angles must be presented in degrees.

Marking criteria for B7.

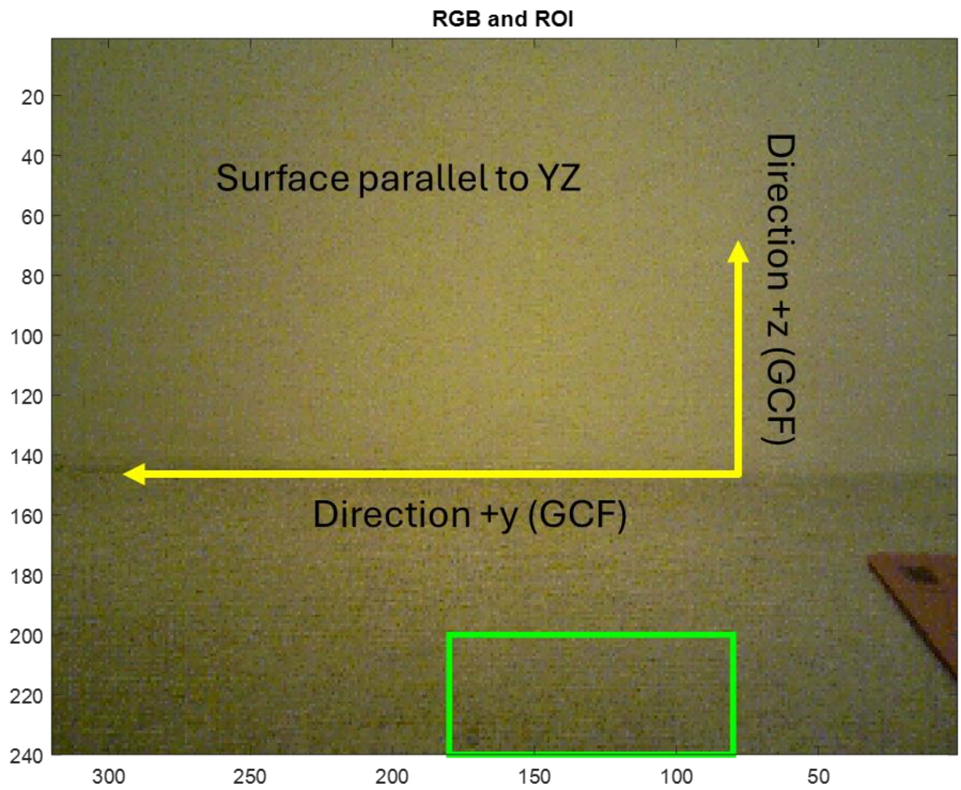
a) EKF prediction step is properly implemented (10%).
b) EKF updates, for cases B4 and B5 are implemented (10%)
c) Module for B1 is well integrated to the EKF (10%).
d) Estimates of the EKF stay close enough to those of the ground truth (tolerance =6 degrees, after 5 seconds of starting, i.e., t>20 seconds, until the end of the dataset.) **(50%)**. This item requires visualization via oscilloscope, whose specification was previously given. There are no marks given for the visualization, however, for item (d) to be accepted, the specified visualization is a necessary condition.

Additional information

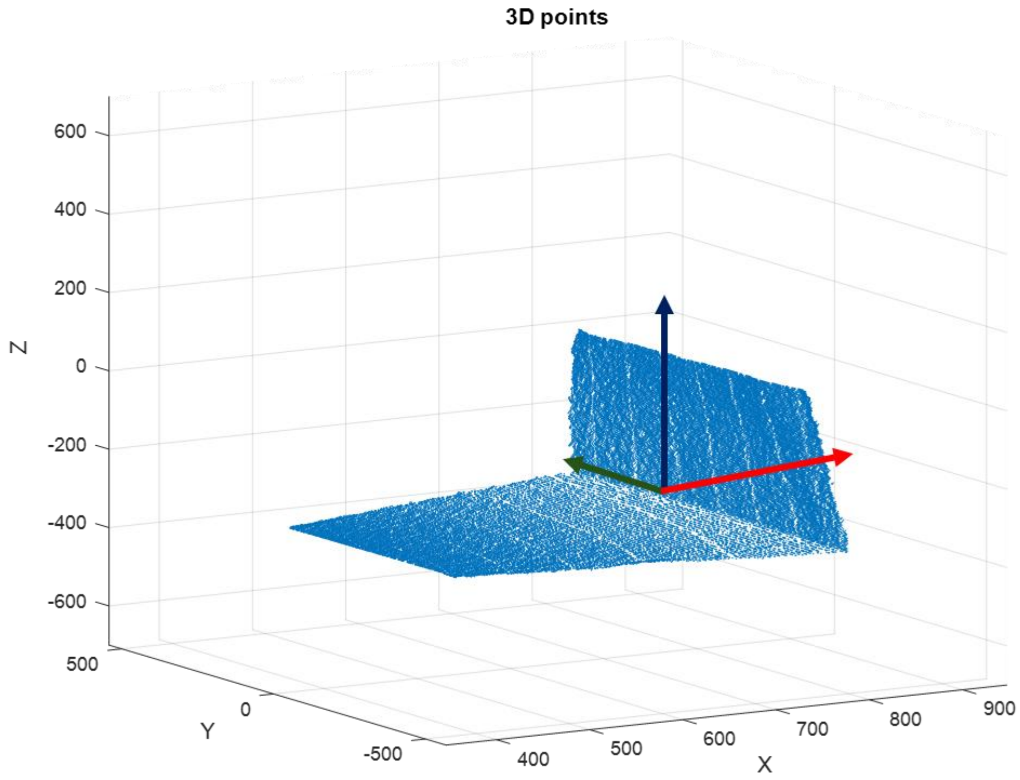
API version, to be used in Part B7**: API4010_v11();**

