翻译：A practical approach to Kalman filter and how to implement it —— Lauszus

网址：http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/

前言：Kalman 滤波是我很长时间都没弄明白的一个东西，每次读写文档都感觉略懂，但又不是真的明白。最近工作上遇到需要用 Kalman 滤波来融合加速度计和陀螺仪，虽然能在网上直接找到代码，但还是希望能够弄明白基本思想。在网络上搜索了很久，虽然有许多相关的内容，但写的都很模糊（感觉也是从别处找来代码并随便写个总结），好在这次找到了上面这篇文章，觉得有必要翻译一下，并不是为了把英文变成中文，只是想让自己留下更深的印象。

—— 李立群

开始：

I have for a long time been interested in Kalman filters and how they work, I also used Kalman filter for my Balancing robot, but I never explained how it acutally was implemented. Actually I had never taken the time to sit down with a pen and a piece of paper and try to do the math myself, so I actually did not know how it was implemented. It turned out to be a good thing, as I actually discovered a mistake in the original code, but I will get back to that later.

我一直对 Kalman 滤波感兴趣，也曾经把它用于平衡机器人，但我从来没有想过它究竟是怎么工作的。实际上，我从来没有时间坐下来，用纸笔去推导其中的数学，所以并不清楚实现原理。实际上，如果能这么做是非常好的，经过这个过程我发现了之前代码里的一个问题，这在后面会提到。

I actually wrote about the Kalman filter as my master assignment in high school back in December 2011. But I only used the Kalman filter to calculate the true voltage of a DC signal modulated by known Gaussian white noise. My assignment can be found in the following zip file: http://www.tkjelectronics.dk/uploads/Kalman_SRP.zip. It is danish, but you can properly use google translate to translate some of it. If you got any specific questions regarding the assignment, then ask in the comments below.

实际上在我高中的作业中，曾经用到过 Kalman 滤波，那是在 2011 年十一月。但那时，我只是用它来从混杂着已知高斯噪声的直流信号上计算真实电压。我的作业可以在以下链接找到：http://www.tkjelectronics.dk/uploads/Kalman_SRP.zip。作业是丹麦语写的，但相信 Google 翻译能让你看明白大部分内容。如果对作业有什么特别的问题，可以在下面留言。

Okay, but back to the subject. As I said I had never taken the time to sit down and do the math regarding the Kalman filter based on an accelerometer and a gyroscope. It was not as hard as I expected, but I must confess that I still have not studied the deeper theory behind, on why it actually works. But for me, and most people out there, I am more interested in implementing the filter, than in the deeper theory behind and why the equations works.

好的，回到主题。就像我说过的，我从来没有坐下来推导其中的数学。这么做其实这并没有想象的那么难，但我必须说明我其实没有学习过 Kalman 滤波背后的理论——为什么这些公式能够工作。但对于我以及大多数人来说，其实更希望了解如何使用它，而不是去专研背后深刻的理论。

Before we begin you must have some basic knowledge about matrices like multiplication of matrices and transposing of matrices. If not, then please take a look at the following websites:

- http://en.wikipedia.org/wiki/Matrix_multiplication#Matrix_product_.28two_matrices.29
- http://www.mathwarehouse.com/algebra/matrix/multiply-matrix.php
- http://en.wikipedia.org/wiki/Transpose
- http://en.wikipedia.org/wiki/Covariance_matrix

在我们正式开始前，你应该确保自己了解一些矩阵运算的基础，例如矩阵相乘和转置。如果不了解，请

先阅读以下网页。

For those of you who do not know what a Kalman filter is, it is an algorithm which uses a series of measurements observed over time, in this context an accelerometer and a gyroscope. These measurements will contain noise that will contribute to the error of the measurement. The Kalman filter will then try to estimate the state of the system, based on the current and previous states, that tend to be more precise that than the measurements alone.

