Robotics Math 16-811
Sandra Mau
December 2005

What is the Kalman Filter and How can it be used for Data Fusion?

Motivation

Just to explain a little about the motivation for this topic, the project I was working on was called "PROSPECT: Wide Area Prospecting Using Supervised Autonomous Robots." Our goal was to develop a semi-autonomous mutli-robot supervision architecture. One of the major issues for rover navigation is knowing attitude and position. Our initial idea was to use a combination of IMU, GPS and odometry (encoders and visual) to solve this issue. So for my math project, I wanted to explore using the Kalman Filter for attitude tracking using IMU and odometry data.

As an additional follow up note, our NASA sponsored project was one of the many projects cancelled following NASA's change in vision to focusing on methods for a lunar return. Thus I unfortunately did not spend too much time developing this KF further, though hopefully it will become applicable in some future project.

Overview of the Kalman Filter

The Kalman filter can be summed up as an optimal recursive computation of the least-squares algorithm. It is a subset of a Bayes Filter where the assumptions of a Gaussian distribution and that the current state is linearly dependant on the previous state are imposed. In other words, the Kalman filter is essentially a set of mathematical equations that implement a predictor-corrector type estimator that is optimal in the sense that it minimizes the estimated error covariance when the condition of a linear-Gaussian system is met. If the Gaussian assumption is relaxed, the Kalman filter is still the best (minimum error variance) filter out of the class of linear unbiased filters. [8]

From the time it was developed, the Kalman has been the subject of extensive research and application, especially in the field of navigation. [4] The simple and robust nature of this recursive algorithm has made it particularly appealing. Also, even though it is rare that the optimal conditions exist in real-life applications, the filter often works quite well in spite of this and thus contributes to its appeal. [4]

History

The Kalman filter was created by Rudolf E. Kalman in 1960, though Peter Swerling actually developed a similar algorithm earlier. The first papers describing it were papers by Swerling (1958), Kalman (1960) and Kalman and Bucy (1961). [1] It was developed as a recursive solution to the discrete-data linear filtering problem.

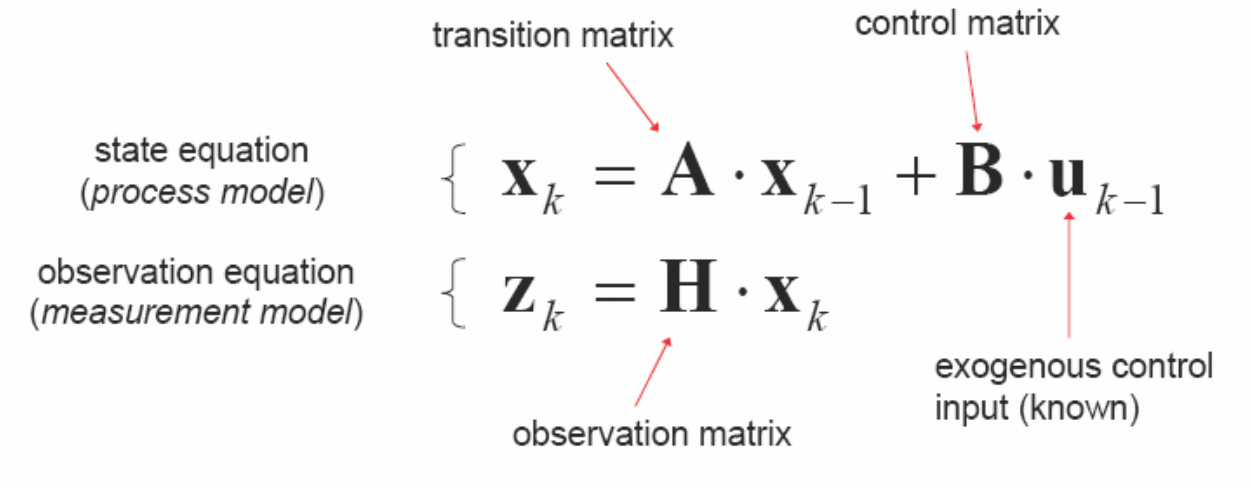
Stanley Schmidt is generally credited with the first implementation of a Kalman filter. In 1959 when NASA was to exploring the problem of navigating to the moon with the Apollo program, Schmidt at NASA Ames Research Center saw the potential of

