A Kalman Filter Approach to Motion Detection of a Mobile Device

Wagner L. Truppel

March 18, 2013

1 Kalman filters

A Kalman filter is a mathematical procedure used to smoothen noisy discrete data. Unlike other smoothing techniques, however, it takes into account the sources of errors that typically make discrete data noisy. As a result, the data is not only smoother but generally also more accurate than the measured data alone.

The essence of a Kalman filter is that we can use the history of the system up to and including the current state of the system to make a *prediction* of the next state (a process that is subjected to errors due to the discrete nature of the data) and then combine that prediction with a *measurement* of the next state (which is subjected to random noise), to obtain an estimate of that next state which is better than either the prediction or the measurement alone.

By themselves, our predictions based on the history of the system suffer from drifting and become more and more incorrect as we make more and more predictions, because of the accumulation of errors intrinsic to the process. On the other hand, measurements alone may be too noisy. However, by combining both pieces of information, we can correct our predictions based on the measurements and simultaneously mitigate the effects of random measurement noise by relying on the accumulated history of the system. The result is a smoother and more robust estimate of the next state of the system.

Roughly, the whole process amounts to taking a weighted average of the predicted state and the measured state of the system,

$$\text{better estimate} = (1 - K) \times (\text{prediction}) + K \times (\text{measurement}),$$

where K, known as the *Kalman Gain*, is (in the simplest case) a number between 0 and 1. It indicates how much we trust the measurement to be correct. The closer the Kalman gain is to 1, the more we trust the measurement. Conversely, the closer it is to 0, the more we trust our prediction rather than the measurement. In a way, then, a Kalman filter has some 'memory' of what data it's seen before and that's what allows it to be resilient against random noise and other random forms of error that affect the data.

Other methods of smoothing discrete noisy data also use a weighted average and, therefore, also display a form of 'memory,' but the averaging process takes place only between *measurements*, as in

$$\begin{aligned}\text{smoothed current measurement} &= (1-K) \times \text{(previous smoothed measurement)} \\ &+ \quad K \times \text{(current [noisy] measurement).}\end{aligned}$$

What makes a Kalman filter particularly effective compared to other approaches is that it incorporates predictions *and* automatically finds the best value of K to use at every step. A Kalman filter not only 'remembers' what it's seen before but it also 'learns' from it.

2 Detecting Motion

The problem of detecting the motion of a mobile device (or of anything else, really) boils down to estimating the device's *speed* relative to a fixed reference frame (say, the ground). If that speed is nearly zero, the device can be considered not to be moving. What makes this a complicated problem for mobile devices is that the current GPS hardware is not of sufficient quality for the task. Even if it was, this would still be a hard problem to solve in practice because GPS signals bounce around tall buildings and other structures, changing their time of transit, and, in any case, do not penetrate structures such as long tunnels. In other words, the GPS data available to the device is noisy, often presents large inaccuracies, and sometimes isn't available at all.

In the absence of a simple solution based on reliable and accurate measurements from a single hardware component, we can use the device's built-in accelerometer to measure the device's acceleration and then integrate that stream of values to obtain the device's velocity. We still need the GPS for an initial estimate of the device's velocity and for periodic corrections, and we also need the magnetometer and gyroscope to determine the relative orientation of the device with respect to the reference frame used by the GPS. It's only then that we can use a Kalman filter to predict the device's speed (using the measured acceleration) and compare it with the speed measured by the GPS.

The accelerometer data, in each of its three directions, includes three components: gravity, the acceleration imparted by the user, and random noise. What we're interested in is the acceleration imparted by the user. Luckily, the device's operating system does the hard work for us and subtracts the contribution due to gravity.

Once we have the device's user-imparted acceleration, which is measured relative to the device's own reference frame, we need to transform it to a reference frame with axes pointing straight up (towards the local Zenith), towards East, and towards the geographical North ('true' North, as opposed to Magnetic North), respectively, as this is the reference frame the GPS speed is reported relative to. Yet again the device's operating system comes to the rescue since there exist API calls to make the necessary transformation.