对那些没听说过 Kalman 滤波的人来说，它是一种算法，其输入是一系列时序观测值，在下面要介绍的例子里，这些观测值特指加速度计和陀螺仪的读数。这些观测值包含误差，所以会导致错误的测量结果。Kalman 滤波的目的是基于这些有误差的观测值来估计系统当前状态，使得比原始测量更加准确。

In this context the problem is that the accelerometer is in general very noise when it is used to measure the gravitational acceleration since the robot is moving back and forth. The problem with the gyro is that it drifts over time – just like a spinning wheel- gyro will start to fall down when it is losing speed.In short you can say that you can only trust the gyroscope on a short term while you can only trust the accelerometer on a long term.

在下面的例子里，加速度计用于测量重力的方向，由于机器人在持续移动，加速度计的原始读数通常带有很大的噪声。另一方面，陀螺仪的问题是其读数随时间漂移——就像一个旋转的陀螺，一旦减速就会倒下来。简而言之，对于陀螺仪，我们倾向于相信其短时间内的读数；对于加速度计，我们更相信其长时间内的读数的平均。

There is actually a very easy way to deal with this by using a complimentary filter, which basicly just consist of a digital low-pass filter on the accelerometer and digital high-pass filter on the gyroscope readings. But it is not as accurate as the Kalman filter, but other people have succesfully build balancing robots using a fine-tuned complimentary filter.

对于这个问题，实际上有个比较容易的解决方法——complimentary 滤波，其基本思想是对加速度计使用低通滤波，而对陀螺仪采用高通滤波。但是，这不如 Kalman 滤波来的有效，但其实经过精细调节的 complimentarty 滤波也是能够达到比较好的效果的。

More information about gyroscopes, accelerometer and complimentary filters can be found in this pdf. A comparison between a complimentary filter and a Kalman filter can be found in the following blog post.

更多关于陀螺仪、加速度计和 complimentary 滤波的信息可以在这里找到（链接无效）。这一篇博文（http://robottini.altervista.org/kalman-filter-vs-complementary-filter） 对 complimentary 滤波和 Kalman 滤波进行了比较。

The Kalman filter operates by producing a statistically optimal estimate of the system state based upon the measurement(s). To do this it will need to know the noise of the input to the filter called the measurement noise, but also the noise of the system itself called the process noise. To do this the noise has to be Gaussian distributed and have a mean of zero, luckily for us most random noise have this characteristic.

Kalman 滤波在原始观测值的基础上从统计上给出最优的系统状态估计。要达到这个目的，我们需要知道观测误差分布和系统运行误差分布。要运用 Kalman 滤波，这些误差随机变量都应该符合均值为 0 的高斯分布。幸运的是，自然界的大多数随机噪声都满足这个条件。

For more information about the theory behind the filter take a look at the following pages:

- http://en.wikipedia.org/wiki/Kalman_filter
- http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf
- http://academic.csuohio.edu/simond/courses/eec644/kalman.pdf

更多相关的理论知识可参见下面的链接。

The system state x_k

系统的状态x_k

The next of this article might seem very confusing for some, but I promise you if you grab a pen and a piece of paper and try to follow along it is not that hard if you are reasonable at math.

下面的部分对于有些人来说可能难以理解，但我相信只要你拿上一支笔和一张纸，理解下面所说的东西并不是很困难的时期，这只需要你了解一些基本的数学知识。

If you, like me, do not have a calculator or computer program that can work with matrices, then I recommend the free online calculator Wolfram Alpha. I used it for all the calculations in this article.

如果你像我一样手头上没有计算器和电脑程序可以处理矩阵，那么我建议你试试免费的在线计算器 Wolfram Alpha（https://www.wolframalpha.com/）。这篇文章中所有的计算都是利用它完成的。

I will use the same notation as the Wikipedia article, but I will like to note that when the matrixes are constants and does not depend on the current time you do not have to write the k after them. So for instance F_k can be simplified to F.

我将使用与 Wikipedia 一样的符号系统，但我想大家注意一点，当矩阵是常数即并不依赖当前时间时，你不需要在它们后面加上一个 k 做下标。例如F_k将被简化为F。

Also I would like to write a small explanation of the other aspects of the notations. First I will make a note about whats called the previous state:

另外，我想先解释一些基本概念。首先，我想强调一下所谓的“前状态“：

$$\hat{\boldsymbol{x}}_{k-1|k-1}$$

Which is the previous estimated state based on the previous state and the estimates of the states before it.