extending the linear Kalman filter to solve the problem of trajectory estimation. The result was called the Kalman-Schmidt filter (now commonly known as the Extended Kalman Filter.) By 1961, Schmidt and John White had demonstrated that this filter, combined with optical measurements of the stars and data about the motion of the spacecraft, could provide the accuracy needed for a successful insertion into orbit around the Moon. The Kalman-Schmidt filter was embedded in the Apollo navigation computer and ultimately into all air navigation systems, and laid the foundation for Ames' future leadership in flight and air traffic research. [6]

Discrete Kalman Filter Algorithm

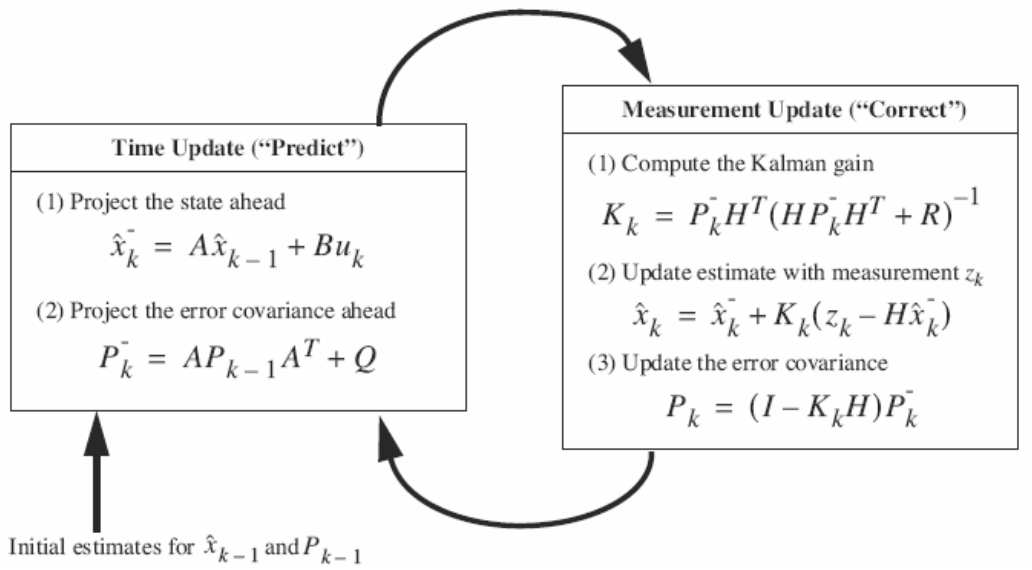
This section describes the filter in its original formulation (Kalman 1960) where the measurements are measured and state estimated at discrete points in time. The mathematical derivation will not be given here but can be looked up in various references [4][8]

Dynamic systems (this is, systems which vary with time) are often represented in a state-space model. State-space is a mathematical representation of a physical system as a set of inputs, outputs and state variables related linearly. The Kalman filter addresses the general problem of trying to estimate the state of a discrete-time controlled process that is governed by the models below [5]:



In addition to this, a random, Gaussian, white noise source $\mathbf{w_k}$ and $\mathbf{v_k}$ are added to the state and observation equations respectively. The process noise $\mathbf{w}$ has a covariance matrix Q and the measurement noise $\mathbf{v}$ has a covariance matrix R, which characterizes the uncertainties in the states and correlations within it.