The bottom line at this stage is that, at discrete moments in time, we have the user acceleration along the North and East directions, which we can use to predict the device's velocity vector parallel to the ground. We also have access to measurements of this velocity, from the GPS. In other words, we have all that we need to use a Kalman filter and improve our estimates of that velocity vector and, in particular, of its magnitude (speed).

Unfortunately, there are two wrinkles in this process. The first is that there are two potentially very different time scales involved, namely, the time intervals at which we obtain accelerometer and GPS speed measurements. It is reasonable to measure the device's acceleration, say, ten times a second (that's an adjustable parameter of the implementation). Doing it less often makes our predictions too noisy but doing it much more frequently won't make them that much more reliable; rather, it will just increase the burden on the hardware, especially on the device's battery. On the other hand, we don't get speed measurements nearly as often. In fact, we only get speed updates when we have location updates, that is, when the device's operating system decides that the device has moved a sufficiently large distance, based on information from the GPS. As a result, it's probably a good idea to have two separate Kalman filters, one to smoothen the acceleration and a second one to smoothen the velocity. The second filter uses the results of the first but doesn't try to model changes in both quantities at the same time, as a single-filter implementation would. It would be considerably more difficult to get a single-filter implementation to behave accurately because of the complexities inherent to dealing with a five-dimensional state space (three components of acceleration and two components of the velocity parallel to the ground), rather than one three-dimensional filter and a separate two-dimensional filter, but also because the two disparate time scales would 'confuse' the single-filter implementation.

The second wrinkle is that the Kalman filter 'remembers' the *entire* history of the system, which may be a problem in our case. An initial implementation suggested that the filter tends to give too much weight to the system history in the presence of noisy measurements. As a result, it's too slow to 'catch up' with changes in velocity. When the device stops moving, the filter still thinks it's moving and will take a while to output a velocity that is nearly zero. The solution to this problem is easy in concept but requires a careful implementation: we make the Kalman filter remember only the last few seconds (an adjustable parameter) worth of system history.

3 The mathematics of a Kalman filter

First, a few definitions. In what follows, a *bar* ($\bar{}$) over a quantity means that the quantity in question is *predicted*, a *tilde* ($\tilde{}$) means that it is *measured*, and a *hat* ($\hat{}$) means it's our *best estimate* at the time. The system of interest has a *state space* described by some collection of quantities. These are organized in what's called the *state vector* $\mathbf{q}$. In our acceleration filter, $\mathbf{q}$ would correspond to the acceleration vector (a three-dimensional vector) and in our velocity filter, $\mathbf{q}$ would correspond to the velocity vector parallel to the ground (a two-dimensional vector).

Let $\hat{\mathbf{q}}_{k-1|k-1}$ be our best estimate of the state of the system at time t_{k-1}, taking into account all the information available at that time (that is, both the history of the system up to that time and a measurement of the state vector at that time). This is called the *a-posteriori* estimate.

3.1 Prediction Step

Now imagine that we use that information to make a *prediction*, $\bar{\mathbf{q}}_{k|k-1}$, of the state of the system at time t_k, but without the benefit of a measurement to correct it. This is called the *a-priori* estimate, because it relies only on the history of the system up to that time. In the linear version of the Kalman filter (the only one we'll consider here), we expect this estimate to be some linear combination of the a-posteriori estimate of the state of the system in the previous time-step, plus a contribution from some external influences, if they exist (known as *control factors*):

$$\bar{\mathbf{q}}_{k|k-1} = \mathbf{F}_k \, \hat{\mathbf{q}}_{k-1|k-1} + \mathbf{B}_k \, \mathbf{u}_{k-1} \, ,$$

where $\mathbf{F}_k$ is the transition matrix, $\mathbf{u}_{k-1}$ the control vector, and $\mathbf{B}_k$ is the control matrix. $\mathbf{F}_k$ and $\mathbf{B}_k$ typically depend on both k and $k-1$.

Now, our best estimate at the previous time-step isn't perfect, that is, it is subjected to some error. We'll also be estimating and correcting that error as we go along, so if $\hat{\mathbf{P}}_{k-1|k-1}$ is our best estimate of the covariance matrix at time t_{k-1}, then what is our a-priori estimate of the same matrix at time t_k? Mr. Kalman has shown that the answer is

$$\bar{\mathbf{P}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{P}}_{k-1|k-1} \mathbf{F}_k^t + \bar{\mathbf{\Sigma}}_k \, ,$$

where $\bar{\mathbf{\Sigma}}_k$, known as the *process error covariance matrix*, is a measure of the error incurred when we make our a-priori prediction of the state of the system at time t_k.

