Extended Kalman Filter for a Miniature Strapdown Inertial Measurement Unit

Attitude estimation

# Introduction : the miniature inertial unit (IMU)

This lab is based on a miniature strapdown inertial unit adapted for education by the technical team of ISAE/DCAS (figure 1).



X

Z

Y

Figure 1: the imu411 Inertial Measurement Unit

The major component is a stm32f411 unit developed by STMicroelectronics. It is a miniature device integrating a controller, several sensors and input-output capabilities. Main components are:

* A 32bits Cortex M4 based microcontroller running at 100Mhz
* A three axis gyrometer (ref: l3gd20) updated at 100Hz
* A three axis accelerometer and magnetometer (ref: lsm303dlhc) updated at 100Hz

For educational purpose we have added two sensors:

* An ultrasonic sensor (ref: srf02) **updated at 10Hz**
* A barometric sensor (ref: hca0811) updated at 100Hz

This unit can be interfaced to a USB port of a computer via a serial-USB adapter. The microcontroller embed a real time code that made available a set of measurement at a fixed frequency of 100Hz.

In order to connect the sensor to a computer the user needs to :

* Install the USB-serial converter[[1]](#footnote-1)
* Find which Port com has been installed
  + Example: Gestionnaire de périphériques / Ports (COM et LPT) / USB Serial Port (COM4)
* Set the latency delay of the serial port at 1ms[[2]](#footnote-2)

The sensor units measurement are then accessed via the virtual serial port.

A Matlab function (imu41.m) is provided in order to access the sensor’s measurement in pseudo real time. The syntax is as follows in Matlab:

* Open the serial port (here COM4):

>> imu411(‘open’,’COM4’)

* Read the sensor’s value:

>> d = imu411(’read’)

This functions pauses the Matlab execution until the next measurement is available (0.01s)

* Close the serial port (if needed):

>>imu411(‘close’)

The value given by the function ‘read’ is a12 component vector (1 component for the current time and 11 components for the sensors values). Units are as follows:

d(1) = Time (s)

d(2:4) = Gyroscope X,Y,Z (°/s)

d(5:7) = Accelerometer X,Y,Z (g)

d(8:10) = Magnetometer X,Y,Z (Gauss)

d(11) = Barometer (Millibar)

d(12) = Ultrasonic sensor, facing up (cm)

# Simulation Mode

The imu411() function provides a simulation mode that replays recorded synthetic observation data along with the true states. The observations feature sampled additive zero mean Gaussian white noise. The simulation mode can be helpful to tune the filter matrices, since a reference of the true states is available.

To enter simulation mode, replace the command

imu411('open',COM);

by

imu411('open',COM, [name of a simlation data file as a string]);

Three maneuvers are provided :

* Roll maneuver for the 2D filter (simimu\_2Dmaneuver.mat)
* Yaw, Pitch, Roll maneuver for the 3D filters (simimu\_3Dmaneuver.mat)
* Static data for calibration (simimu\_static.mat)

Note : Don’t forget to use either simulated data or real measurements in both the calibration script and the filter script. Re-run the calibration after changing modes.

# ATtitude representations

The reference (XYZ) frame is fixed to the earth surface, considered to be Galilean. The moving (xyz) frame is fixed to the IMU.

The final goal is to compute the full orientation of (xyz) with respect to the reference frame (XYZ).

## Euler angles

The classical (and most intuitive) representation is the set of Euler angles : yaw, pitch and roll (y,p,r). Euler angles are routinely used in many application because they minimize the dimensionality of the problem. However they introduce mathematical singularities when pitch approaches (known as “gimbal lock”).

## Orientation matrix (Direct Cosine Matrix)

More generally the Rotation Matrix specifies totally the rotation. Let’s take the example of a vector with coordinates with respect to frame If the frame is obtained by a rotation of by M, the coordinates of in (xyz) are:

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

M is real, orthogonal matrix.

## Quaternions

A rotation of angle about an axis is represented by the quaternion:

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

Euler angle do not present the singularity problem as Euler angles.

## From quaternion to orientation matrix

The quaternion to orientation matrix conversion is:

|  |  |  |
| --- | --- | --- |
|  | function M = quat2M(q)  M = [q0^2+q1^2-q2^2-q3^2, 2\*(q1\*q2-q0\*q3), 2\*(q0\*q2+q1\*q3)  2\*(q1\*q2+q0\*q3), q0^2-q1^2+q2^2-q3^2, 2\*(q2\*q3-q0\*q1)  2\*(q1\*q3-q0\*q2), 2\*(q0\*q1+q2\*q3), q0^2-q1^2-q2^2+q3^2]; | (8) |

## Derivative of a quaternion

When the rotational speed of the moving frame (xyz) is (p,q,r) (rotation speed expressed in the (x,y,z) frame) the quaternion derivative is given by:

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

# 2D orientation estimation



Figure 2: a spirit level

A classical “spirit level” (fig 2) is horizontal when the gravity vector is perfectly orthogonal to the device. An electronic “spirit level”, instead of using a mechanical device (the bubble) uses an accelerometer.

Template : *kalman2d.m*

## A basic spirit level

As a trial run, we only estimate the roll angle using the accelerometer sensor along y and z axis. The state is one dimensional : .

* Write the state space equation : state update, output (nonlinear) equation…
* Compute the Jacobians
* Code and test the EKF in real time
* Try different values of Q and R

## An enhanced spirit level

We now take into account the gyrometer measurement (x axis). The state is two dimensional : .

* Modify the code and observe how the “spirit level” performance is enhanced.

# 3D orientation estimation

## A basic EKF for 3D estimation

The template kalman3d.m codes a full 3D estimator based on accelerometer and magnetometer sensor.

The full state is the orientation quaternion:

The measurement vector is made of accelerometer and magnetometer readings:

The accelerometer measures, in the moving frame of the sensor, the full acceleration of the device : the sum of gravity acceleration and the inertial acceleration :

The contribution of the inertial acceleration is considered as a perturbation .

The magnetometer measures, in the moving frame of the sensor, the magnetic field : the sum of earth magnetic field plus other perturbation fields :

## An EKF 3D orientation estimator

We now take into account the gyrometer reading. The gyrometer can be considered in two ways: as a “new sensor” or as a “control input”.

### The gyrometer as a « new sensor »

The state vector is made of the orientation quaternion augmented with the rotation speed about xyz directions:

The measurement vector is augmented with the gyrometer readings:

Code the EKF, starting from the 3D template. Do not forget to:

* Adapt the size of vectors and matrixes
* Adapt the prediction step (with equation (9))
* Adapt the update step

### The gyrometer as a « control input »

The full state is the orientation quaternion:

The measurement vector is made of accelerometer and magnetometer readings:

In this version the updated step is enhanced by the gyrometer reading, using equation (9).

# Discussion

* An aircraft is often subject to a constant acceleration (centripetal acceleration during a coordinated turn). Is the EKF coded in the previous chapter adapted to this situation? Do you suggest any improvement?
* If the sensor is designed for a specific purpose (human held, automotive, quadrotor, etc.) how can the code be enhanced ?
* For many applications the gyrometer bias is not constant and must be estimated. Explain how to modify the code in order to estimate this bias.
* Etc.

1. Needs administrator rights [↑](#footnote-ref-1)
2. Needs administrator rights [↑](#footnote-ref-2)