At each time step, the algorithm propagates both a state estimate $\mathbf{x_k}$ and an estimate for the error covariance $\mathbf{P_k}$, also known as a priori estimates. The latter provides an indication of the uncertainty associated with the current state estimate. These are the predictor equations. The measurement update equations (corrector equations) provide a feedback by incorporating a new measurement value into the a priori estimate to get an improved a posteriori estimate. The equations for this recursive algorithm are shown below:

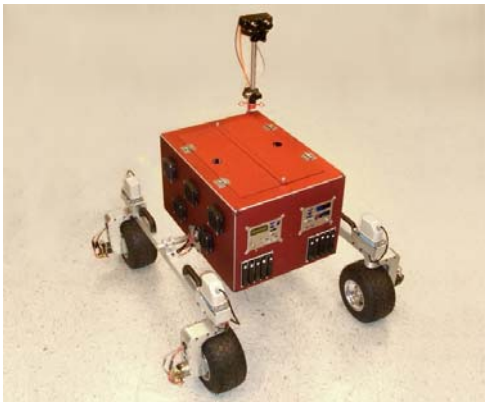


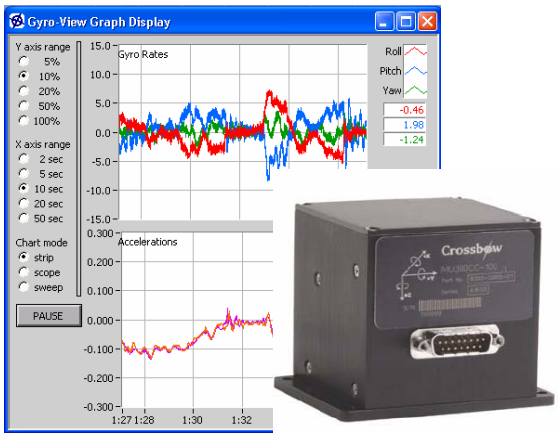
The Kalman Gain (**K** in measurement update equation 1) is derived from minimizing the a posteriori error covariance [4][5]. What the Kalman Gain tells us is that as the measurement error covariance **R** approaches zero, the actual measurement **z** is "trusted" more and more, while the predicted measurement $H\hat{x}$ is trusted less and less. On the other hand, as the a priori estimate error covariance **P** approaches zero the actual measurement is trusted less and less, while the predicted measurement is trusted more and more.

Application to research

Apparatus

For my experiments, the main sensor readings came from a Crossbow IMU 300CC, 6-DOF inertia measurement unit. The three angular rate sensors are vibratory MEMS sensors that use the coriolis force for measurements and the three MEMS accelerometers are silicon devices that use differential capacitance for measurements. This IMU was mounted inside the NASA K-10 rover at the turning center of the machine. All measurements were made by polling the IMU through a RS-232 serial port. A Visual C++ .NET 2003 program was developed to collect data.





Basic Kalman Filter Without Input to Track Constant Value
(use basic KF for yaw rate of stationary rover from IMU measurements)

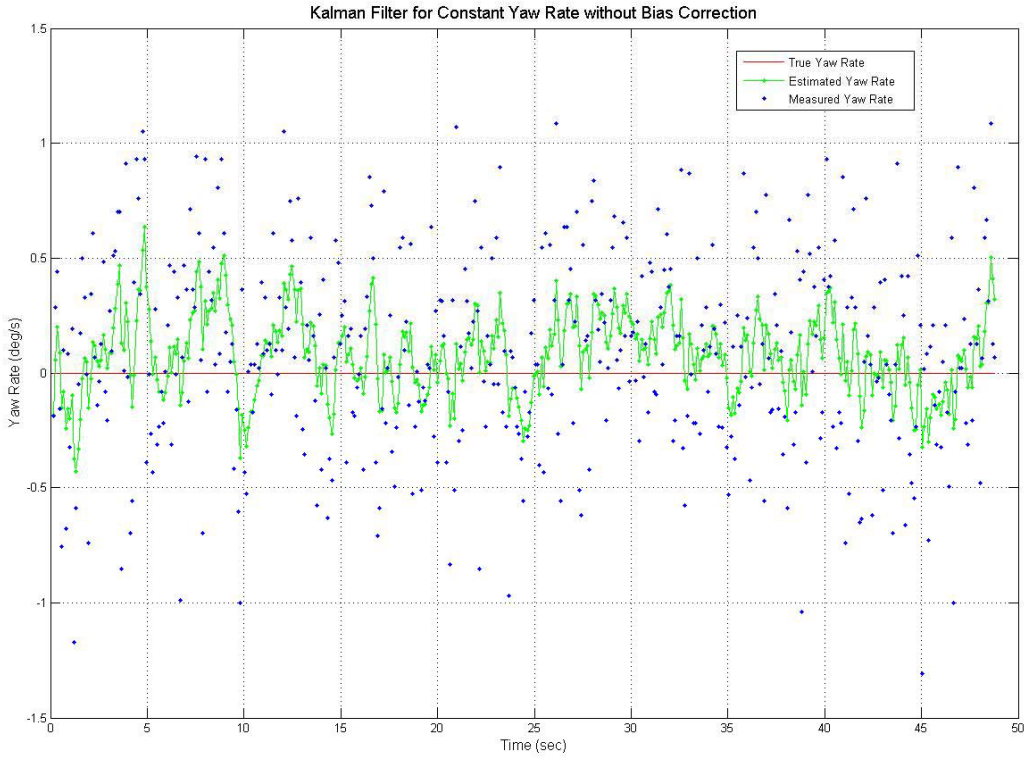
One of the classic basic Kalman Filter examples is to track a constant value with noisy measurements. So this too was my first step. I developed a basic Kalman Filter for tracking the constant yaw rate value of a stationary rover. The state model for this would be:

$$\hat{x}^{-}_{k+1} = \dot{\varphi}_{k+1}$$

$$\hat{z}_{k+1} = \dot{\varphi}_{k+1}$$

which means that coefficients A = 1, B = 0, H (or sometimes written as C) = 1.

Since the rover is stationary, the true raw rate should remain stationary at 0, but as can be seen in the figure below, the IMU gyro measurements are very noisy with bias error of up to +/- 2 deg/s (according to the manual). It can be seen that running the data through the Kalman Filter reduces the inaccuracies of the measurements quite dramatically, but is still not yielding the true heading exactly. Ofcourse, how well it tracks the constant depends a lot on the error covariances Q and R. The values I used were Q = 0.028^2 and R = 0.095^2 based on research from a paper using a similar Crossbow IMU [2].



Basic Kalman Filter Without Input to Track Yaw Angle (integrated yaw rate)

The next step was to integrate the yaw rate values to get heading (yaw angle.) From the yaw equation below, the state space model was developed. Note that B = 0 since there was no additional information input into the system.

Most Basic Version (just IMU yaw rate, no input u)

- Yaw equation: $\varphi_{k+1} = \varphi_k + (\dot{\varphi}_{k+1})(\Delta t_{k+1})$
- State space model:

$$\hat{x}^-_{k+1} = \begin{bmatrix} \varphi \\ \dot{\varphi} \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}_{k+1} \begin{bmatrix} \varphi \\ \dot{\varphi} \end{bmatrix}_k = A\hat{x}_k$$

$$\hat{z}_{k+1} = \begin{bmatrix} 1 & \Delta t \end{bmatrix}_{k+1} \hat{x}^-_{k+1} = H\hat{x}^-_{k+1}$$

Running this KF model on about 50 seconds of data from a stationary rover, it was found that the angle drifted off by nearly 2.2 degrees. After doing some research, I found that the major errors for these low-cost IMUs are bias and drift [7]. I also found other research that has tried to address this issue by estimating the bias to be able to compensate the measured value [3][9]. This lead to the next iteration described next.

Two-Tier Kalman for Yaw Angle with Yaw Rate Bias Correction

The method I used to estimate a bias value was to implement re-modify the basic KF for tracking a constant such that the a priori state estimate for every iteration was always zero. By doing this, it means that the deviation from zero in the estimate of our yaw rate constant results completely from the measurement and noise model (xhat = 0 + K * [z - 0] = K * z). Thus, this should give us an idea of how much the measurement is biased.