3.2 Update Step

Once we have an a-priori estimate of the state of the system at time t_k, and of the error we're incurring on when making that estimate, we can update or correct both of them by means of a measurement of the actual state of the system at time t_k. Let $\tilde{\mathbf{q}}_k$ be the measurement of the state of the system at time t_k. In the linear version of the Kalman filter, that measurement is expected to be close to a linear combination of our a-priori estimate of the state of the system at that time, so we're interested in their difference, known as the *residual* or *innovation*:

$$\delta_k = \tilde{\mathbf{q}}_k - \mathbf{H}_k\,\bar{\mathbf{q}}_{k|k-1}\,.$$

The smaller this residual is (in some well-defined sense), the closer our prediction of the state of the system at time t_k is to the actual measurement of the state of the system at that time. However, the measurement itself has some error, typically just noise, described by the *measurement covariance matrix* $\tilde{\mathbf{\Sigma}}_k$, and it contributes to degrade our best estimate of the state of the system at time t_k. In other words, we also have a residual covariance, given by:

$$\mathbf{S}_k = \mathbf{H}_k\,\bar{\mathbf{P}}_{k|k-1}\,\mathbf{H}_k^t + \tilde{\mathbf{\Sigma}}_k\,.$$

If we insert into this expression the expression for $\bar{\mathbf{P}}_{k|k-1}$, we can gain a better appreciation for what is happening, error-wise:

$$\mathbf{S}_k \quad = \quad \mathbf{H}_k\,\mathbf{F}_k\hat{\mathbf{P}}_{k-1|k-1}\mathbf{F}_k^t\,\mathbf{H}_k^t \quad + \quad \mathbf{H}_k\,\bar{\mathbf{\Sigma}}_k\,\mathbf{H}_k^t \quad + \quad \tilde{\mathbf{\Sigma}}_k\,.$$

This expression tells us that the a-posteriori prediction at time t_k that we're about to make is degraded by three sources of errors: the a-posteriori error that we had already incurred on in the previous step, the process error inherent in making an a-priori estimate of the next state, and the error due to measuring the actual value of the next state of the system.

At every step, the Kalman filter attempts to minimize the estimated a-posteriori error for that step. It does so by constructing the so-called *optimal Kalman gain*, a matrix $\mathbf{K}_k$ defined by

$$\mathbf{K}_k = \bar{\mathbf{P}}_{k|k-1}\,\mathbf{H}_k^t\,\mathbf{S}_k^{-1}\,.$$

We use it to build our best estimates of the next state of the system as well as of the accumulated error we're incurring when making the state estimate:

$$\begin{aligned}\hat{\mathbf{q}}_{k|k} &= \bar{\mathbf{q}}_{k|k-1} + \mathbf{K}_k\,\delta_k = \mathbf{K}_k\,\tilde{\mathbf{q}}_k + (\mathbf{I} - \mathbf{K}_k)\,\bar{\mathbf{q}}_{k|k-1}\\ \hat{\mathbf{P}}_{k|k} &= (\mathbf{I} - \mathbf{K}_k\,\mathbf{H}_k)\,\bar{\mathbf{P}}_{k|k-1}\,.\end{aligned}$$

We now have adanced to the next step of the cycle and can repeat the entire procedure, thereby 'evolving' the system forward in time, constantly predicting and correcting our estimates based on the system history and on measurements made along the way.

