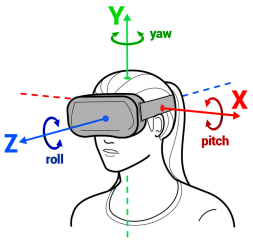
Extended Kalman Filtering

Andre Cornman, Darren Mei
Stanford EE 267, Virtual Reality, Course Report, Instructors: Gordon Wetzstein and Robert Konrad



Abstract

When working with virtual reality, one of the major issues is user immersion. Without an immersive experience, virtual reality quickly loses its appeal. One of the main factors in creating a successful experience is the integration of the Inertial Measurement Unit, or IMU.

This paper focuses on optimizing the integration of the IMU through Extended Kalman Filtering. Using the processes defined in previous research on Kalman Filtering, the method was implemented on MATLAB and compared with the Complementary Filter method. The constants within the Kalman Filter were optimized to best correct for sensor noise from the IMU.

Besides optimizing the Kalman Filter, this process was also implemented using the Teensy USB Development Board. Building off of previous work done utilizing the Arduino Microprocessor, the Teensy USB allows for users to experience the differences between the Complementary Filter and the Extended Kalman Filter.

Keywords: virtual reality, IMU, Extended Kalman Filtering, complementary filter

Concepts: Filtering, data analysis

1 Introduction

Head orientation tracking is an important aspect of HMD virtual reality because it allows the user to feel immersed in the environment and look around in a natural way. Orientation tracking can be performed using Euler angles, by tracking the pitch, yaw, and roll angles of the HMD, or by tracking quaternions. To track these angles, an IMU (inertial measurement unit) can be embedded into the HMD and read out gyroscope angular velocity, and linear acceleration from the accelerometer. Both of these readings can be used to predict the orientation of the user's head, but by fusing these measurements, we can more accurately predict the angles. The gyroscope is often accurate for orientation tracking in the short run, but will also drift over time due to integration error. The accelerometer is accurate over longer periods of time because it will not drift, but is also noisy and is affected by motion. To account for this, complementary filters are often used to fuse the gyroscope and accelerometer data. The complementary filter is a linear interpolation between the angle predicted by the gyroscope and the accelerometer. In effect, this acts as a low pass filter for the accelerometer, and a high pass filter for the gyroscope. An alternative approach to the IMU sensor fusion is Extended Kalman Filtering. This technique is an algorithm which estimates the state of the system and the variance or uncertainty of the estimate. The estimate is updated using a state transition model and measurements from the IMU. The Kalman filter is a two-step process. First, the prediction step produces estimates of the state variables, and their uncertainties. The update step then uses the next observed measurement to update the state variables using a weighted average, where measurements with more uncertainty are weighted less.

This paper compares the complementary filter to the Extended Kalman filter, specifically for use in orientation tracking with 6-DOF sensor fusion from gyroscope and accelerometer values. We compare these two filters based on the accuracy of the result compared to the true rotation, as well as noise reduction of the filter and latency caused by the computation. We first compare the filters using a MATLAB simulation, then we implement the filters using an Arduino and real measurements from the IMU.

2 Related Work

The complementary filter is widely used in the hobbyist community for its ease of implementation and use. It works generally well at tracking head angles accurately and reducing noise from the accelerometer. Profession head mounted displays like the Oculus Rift use Kalman filtering implemented using quaternions instead of Euler angles. While implementing orientation tracking using quaternions provides advantages like avoiding gimbal lock, it is significantly more complex than the Kalman filter using Euler angles. Thus we decided to compare the complementary filter with the Kalman filter only using Euler angles. For additional details on the quaternion Kalman filter, see "A Quaternion-based Unscented Kalman Filter for Orientation Tracking" by Edgar Kraft.

3 Our Approach

To compare the Extended Kalman Filter to the complementary filter, we first implemented both algorithms in a MATLAB simulation. We started by defining a true-rotation for the yaw, pitch, and roll angles for a ten second period. Then we calculated the expected gyroscope and accelerometer measurements for every millisecond from the true-rotation, resulting in a six-

element measurement model with 10,000 exact measurements. We then added random, normally distributed zero mean noise to both the gyroscope and accelerometer measurements. From these measurements, we computed the resulting predicted angles for both the complementary filter and the Kalman filter, and compared them based on accuracy and noise reduction.

As a second step, we implemented both filters on a Teensy 3.2 with real accelerometer and gyroscope data, and compared their outputs to the accelerometer measurements. We also measured the amount of latency accumulated by the computation required for both filters.

To implement the Kalman filter, we used the following equations for the prediction step.

$$x_{t|t-1} = A x_{t-1|t-1}$$

$$P_{t|t-1} = A P_{t-1|t-1} A^T + Q_t$$

Our state vector x was composed of the three Euler angles, and three angular velocity terms. Our predict step assumed constant velocity, such that the A matrix added the constant velocity to the next time step. We used a diagonal covariance matrix Q where the first three terms were the time step squared, and the next three terms were 0.01*time-step. See our code for additional details.