Add Yaw Rate Bias Correction (2-tier Kalman)

- Bias for stationary rover:

$$\hat{x}^-_{k+1} = \dot{\varphi}_{k+1} = 0$$

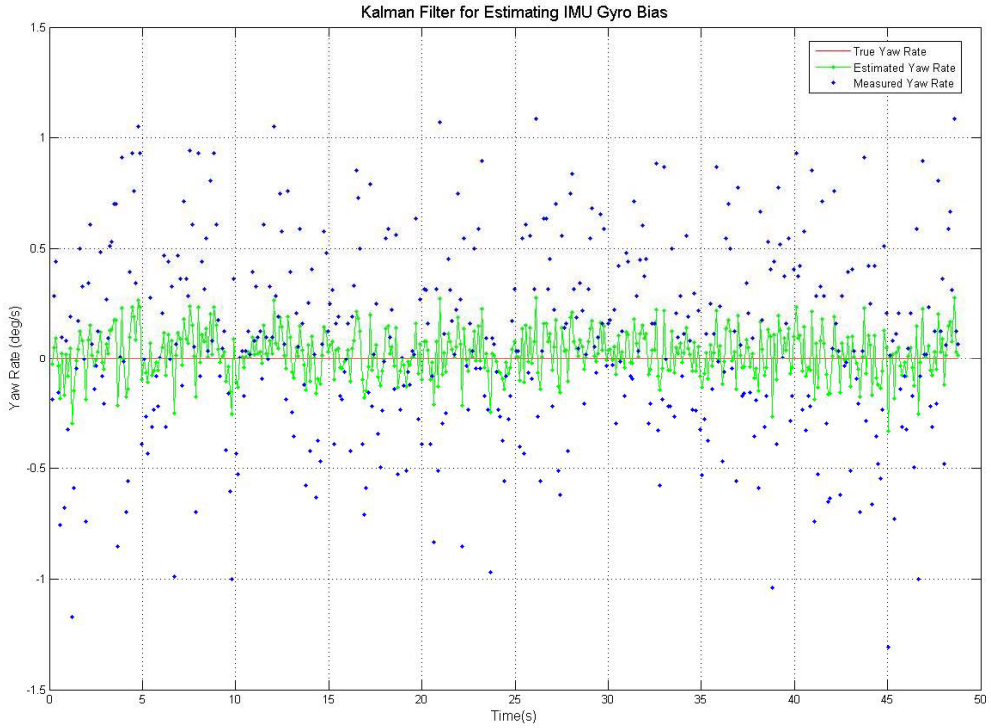
$$\hat{z}_{k+1} = [1]\dot{\varphi}_{k+1}$$