它是指在前一时刻（k-1）估计的系统状态，前状态的估计基于所有 k-1 时刻之前（包括 k-1 时刻）的系统输入。

The next is the a priori state:

另一个概念是“先验状态”

$$\hat{\boldsymbol{x}}_{k|k-1}$$

A priori means the estimate of the state matrix at the current time k based on the previous state of the system and the estimates of the states before it.

它仍然基于所有 k-1 时刻之前（包括 k-1 时刻）的信息，代表在观测前（这里特指观测加速度计读数之前）的系统当前状态。

The last one is called a posteriori state:

最后是“后验状态”

$$\hat{\boldsymbol{x}}_{k|k}$$

Is the estimated of the state at time k given observations up to and including at time k.

是指基于 k 时刻之前（包括 k 时刻）的所有信息（包括加速度计和陀螺仪）估计的系统当前状态。 注意：统一来说，$\hat{x}_{n|m}$ $m \leq n$ 均是指根据从 0 时刻到 m 时刻的所有观测来预测 n 时刻的系统状态。

The problem is that the system state itself is hidden and can only be observed through observation z_k. This is also called a Hidden Markov model.

系统的真实状态其实是无法被直接测量的，只能观测到现象z_k，这种模型被称为隐马尔科夫模型。

This means that the state will be based upon the state at time k and all the previous states. That also means that you can not trust the estimate of the state before the Kalman filter has stabilized – take a look at the graph at the front page of my assignment.

这意味着当前状态由之前所有状态决定，也意味着在 Kalman 滤波稳定之前，对当前状态的估计是不可信的——这一点可以参考我作业封面上的图。（这段话讲得没什么逻辑，忽略就好）

The hat over the $\hat{x}$ means that is the estimate of the state. Unlike just a single x which means the

true state – the one we are trying to estimate.
So the notation for the state at time k is:
状态 x 上面的小帽（$\hat{x}$）表示这是一个估计的状态，没有小帽的 x 则表示真实状态，这才是我们真正想估计的。对应的在 k 时刻的系统状态是：

$$x_k$$

The state of the system at time k if given by:
k 时刻的系统状态可以由下面公式描述：

$$\boldsymbol{x}_k = \boldsymbol{F}x_{k-1} + \boldsymbol{B}u_k + w_k$$

Where x_k is the state matrix which is given by:
其中x_k是系统状态矩阵，在这个例子里表示为：

$$\boldsymbol{x}_k = \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_k$$

As you can see the output of the filter will be the angle θ but also the bias $\dot{\theta}_b$ based upon the measurements from the accelerometer and gyroscope. The bias is the amount the gyro has drifted. This means that one can get the true rate by subtracting the bias from the gyro measurement.
正如你看到的，系统状态包括角度θ和角速度偏斜$\dot{\theta}_b$。所谓偏斜，是指陀螺仪硬件本身引入的误差。一旦知晓陀螺仪的偏斜，就能从测量值上减去误差，得到真实的角度变化。

The next is the F matrix, which is the state transition model which is applied to the previous state x_{k-1}.
F 矩阵是系统状态转移模型，应用在系统的前状态x_{k-1}上。
In this case F is defined as:
这种情况下，F 定义为：

$$\boldsymbol{F} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix}$$

I know that the $-\Delta t$ might seem confusing, but it will make sense later (take a look at my comment).
我知道$-\Delta t$看起来很奇怪，但我会在后面解释的。
The next is the control input u_k, in this case it is the gyroscope measurement in degrees per second (°/s) at time k, this is also called the rate $\dot{\theta}$. We will actually rewrite the state equation as:
另一方面，控制输入记做u_k。在这个场景下，它表示陀螺仪在时刻 k 的读数，单位是度每秒，这也可以叫做角速度$\dot{\theta}$。将它放进上面的等式得到:

$$\boldsymbol{x}_k = \boldsymbol{F}x_{k-1} + \boldsymbol{B}\dot{\theta}_k + w_k$$