The update step followed the following equations:

$$y_t = z_t - H\left(x_{t|t-1}\right)$$
$$S_t = J_H^{(x_{t|t-1})} P_{t|t-1} J_H^{(x_{t|t-1})^T} + R_t$$
$$K_t = P_{t|t-1} J_H^{(x_{t|t-1})^T} S_t^{-1}$$
$$x_{t|t} = x_{t|t-1} + K_t y_t$$
$$P_{t|t} = \left(I - K_t J_H^{(x_{t|t-1})}\right) P_{t|t-1}$$

To determine the error covariance matrix R for the MATLAB simulation, we used a diagonal matrix where the first three values are simply the variance of the noise applied to the gyroscope measurements, and the next three values are the variance of the noise applied to the accelerometer measurements.

To find the values of the matrix R for the Arduino implementation, we estimated the variance of the measurement for both the accelerometer and gyroscope by keeping the IMU immobile and collecting many samples. From these samples, we calculated the variances for all three values from the gyroscope and the accelerometer. Then we used these six values as the diagonal of the R matrix. Again, see code for more detail.

4 Evaluation

We tested a few different cases in the MATLAB simulation.
We tested response of the filters to different true-rotations, as well as different noise levels for both the accelerometer and gyroscope.

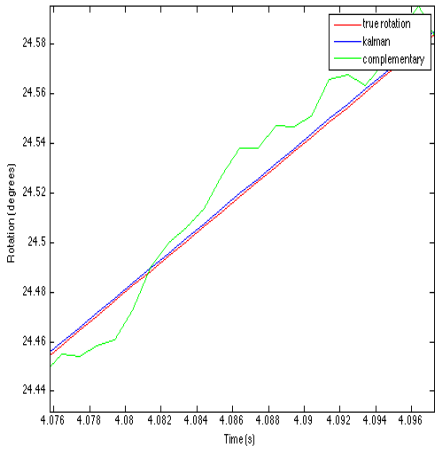


Figure 1: Comparison of Kalman and complementary filter roll angle output for linear true rotation.

In the linear test case, we added random normally distributed noise, with the accelerometer noise of variance 0.001 and gyroscope noise variance of 10^{-6}. In this case, the Extended Kalman filter had a lower RMS error, of 0.0069, compared to the complementary filter error of 0.0165. Additionally, the signal to noise ratio was greater for the Kalman filter, with 73.97 SNR, compared to 66.4 SNR for the complementary filter.

We also tested the filters in non-linear cases. We used the hamming function to model non-linear rotation, and took the difference of the discrete values to find the approximate derivative used as gyroscope measurements.

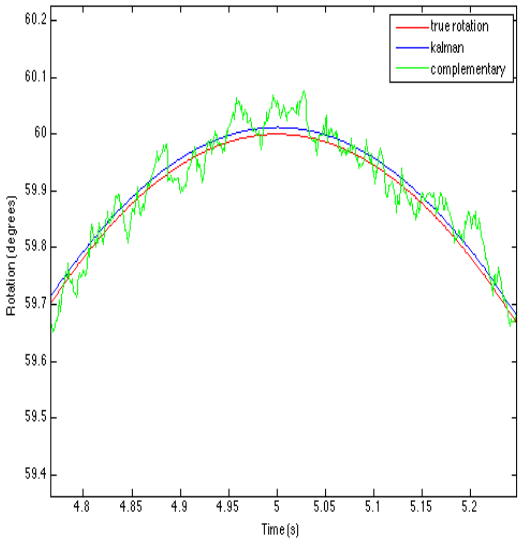


Figure 2: Comparison of Kalman and complementary filter roll

angle output for curved true rotation.

In this non-linear case, we added random normally distributed noise, with the accelerometer noise of variance 0.001 and gyroscope noise variance of 10^{-4}. Similar to the linear case, the Extended Kalman filter had a lower RMS error, of 0.0119, compared to the complementary filter error of 0.0512. Note that both filters incurred more error in the curved test compared to the linear test. The signal to noise ratio was greater for the Kalman filter, with 70.07 SNR, compared to 57.38 SNR for the complementary filter.