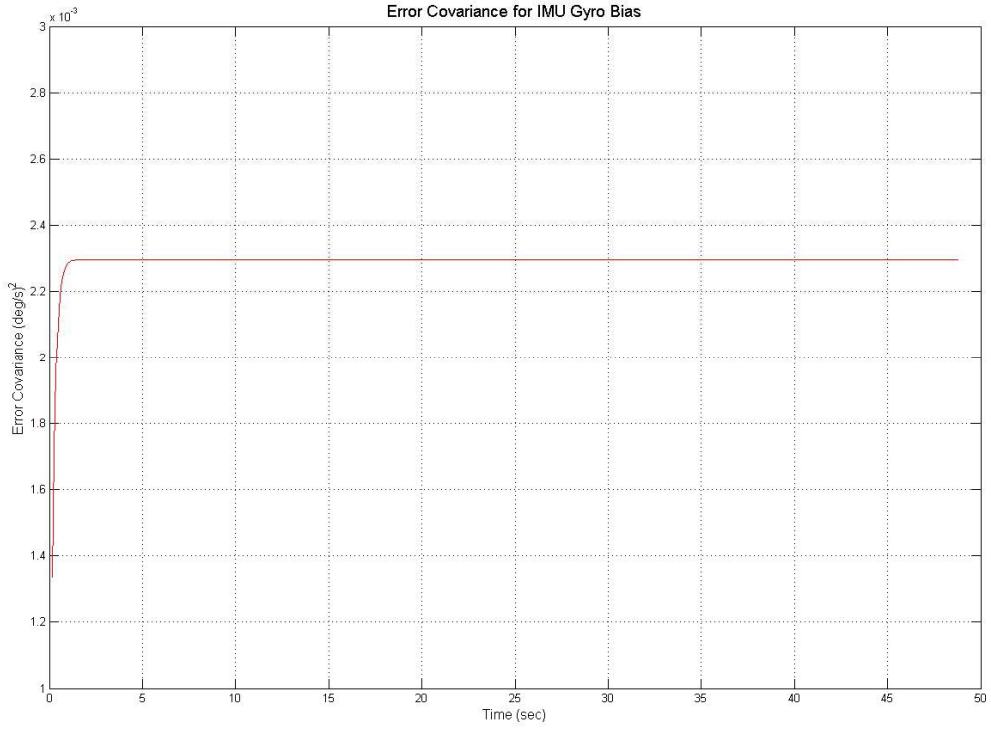
$$\Rightarrow \dot{\varphi}_{bias}$$

- New yaw equation: $\varphi_{k+1} = \varphi_k + (\dot{\varphi}_{k+1} - \dot{\varphi}_{bias})(\Delta t_{k+1})$

Below is the result of the yaw rate bias KF. Compared to the results of the basic yaw rate constant KF, it was observed that the estimate was better. This is because we

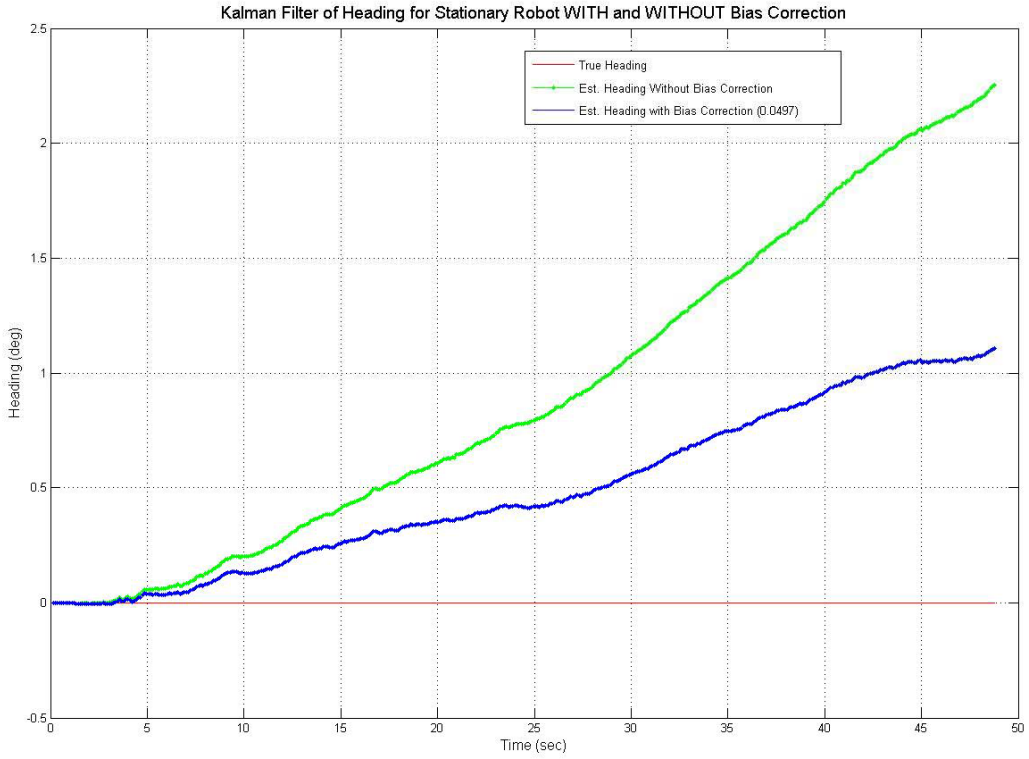
are incorporating other known data into the system (i.e. we know the heading should be zero, so we set our a priori estimate to zero based on that knowledge). This illustrates a form of data fusion.