3.3 Summary

Starting from the initial values $\hat{\mathbf{q}}_{0|0}$, $\hat{\mathbf{P}}_{0|0}$, $\bar{\mathbf{\Sigma}}_1$, and $\tilde{\mathbf{\Sigma}}_1$, we compute, in sequence:

$$\begin{aligned}
\bar{\mathbf{q}}_{k|k-1} &= \mathbf{F}_k\, \hat{\mathbf{q}}_{k-1|k-1} + \mathbf{B}_k\, \mathbf{u}_{k-1} \\
\bar{\mathbf{P}}_{k|k-1} &= \mathbf{F}_k \hat{\mathbf{P}}_{k-1|k-1} \mathbf{F}_k^t + \bar{\mathbf{\Sigma}}_k \\
\mathbf{S}_k &= \mathbf{H}_k\, \bar{\mathbf{P}}_{k|k-1}\, \mathbf{H}_k^t + \tilde{\mathbf{\Sigma}}_k \\
\mathbf{K}_k &= \bar{\mathbf{P}}_{k|k-1}\, \mathbf{H}_k^t\, \mathbf{S}_k^{-1} \\
\hat{\mathbf{q}}_{k|k} &= \mathbf{K}_k\, \tilde{\mathbf{q}}_k + (\mathbf{I} - \mathbf{K}_k)\, \bar{\mathbf{q}}_{k|k-1} \\
\hat{\mathbf{P}}_{k|k} &= (\mathbf{I} - \mathbf{K}_k\, \mathbf{H}_k)\, \bar{\mathbf{P}}_{k|k-1}\, .
\end{aligned}$$

For the application we have in mind, both the transition matrix $\mathbf{F}_k$ and the measuring matrix $\mathbf{H}_k$ are equal to the identity matrix at every step. This simplifies things a bit, leading to:

$$\begin{aligned}
\bar{\mathbf{q}}_{k|k-1} &= \hat{\mathbf{q}}_{k-1|k-1} + \mathbf{B}_k\, \mathbf{u}_{k-1} \\
\mathbf{S}_k &= \hat{\mathbf{P}}_{k-1|k-1} + \bar{\mathbf{\Sigma}}_k + \tilde{\mathbf{\Sigma}}_k \\
\mathbf{K}_k &= \mathbf{I} - \tilde{\mathbf{\Sigma}}_k\, \mathbf{S}_k^{-1} \\
\hat{\mathbf{q}}_{k|k} &= \mathbf{K}_k\, \tilde{\mathbf{q}}_k + (\mathbf{I} - \mathbf{K}_k)\, \bar{\mathbf{q}}_{k|k-1} \\
\hat{\mathbf{P}}_{k|k} &= \mathbf{K}_k\, \tilde{\mathbf{\Sigma}}_k\, .
\end{aligned}$$

3.4 Initial Values

Since, at the start, we don't have any system history to rely upon, our confidence in our predictions should be low. That can be accomplished by choosing a large value (in some sense) for $\hat{\mathbf{P}}_{0|0}$ since that makes the optimal Kalman gain very close to the identity matrix, thereby emphasizing the measured data rather than our predictions. It's customary to set $\hat{\mathbf{P}}_{0|0} = s\mathbf{I}$ where s is some large positive number and $\mathbf{I}$ is the identity matrix. It's in the nature of the Kalman filter to automatically adjust $\hat{\mathbf{P}}_{k|k}$ to more reasonable values as we gather more data.

$\bar{\mathbf{\Sigma}}_1$ is hard to select because we generally don't have enough information about the process error to begin with. $\tilde{\mathbf{\Sigma}}_1$, on the other hand, can often be estimated by the incoming data. We deal with estimating these matrices for any time-step in the sections to follow.

Finally, $\hat{\mathbf{q}}_{0|0}$ is easy to set but is application-dependent.

3.5 Stability Issues

Due to rounding errors, it's possible for the covariance matrices to become asymmetric, in which case instabilities could grow and potentially cause $\mathbf{S}$ to become singular. To protect the Kalman filter from having this problem we can first make sure that each covariant matrix is made symmetric by replacing it with half the sum of it and its transpose:

$$\mathbf{S} \to \frac{1}{2}\left(\mathbf{S} + \mathbf{S}^t\right).$$

Next, we can replace them by their $\mathbf{LDL}^t$ decompositions, which always exist for symmetric positive semi-definite matrices such as any covariance matrix. This decomposition lets us monitor signs of singularity by looking at the diagonal matrix in the middle. If any of its diagonal elements approaches zero, we can regularize it by adding to it a small positive number.