Dataset**: HH04**

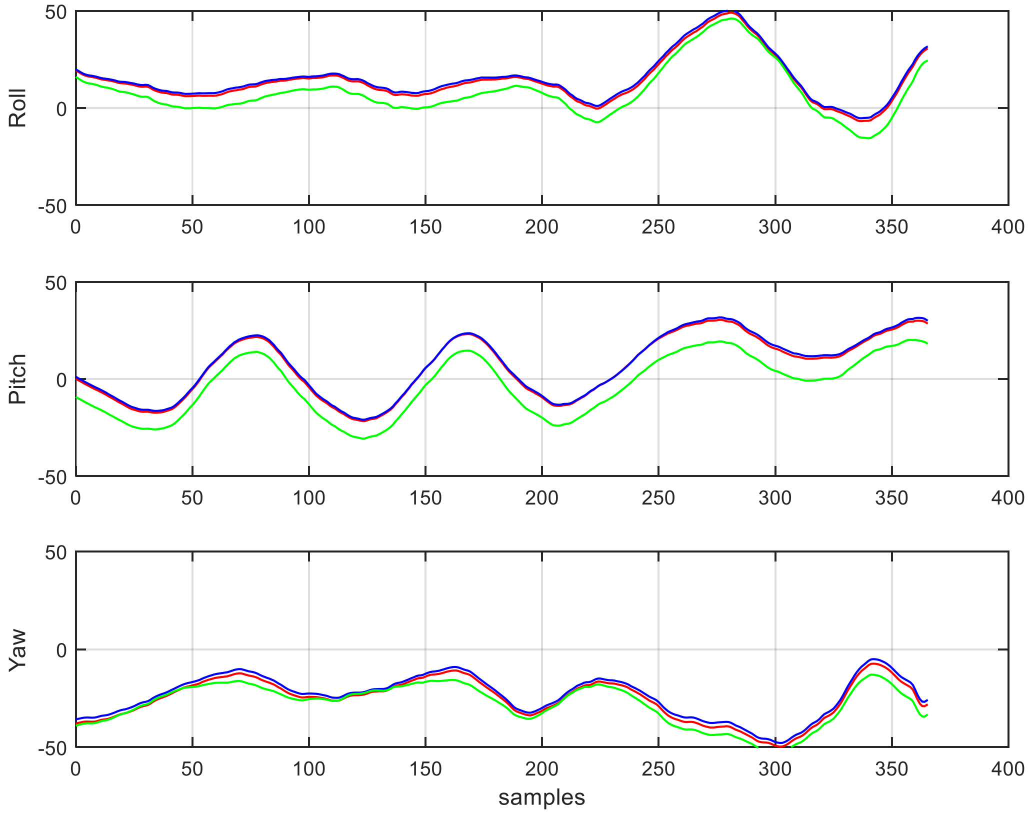
Context of operation (datasets family **HH0x)**



At t=0, the platform is parked facing the wall (plane YZ).
The predefined ROI (the same we used in Proj1) is at that moment covering a flat patch of the floor.



Directions, +x (red),+y (green),+z (blue). Those can also be appreciated by inspecting the figure axes. The wall YZ (whose normal is [-1;0;0]). The platform is always on the side the space pointed by the normal vector [-1;0;0]. Students may use other conventions. These 3D points, shown in the platform's CF, were scanned at time t=0, when the platform was almost aligned with the GCF. Later, when the platform moves, the ROI, during some intervals of time will capture patches of the front wall, or of the lateral one (but we will not use that one), and the floor again, etc. Sometimes it would over sections that are not flat patches.



Typical result. The blue curves are the GT, the red ones the EKF, the green ones the uncalibrated predictor (samples were taken every 50milliseconds). The shown samples correspond to an interval of around 20seconds (from time t=80 seconds.). It can be appreciated that the EKF estimates stay close to the GT.

Complexity of B7. The complexity of B7 has be substantially reduced in the following way: The implementation of the augmented version, for estimating the gyroscopes biases in real-time, described in part B6, was originally included in B7. Now, it has been removed, for adjusting the project to time constraints, and based on observed results in Proj1 and other assessments.

Submission deadline: Extended The deadline for Part B7 is week 10, Friday, t=23:55.