After about 20 iterations, P converged to 0.0023 (see second image below). This means that the error covariance of the a posteriori state estimate $(x\text{-}xhat)^2$ was 0.0023

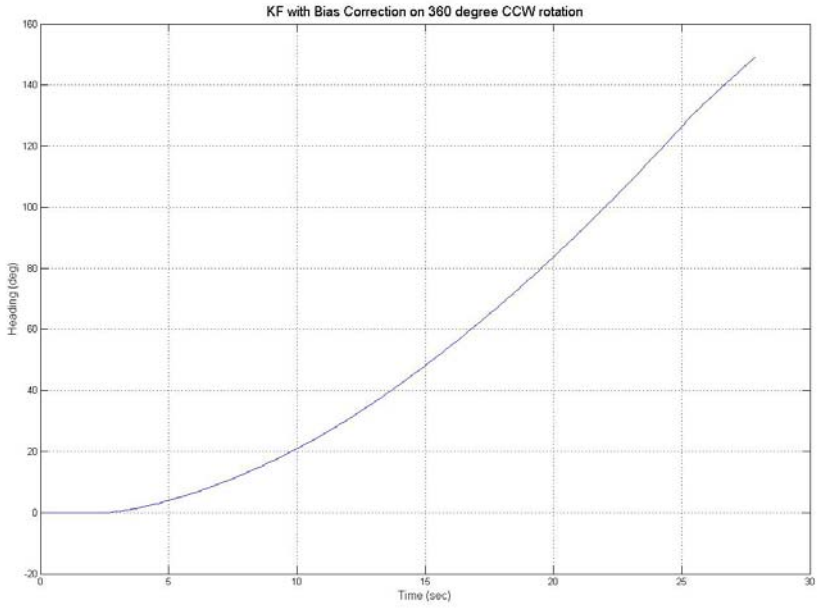
[deg/s]^2, thus the bias error in the state estimate was the square root of that, or 0.0479 deg/s. This value was then used in the basic KF for yaw angle by subtracting each measurement by this bias. The result was quite promising, giving only about 1 deg error after 50 seconds, which is over 100% improvement from KF without bias correction as can be seen graphically below.

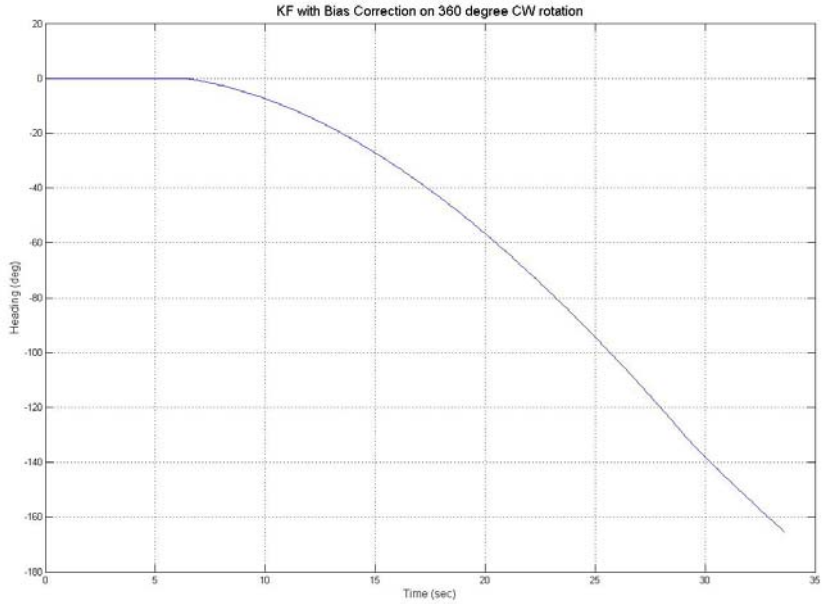


One thing of note is that the bias also drifts as time goes on, so any bias estimation would likely become invalid quickly. [3][10] Since bias has a quadratic effect on position derived from IMU [7], it can become a serious problem. One proposed solution was to stop the rover regularly to recalibrate the bias [3], another was to pre-filter the noisy IMU measurements [10] and a third was to get more data to obtain a better a priori estimate or measurement (a general solution).

Issues

The KF for yaw with bias correction developed above was applied to the rover turning 360 degrees both CW and CCW. It was found that the tracking was not very good in this instance at all. For both results, the KF yielded about 160 deg rather than the actual 360 deg as can be seen below.





The graph itself looks good at the beginning: a flat line indicating when the rover was still stationary, then a nearly linear slope indicating when the rover was turning at a nearly constant velocity. However, I noticed that there is no flat line at the end indicating when the rover stopped turning and became stationary again. This leads me to suspect that it might be that there is a significant communication delay between the time the IMU is polled for data and when the data is actually read by my code.