For 2-by-2 covariance matrices such as those that will arise in the Kalman filter for velocity that we'll be designing in the next sections, the $\mathbf{LDL}^t$ decomposition is particularly simple: $\mathbf{S} = \mathbf{LDL}^t$ where

$$\mathbf{L} = \begin{bmatrix} 1 & 0 \\ \ell_{21} & 1 \end{bmatrix} \qquad \text{and} \qquad \mathbf{D} = \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix},$$

with

$$d_1 = S_{11}\,, \qquad \ell_{21} = \frac{S_{12}}{d_1}\,, \qquad \text{and} \qquad d_2 = S_{22} - d_1\,\ell_{21}^2\,.$$

It then follows that the inverse is

$$\mathbf{S}^{-1} = (\mathbf{L}^{-1})^t\,\mathbf{D}^{-1}\,\mathbf{L}^{-1} = \begin{bmatrix} 1 & -\ell_{21} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1/d_1 & 0 \\ 0 & 1/d_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\ell_{21} & 1 \end{bmatrix}.$$

For the 3-by-3 covariance matrices in the Kalman filter for acceleration, we also have $\mathbf{S} = \mathbf{LDL}^t$ but, now,

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ \ell_{21} & 1 & 0 \\ \ell_{31} & \ell_{32} & 1 \end{bmatrix} \qquad \text{and} \qquad \mathbf{D} = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix},$$

with

$$\begin{aligned}
d_1 &= S_{11} \\
\ell_{21} &= S_{12}/d_1 \\
\ell_{31} &= S_{13}/d_1 \\
d_2 &= S_{22} - d_1\,\ell_{21}^2 \\
\ell_{32} &= (S_{23} - \ell_{21}\,\ell_{31}\,d_1)/d_2 \\
d_3 &= S_{33} - d_1\,\ell_{31}^2 - d_2\,\ell_{32}^2\,.
\end{aligned}$$

The inverse is then

$$\begin{aligned}
\mathbf{S}^{-1} &= (\mathbf{L}^{-1})^t\,\mathbf{D}^{-1}\,\mathbf{L}^{-1} \\
&= \begin{bmatrix} 1 & -\ell_{21} & \ell_{21}\,\ell_{32} - \ell_{31} \\ 0 & 1 & -\ell_{32} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/d_1 & 0 & 0 \\ 0 & 1/d_2 & 0 \\ 0 & 0 & 1/d_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -\ell_{21} & 1 & 0 \\ \ell_{21}\,\ell_{32} - \ell_{31} & -\ell_{32} & 1 \end{bmatrix}.
\end{aligned}$$

3.6 Estimating Confidence Intervals

At the end of each time-step, our best estimate of the state of the system is accompanied by an estimate of the covariance matrix associated with that state. We would like to use that estimated covariance matrix to obtain some kind of confidence interval on the best estimate of the state of the system. In particular, we'd like to obtain a confidence interval on the magnitude of the state vector at each step.

If the components of the state vector were independently distributed random variables with identical Gaussian distributions of zero mean and unit variance, then the magnitude of the state vector would be distributed according to a Rayleigh or Maxwell distribution (Rayleigh in 2 dimensions, Maxwell in 3). But they're not independently distributed, nor is their individual distributions necessarily a Gaussian (though that's a common assumption in Kalman filter implementations) and, even if they were, their means and variances aren't the same. Not sure how to proceed here. Need to think some more.

3.7 Limiting System History

As mentioned earlier, one of the issues we need to address is that the Kalman filter 'remembers' the *entire* past history of the system up to the time-step under consideration and that may not be a good thing because it tends to make the filter very 'stiff', in the sense that it will respond very slowly to changes in the data. To fix that we need to make the filter 'remember' only the most recent data points, say, the most recent n of them. The problem, though, is that until we actually have n data points, the filter must 'remember' the entire history of the system up to that point.

