Simen Bootsma
19-10-2018

Filter acceleration data

From the Xsens MTi-3 chip, we receive the x,y,z-acceleration data in the local frame. There appears to be some offset and some high frequency noise in this data. This document will give several options to improve this.

Internal filter

During the initialization of the MTi at startup, it should remove its offset for both the acceleration and gyroscope data. This init function is called in main.c:

```
MTi = MTi_Init(NO_ROTATION_TIME, XSENS_FILTER);
```

It has two parameters: the initialization time during which it should not move and the filter setting. The first parameter is equal to the time you can assure the robot to be still after startup. Increasing this number will decrease the offset, but the offset will not improve significantly when this value is higher than 3 seconds according to Xsens. The second parameter is one out of five options for the filter settings, they are listed in the table below.

Table 1: Different MTi filter types (source: MTi-1 series datasheet)

Name	Number	Product	Description
General	50	MTi-3	Suitable for most applications.
High_mag_dep	51	MTi-3	Heading corrections strongly rely on the magnetic field measured and should be used when magnetic field is homogeneous.
Dynamic	52	MTi-3	Assumes that the motion is highly dynamic.
North_referenced	53	MTi-3	Assumes a good Magnetic Field Mapping (MFM) and a homogeneous magnetic field. Given stable initialization procedures and observability of the gyro bias, after dynamics, this filter profile will trust more on the gyro solution and the heading will slowly converge to the disturbed mag field over the course of time.
VRU_general	54	MTi-2 MTi-3	Magnetometers are not used to determine heading. Consider using VRU_general in environments that have a heavily disturbed magnetic field or when the application only requires unreferenced heading (see also Section 4.3.3).

CMSIS library filters

The internal filter is the basic solution to removing the offset. However, in the case that this doesn't work properly or you want to remove the high frequency noise, you can choose to make use of the filter functions from the CMSIS library, which is already included in the `roboteam_microcontroller3.0` repository. I will walk you through the steps I took to apply an IIR filter to remove the bias.

First I designed the high pass filter using the MATLAB tool *filterDesigner*. I applied this filter to some data I recorded from the robot to see if it worked properly. Once I was satisfied with the filter, I exported it to coefficients such that I had a SOS-matrix and a gain matrix. I used this tool to convert these coefficients to a header file that can be used by the CMSIS functions, explanation can be found here. In that explanation Phil Birch also shows how to use the CMSIS library functions. It basically boils down to:

Include the library and your header file with coefficients.

```
#include "arm_math.h"
#include "coefficients.h"
```

Create variables that are used in the functions.

```
float32_t pState[NUM_SECTIONS*4]={0};  //NUM_SECTIONS is defined in coefficients.h
arm_biquad_casd_df1_inst_f32 S;   //structure that defines the filter
```

Initialize the filter function once.

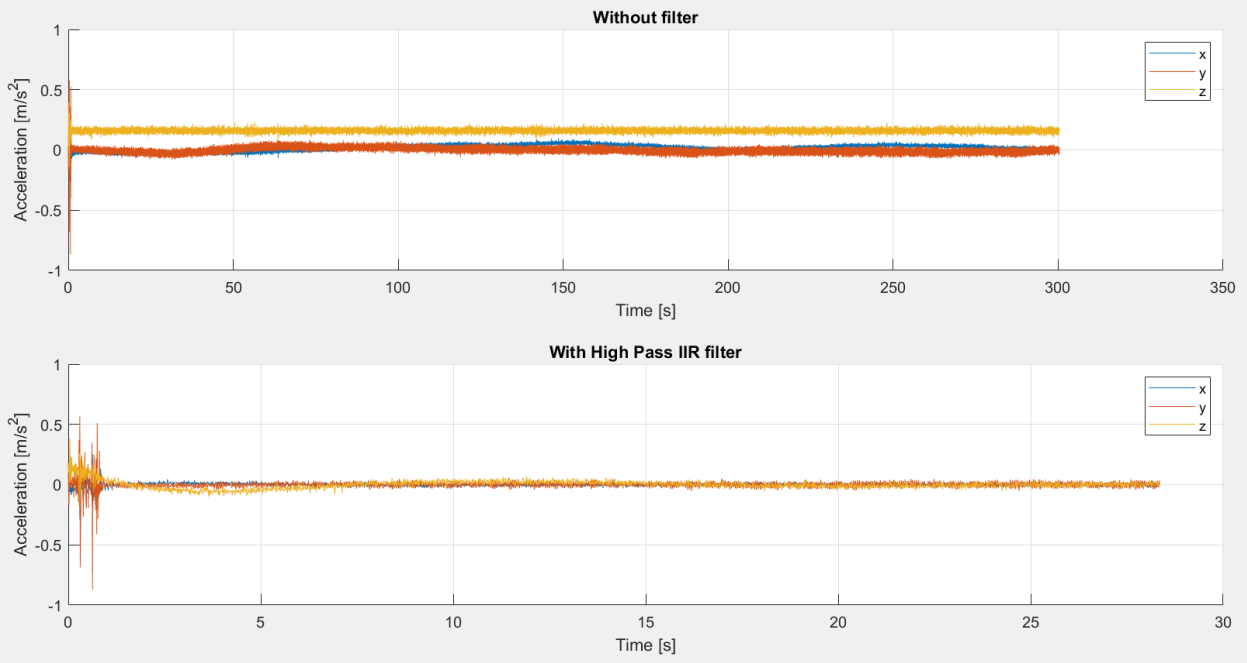
```
arm_biquad_cascade_df1_init_f32(&S,NUM_SECTIONS,pCoeffs,pState);
```

Call the function every time you want to filter new data. The filtered data is put in `&acc[0]`.

```
arm_biquad_cascade_df1_f32(&S,&acc[0],&acc[0],1);
```

And that's it! We never got it working as well as we wanted to and the acceleration data didn't seem to be that important after all, so this is as far as we got. At least, it gives an option to use a filter in the robot code. It is also possible to use other types of filters, you can find an overview of the available functions here.

The effect of the IIR filter in MATLAB, this is not the data of the applied filter on the robot.


Without filter
Acceleration [m/s^2]
Time [s]
x
y
z
With High Pass IIR filter
Acceleration [m/s^2]
Time [s]
x
y
z