Another problem might be that the data is not being fast enough for a great estimate. Right now, the data is being received at 10 Hz. This may be increased by changing the IMU from polled to continuous mode.

Sensor Fusion

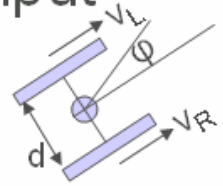
Kalman with Motion Control Input and IMU Measurement to Track Yaw Angle

As was briefly touched upon before, data or sensor fusion can be made through the KF by using various sources of data for both the state estimate and measurement update equations. By using these independent sources, the KF should be able to track the value better. The next iteration for my KF would be to incorporate the wheel velocities in making the state measurement as seen in the model below.

IMU measurement + motion control input

- Yaw equation:

$$\varphi_{k+1} = \varphi_k + \left(\frac{v_R - v_L}{d/2} \right)_{k+1}$$



- State space model:

$$\hat{x}^-_{k+1} = \begin{bmatrix} \varphi \\ \dot{\varphi} \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \varphi \\ \dot{\varphi} \end{bmatrix}_k + \begin{bmatrix} \frac{2}{d} & \frac{-2}{d} \end{bmatrix} \begin{bmatrix} v_R \\ v_L \end{bmatrix} = A\hat{x}_k + Bu_{k+1}$$

$$\hat{z}_{k+1} = \begin{bmatrix} 1 & \Delta t \end{bmatrix}_{k+1} \hat{x}^-_{k+1} = H\hat{x}^-_{k+1}$$

I unfortunately did not have the foresight to collect motor control information which can be converted to wheel velocities to run my code for the above model. I will leave this as future work.

Conclusion

The Kalman Filter is the most widely used filter for good reason. It is a very simple and intuitive concept with good computational efficiency. However, that being said, to implement it just right is somewhat of an art and very dependent on the system at hand.

There are also many variants for Kalman (and more generally, Bayesian) filters with their respective benefits and drawbacks. It is no wonder that the field is so well explored and yet still being developed further.

References

[1] "Kalman Filter". Wikipedia, the free encyclopedia. Available [Online] <http://en.wikipedia.org/wiki/Kalman_filter> Las modified: December 2005.

[2] Chris Hide, Terry Moore, Chris Hill, David Park. "Low cost, high accuracy positioning in urban environments." IESSG, University of Nottingham. UK. 2005.

[3] E. T. Baumgartner, H. Aghazarian, and A. Trebi-Ollennu. "Rover Localization Results for the FIDO Rover." Jet Propulsion laboratory, CalTech, Pasadena, CA.

[4] Gary Bishop and Greg Welch. "An Introduction to the Kalman Filter". Universityy of Norh Carolina SIGGRAPH 2001 course notes. ACM Inc., North Carolina, 2001.

[5] John Spletzer. "The Discrete Kalman Filter". Lecture notes CSC398/498. Lehigh university. Bethlehem, PA, USA. March 2005.

[6] Jonas Dino. "NASA Ames Research Center, Moffett Field, Calif., history related to the Apollo Moon Program and Lunar Prospector Mission." NASA News. Available [Online] <http://www.nasa.gov/centers/ames/news/releases/2004/moon/apollo_ames_atmos.html> Last modified: March 2005.

[7] Kevin J. Walchko. "Low Cost Inertial Navigation Learning to integrate Noise and Find Your Way." Master's Thesis. University of Florida. 2002.

[8] Rudolph van der Merwe, Alex T. Nelson, Eric Wan. "An Introduction to Kalman Filtering." OGI School of Science & Engineering lecture. Oregon Health & Sciences University. November 2004.

[9] Sanghyuk Park and Jonathan How. "Examples of Estimatiion Filters from Recent Aircraft Projects at MIT." November 2004.

[10] Srikanth Saripalli, Jonathan M. Roberts, Peter I. Corke, Gregg Buskey. "A Tale of Two Helicopters." Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 805-810. Las Vegas, USA, Oct 2003.