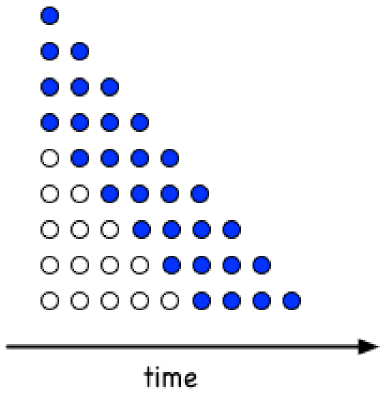
Now, we'll be estimating the $\bar{\mathbf{\Sigma}}_k$ and $\tilde{\mathbf{\Sigma}}_k$ covariance matrices using the sample variances of the various quantities of interest. As it turns out, it's possible to update these matrices in a running fashion, without having to physically store the entire history of the system, as follows. We start with the definition of the sample mean of the quantity $\mathbf{q}$, defined by:

$$\mathbf{m}_k = \frac{1}{k} \sum_{i=1}^{k} \mathbf{q}_i \, ,$$

where k is the number of data points in the sample. It's easy to show that $\mathbf{m}_k$ satisfies the recursive relation

$$\mathbf{m}_{k+1} = \frac{k\,\mathbf{m}_k + \mathbf{q}_{k+1}}{(k+1)} \, ,$$

with $k \geq 0$ and $\mathbf{m}_0 \equiv \mathbf{0}$. Up until we have n data points, we want to keep considering all data points seen so far, so we must limit k to the range $0 \leq k < n$. Once we have seen n data points, we want to drop the earlier ones as we get more data, in such a way as to always have a sample of size n. See the figure below for a pictorial representation when $n = 4$. We keep all data points (the blue dots) up until we have $n = 4$ of them and then we start dropping the earlier ones but always keeping a sample size of, in this example, 4.



The recursive relation above lets us update the sample mean as each new data point arrives but it includes all data points from the start. Once we have n data points, we need a different recursive relation that lets us update the sample mean by including each new point while dropping the oldest point. This second recursive relation turns out to be:

$$\mathbf{m}_{n+k} = \mathbf{m}_{n+k-1} + \frac{\mathbf{q}_{n+k} - \mathbf{q}_k}{n} \, , \qquad k \geq 1 \, .$$

The same ideas apply to limiting the history 'stored' in the covariance matrices. The sample covariance matrix for a sample with k data points is defined by

$$\mathbf{\Sigma}_k = \frac{1}{k} \sum_{i=1}^{k} (\mathbf{q}_i - \mathbf{m}_k)(\mathbf{q}_i - \mathbf{m}_k)^t = \frac{1}{k} \sum_{i=1}^{k} \mathbf{q}_k \, \mathbf{q}_k^t - \mathbf{m}_k \, \mathbf{m}_k^t \, .$$

It, too, satisfies a recursive relation that lets us update its value until we have n data points,

$$\mathbf{\Sigma}_{k+1} = \frac{k}{k+1} \left(\mathbf{\Sigma}_k + \mathbf{m}_k \, \mathbf{m}_k^t\right) + \frac{\mathbf{q}_{k+1} \, \mathbf{q}_{k+1}^t}{k+1} - \mathbf{m}_{k+1} \, \mathbf{m}_{k+1}^t \, ,$$

with $0 \leq k < n$ and $\mathbf{\Sigma}_0 \equiv \mathbf{0}$. Similarly, in order to update its value when we want to keep only the most recent n data points, we can use a second recursive reltaion,

$$\mathbf{\Sigma}_{n+k} \; = \; \mathbf{\Sigma}_{n+k-1} + \frac{\mathbf{q}_{n+k} \, \mathbf{q}_{n+k}^t - \mathbf{q}_k \, \mathbf{q}_k^t}{n} - \left(\mathbf{m}_{n+k} \, \mathbf{m}_{n+k}^t - \mathbf{m}_{n+k-1} \, \mathbf{m}_{n+k-1}^t\right) ,$$

where $k \geq 1$.