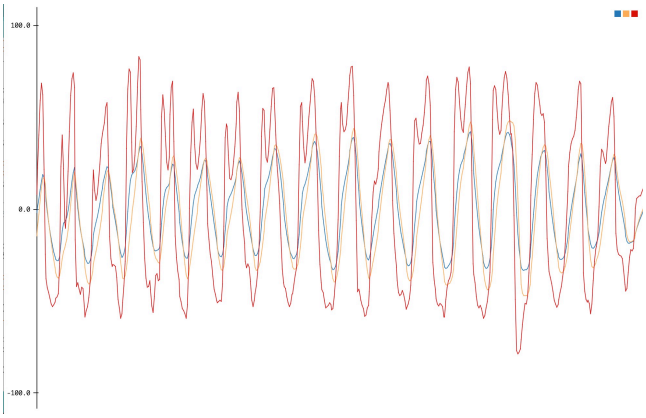


Figure 4: Pitch Comparison with an α-value of 0.9. Blue, yellow, and red lines correspond to Complementary Filter, Kalman Filter, and Accelerometer outputs, respectively.

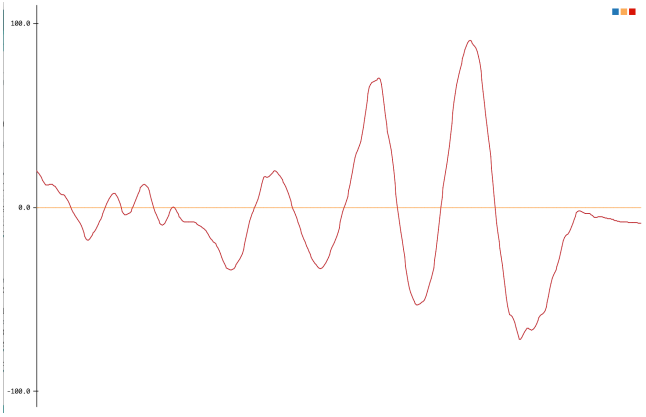


Figure 5: Yaw Comparison with an α-value of 0.9. Blue, yellow, and red lines correspond to Complementary Filter, Kalman Filter, and Accelerometer outputs, respectively.

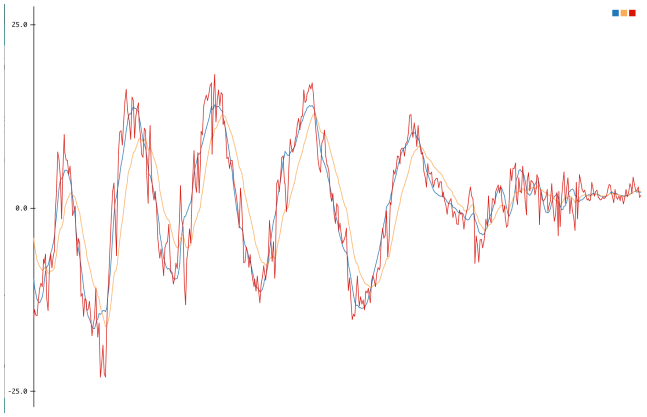


Figure 6: Roll Comparison with an α-value of 0.9. Blue, yellow, and red lines correspond to Complementary Filter, Kalman Filter, and Accelerometer outputs, respectively.

The implementation displayed both the pros and cons of the Extended Kalman Filter. As seen in Figures 4 and 6, the output of the Extended Kalman Filter is smoother than the output of the Complementary Filter. Both filters follow the general trend of the Accelerometer data, but the Complementary Filter tends to reflect more of the noise from the accelerometer data. For the Yaw data, as seen in Figure 5, the Extended Kalman Filter outputs 0. This is because the yaw rotation cannot be calculated from the accelerometer, thus the Kalman filter cannot perform the sensor fusion. For implementation purposes, the gyroscope integration for yaw is used instead. These graphs were generated with an α-value of 0.9.

The reason an α-value of 0.9 is used is because it is the value that optimizes the complementary filter. Since the α-value determines the weight of the accelerometer's contribution, having a higher α-value reduces the amount of noise from the accelerometer. This can be seen in Figure 7, which compares the pitch outputs when the α-value is 0.1.

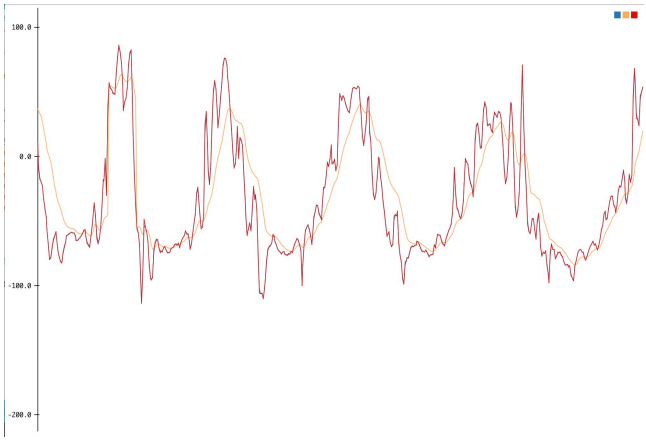


Figure 7: Pitch Comparison with an α-value of 0.1. Blue, yellow, and red lines correspond to Complementary Filter, Kalman Filter, and Accelerometer outputs, respectively.