3.7.1 Summary

The recursive relations we'll need to limit the system history are:

$$\begin{aligned}
\mathbf{m}_{k+1} &= \frac{k \, \mathbf{m}_k + \mathbf{q}_{k+1}}{(k+1)} \, , && \mathbf{m}_0 \equiv \mathbf{0} \, , && 0 \leq k < n \, , \\
\mathbf{m}_{n+k} &= \mathbf{m}_{n+k-1} + \frac{\mathbf{q}_{n+k} - \mathbf{q}_k}{n} \, , && && k \geq 1 \\
\mathbf{\Sigma}_{k+1} &= \frac{k}{k+1} \left(\mathbf{\Sigma}_k + \mathbf{m}_k \, \mathbf{m}_k^t\right) + \frac{\mathbf{q}_{k+1} \, \mathbf{q}_{k+1}^t}{k+1} - \mathbf{m}_{k+1} \, \mathbf{m}_{k+1}^t \, , && \mathbf{\Sigma}_0 \equiv \mathbf{0} \, , && 0 \leq k < n \, , \\
\mathbf{\Sigma}_{n+k} &= \mathbf{\Sigma}_{n+k-1} + \frac{\mathbf{q}_{n+k} \, \mathbf{q}_{n+k}^t - \mathbf{q}_k \, \mathbf{q}_k^t}{n} - \left(\mathbf{m}_{n+k} \, \mathbf{m}_{n+k}^t - \mathbf{m}_{n+k-1} \, \mathbf{m}_{n+k-1}^t\right) , && && k \geq 1 \, .
\end{aligned}$$

It's clear that we'll need to physically store the last n data points as the algorithm runs.

4 A Kalman filter for the device's acceleration

Having described the necessary mathematical results to implement a general Kalman filter, we can now narrow those results to the case of a filter to smoothen the three-dimensional acceleration vector, after it's been transformed to the appropriate coordinate system fixed to the ground. The necessary equations are as follows:

$$\begin{aligned}
\bar{\mathbf{a}}_{k|k-1} &= \hat{\mathbf{a}}_{k-1|k-1} \\
\mathbf{S}_k &= \hat{\mathbf{P}}_{k-1|k-1} + \bar{\boldsymbol{\Sigma}}_k + \tilde{\boldsymbol{\Sigma}}_k \\
\mathbf{K}_k &= \mathbf{I} - \tilde{\boldsymbol{\Sigma}}_k \, \mathbf{S}_k^{-1} \\
\hat{\mathbf{a}}_{k|k} &= \mathbf{K}_k \, \tilde{\mathbf{a}}_k + (\mathbf{I} - \mathbf{K}_k) \, \bar{\mathbf{a}}_{k|k-1} \\
\hat{\mathbf{P}}_{k|k} &= \mathbf{K}_k \, \tilde{\boldsymbol{\Sigma}}_k \, .
\end{aligned}$$

Eliminating the first expression by inserting it into the second-to-last one, the equations above simplify to:

$$\begin{aligned}
\mathbf{S}_k &= \hat{\mathbf{P}}_{k-1|k-1} + \bar{\boldsymbol{\Sigma}}_k + \tilde{\boldsymbol{\Sigma}}_k \\
\mathbf{K}_k &= \mathbf{I} - \tilde{\boldsymbol{\Sigma}}_k \, \mathbf{S}_k^{-1} \\
\hat{\mathbf{a}}_{k|k} &= \mathbf{K}_k \, \tilde{\mathbf{a}}_k + (\mathbf{I} - \mathbf{K}_k) \, \hat{\mathbf{a}}_{k-1|k-1} \\
\hat{\mathbf{P}}_{k|k} &= \mathbf{K}_k \, \tilde{\boldsymbol{\Sigma}}_k \, ,
\end{aligned}$$

where $k \geq 1$.

4.1 Estimating $\hat{\mathbf{a}}_{0|0}$, $\bar{\boldsymbol{\Sigma}}_k$ and $\tilde{\boldsymbol{\Sigma}}_k$

A reasonable (albeit debatable) approach to estimating the process covariance matrix $\bar{\boldsymbol{\Sigma}}_k$ is to compute the sample variance of the a-posteriori acceleration estimates $\hat{\mathbf{a}}_{k|k}$, subjected to the history-limiting ideas discussed earlier. Similarly, we'll estimate the measurement covariance matrix $\tilde{\boldsymbol{\Sigma}}_k$ by computing the sample variance of the measured acceleration data $\tilde{\mathbf{a}}_k$ (after transforming it to the appropriate coordinate system), also subjected to the history-limiting process already described that prevents the filter from sticking too stiffly to the average acceleration observed throughout the entire history of the system. As for $\hat{\mathbf{a}}_{0|0}$, we can set its value to the zero vector in three dimensions.