It is difficult to discern the Complementary Filter output from the Accelerometer output, because they are so close together. By having the α-value closer to 1, it is easier for the differences between the Kalman Filter and Complementary Filter to be evaluated.

However, while the Extended Kalman Filter is smoother than the Complementary Filter, it does come with a larger latency. When running the Extended Kalman Filter 1000 times, an average loop time of approximately 9.26 milliseconds was observed. This is 2.87 times greater than the average loop time of the Complementary Filter, which was 3.23 milliseconds. The longer loop time is due to the increased computational complexity of the Extended Kalman Filter, and the required matrix inversion for each iteration of data.

5 Discussion

From the data observed, it appears that, while the Extended Kalman Filter offers greater noise reduction than the Complementary Filter, it has a much longer loop time. With the Inertial Measurement Unit, having an increased latency seriously impacts the user experience, and this issue needs to be further researched or resolved for the Kalman Filter to be viable, unless a faster processor is used.

During the Arduino implementation phase of the project, numerous issues arose. Many of these came with using the Teensy USB Development Board. For example, there was an issue connecting the Serial output of the Teensy Board to Unity, and keeping the data transfer in real-time. When implementing the original starter code for the quaternion Complementary Filter, the Unity program experienced much more lag than it did with the Arduino board. This was not lag due to the Kalman Filter, and instead suggested that there was a problem with the Teensy's Serial port. However, when using the Serial Plotter in the Arduino platform, there did not appear to be any latency issues. More research into the latency issues of the Teensy USB Development Board needs to be done before it will work with this implementation.

Another issue that was encountered in the Serial Plotter was that there was 180 degree offsets to the data output of the Extended Kalman Filter. The Extended Kalman Filter, at least in certain instances of measuring the roll, was 180 degrees higher or lower than the Accelerometer and Complementary Filter outputs. It is unclear what caused this offset, and it was manually corrected for in the program to make the outputs consistent around the 0 degrees.

However, in addition to this 180 degree offset, there also was a small DC offset between the Extended Kalman Filter and 0 degrees at certain points. When rapidly altering the values for the pitch and roll, this effect did not occur, but at very small changes, there was an approximately 10 degree offset for the Extended Kalman Filter. This suggests that there is still some bias within the Extended Kalman Filter, which can be corrected for. However, regardless of the offset, the Extended Kalman Filter did follow the general trend of the Accelerometer output.

From here, additional research can be done to more comprehensively determine the Extended Kalman Filter's effectiveness, and improve the general noise reduction of the IMU sensor. Since the Kalman does not return a value for the yaw, a low pass filter could be applied to the gyroscope readings to reduce noise. This approach could also be used in addition to the complementary filter, because it would reduce slightly more noise while still not significantly increasing the loop time.

Besides working on noise reduction with the complementary filter, more work could be done on trying to reduce the loop time for the Kalman Filter. Alternate implementations can be used to optimize run time. A potential method to do so would be running a Kalman Filter on each axis individually. This would reduce the size of each matrix, effectively cutting down on the run time for each axis.

References

S. Thrun, W. Burgard, D. Fox "Probabilistic Robotics", MIT Press 2005, chapter 3

G. Welch, G. Bishop "An Introduction to the Kalman Filter", UNC Tech. Report TR 95-041, 2006

St-Pierre, M., and D. Gingras. "Comparison between the Unscented Kalman Filter and the Extended Kalman Filter for the Position Estimation Module of an Integrated Navigation Information System." IEEE Intelligent Vehicles Symposium, 2004 (n.d.): n. pag. Web.

<https://github.com/TKJElectronics/KalmanFilter/blob/master/Kalman.cpp>

"The Extended Kalman Filter: An Interactive Tutorial." N.p., n.d. Web. 03 June 2016. <http://home.wlu.edu/~levys/kalman_tutorial/>.

Lauszus, Kristian. "TKJ Electronics." *» A Practical Approach to Kalman Filter and How to Implement It*. N.p., n.d. Web. 03 June 2016. <http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/>.

Magnusson, Thom. "State Estimation of UAV Using Extended Kalman" (n.d.): n. pag. Web. <http://liu.diva-portal.org/smash/get/diva2:627864/FULLTEXT01.pdf>.

Pedley, Mark. "Tilt Sensing Using a Three-Axis Accelerometer." *Freescale Semiconductor* (n.d.): n. pag. 6 May 2013. Web. <http://www.nxp.com/files/sensors/doc/app_note/AN3461.pdf>.

"Sensor Fusion Using the Kalman Filter." *Technical University Munchen*. N.p., n.d. Web. 03 June 2016. <http://campar.in.tum.de/Chair/KalmanFilter>.