5 A Kalman filter for the device's velocity

For the velocity filter, we'll use the following version of the Kalman equations:

$$\begin{aligned}
\bar{\mathbf{v}}_{k|k-1} &= \hat{\mathbf{v}}_{k-1|k-1} + \hat{\mathbf{a}}_{k-1|k-1}\,(t_k - t_{k-1}) \\
\mathbf{S}_k &= \hat{\mathbf{P}}_{k-1|k-1} + \bar{\boldsymbol{\Sigma}}_k + \tilde{\boldsymbol{\Sigma}}_k \\
\mathbf{K}_k &= \mathbf{I} - \tilde{\boldsymbol{\Sigma}}_k\,\mathbf{S}_k^{-1} \\
\hat{\mathbf{v}}_{k|k} &= \mathbf{K}_k\,\tilde{\mathbf{v}}_k + (\mathbf{I} - \mathbf{K}_k)\,\bar{\mathbf{v}}_{k|k-1} \\
\hat{\mathbf{P}}_{k|k} &= \mathbf{K}_k\,\tilde{\boldsymbol{\Sigma}}_k\,.
\end{aligned}$$

Naturally, the various matrices here are not the same as those in the acceleration filter, despite having the same symbols. Note that we're using the a-posteriori estimate of the acceleration at time t_{k-1}, from the acceleration filter, as a control vector for the velocity.

5.1 Estimating $\hat{\mathbf{v}}_{0|0}$, $\bar{\boldsymbol{\Sigma}}_k$ and $\tilde{\boldsymbol{\Sigma}}_k$

Estimating these matrices is quite a bit trickier now. In order to estimate $\bar{\boldsymbol{\Sigma}}_k$, consider first the sample mean of $(\mathbf{a}\,\Delta t)_k \equiv \hat{\mathbf{a}}_{k|k}\,(t_{k+1} - t_k)$ after n steps, defined by

$$(\mathbf{a}\,\Delta t)_n = \frac{1}{n}\sum_{k=1}^{n} \hat{\mathbf{a}}_{k|k}\,(t_{k+1} - t_k)\,.$$

Then the process covariance matrix after n steps can be estimated as the sample variance of the deviation $[\,(\mathbf{a}\,\Delta t)_k - (\mathbf{a}\,\Delta t)_n\,]$ from the sample mean, namely,

$$\bar{\boldsymbol{\Sigma}}_n = \frac{1}{n}\sum_{k=1}^{n} [\,(\mathbf{a}\,\Delta t)_k - (\mathbf{a}\,\Delta t)_n\,]\,[\,(\mathbf{a}\,\Delta t)_k - (\mathbf{a}\,\Delta t)_n\,]^t\,.$$

Similarly, we may estimate the measurement covariance matrix by the sample variance of the deviation $[\,\tilde{\mathbf{v}}_k - (\mathbf{a}\,\Delta t)_n\,]$,

$$\tilde{\boldsymbol{\Sigma}}_n = \frac{1}{n}\sum_{k=1}^{n} [\,\tilde{\mathbf{v}}_k - (\mathbf{a}\,\Delta t)_n\,]\,[\,\tilde{\mathbf{v}}_k - (\mathbf{a}\,\Delta t)_n\,]^t\,.$$

The reason for considering deviations from $(\mathbf{a}\,\Delta t)_n$ is that we're trying to estimate the correlation between the velocity values *after* removing any trending behaviour due to sustained accelerations. Naturally, the mean value $(\mathbf{a}\,\Delta t)_n$ is also history-limited as described previously, although with a sample size possibly different from that used for the acceleration filter. As for $\hat{\mathbf{v}}_{0|0}$, we can set it to the initial velocity measurement from the GPS or even to the zero vector in two dimensions.

6 Customizable Parameters

As designed, the only two customizable parameters of the motion detector itself (that is, not considering possible customizable parameters for client APIs that use the motion detector) are the number of data points to 'remember' for each of the two Kalman filters. The larger these values are, the stiffer the corresponding filters will be.

■