The next thing is the B matrix. Which is called the control-input model, which is defined as:
B 矩阵叫做控制输入模型，定义为：

$$\boldsymbol{B} = \begin{bmatrix} \Delta t \\ 0 \end{bmatrix}$$

This makes perfectly sense as you will get the angle $\theta$when you multiply the rate $\dot{\theta}$ by the delta time Δt and since we can not calculate the bias directly based on the rate we will set the bottom of the matrix to 0.
这符合实际的物理意义，为了得到估计的角度θ，我们需要将角速度$\dot{\theta}$乘以时间Δt。另一方面，我们没法直接计算角速度偏斜的值，所以我们只能用 0 来作为系统控制输入。

w_k is process noise which is Gaussian distributed with a zero mean and with covariance Q to the

time k:

w_k表示系统运行过程中的噪声，它是一个以零为均值的高斯分布，其方差为 Q，方差是与时间相关的：

$$\boldsymbol{w}_k \sim N\left(0, \boldsymbol{Q}_k\right)$$

$Q_k$is the process noise covariance matrix and in this case the covariance matrix of the state estimate of the accelerometer and bias. In this case we will consider the estimate of the bias and the accelerometer to be independent, so it's actually just equal to the variance of the estimate of the accelerometer and bias.

Q_k是系统运行噪声的协方差矩阵，在这里矩阵刻画陀螺仪及其偏斜估计的协方差。这里我们假设陀螺仪及其偏斜的估计之间是相互独立的，所以协方差矩阵实际上只是两者估计的方差构成的对角阵。

The final matrix is defined as so:

最后，这个矩阵定义如下：

$$\boldsymbol{Q}_k = \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t$$

As you can see the Q_k covariance matrix depends on the current time k, so the accelerometer variance Q_θ and the variance of the bias $Q_{\dot{\theta}_b}$ is multiplied by the delta time Δt.

你可以看到Q_k协方差矩阵是与时间 k 关联的，陀螺仪的方差Q_θ和偏斜的方差$Q_{\dot{\theta}_b}$需要乘以时间差Δt。

This makes sense as the process noise will be larger as longer time it is since the last update of the state. For instance the gyro could have drifted.

这样定义是有物理含义的，距离上次估计的时间越长这段时间内形成的误差可能就越大。例如，陀螺仪的读数就是随时间越来越偏移的。

You will have to know these constants for the Kalman filter to work.

在使用 Kalman 滤波前，你需要知道这些参数的值。

Note if you set a larger value, the more noise in the estimation of the state. So for instance if the estimated angle starts to drift you have to increase the value of $Q_{\dot{\theta}_b}$. Otherwise if the estimate tends to be slow you are trusting the estimate of the angle too much and should try to decrease the value of Q_θ to make it more responsive.

注意，如果你将方差设为一个较大的值，也就表示估计的状态会有较大误差。例如，如果你发现估计的角度不断随时间偏移，你应该增大偏移的方差$Q_{\dot{\theta}_b}$；否则，如果发现估计的角度变化过于缓慢，这是由于太过相信观测到的角度（此处应该是指从加速度计的观测）而造成的，就应该降低Q_θ来提高系统的响应速度。

The measurement $\boldsymbol{z}_k$

观测值z_k

Now we will take a look at the observation or measurement z_k of the true state x_k. The observation z_k is given by:

现在我们来看看对于真实状态x_k的观测值z_k，观测值z_k表示如下：

$$\boldsymbol{z}_k = \boldsymbol{H} x_k + v_k$$

As you can see the measurement z_k is given by the current state x_k multiplied by the H matrix plus the measurement noise v_k.

你可以看到观测值z_k是当前状态x_k乘以 H 矩阵加上观测误差v_k得到的。

H is called the observation model and is used to map the true state space into the observed space. The true state can not be observed. Since the measurement is just the measurement from the accelerometer, H is given by:

H 被称为观测模型，它将状态空间向观测空间映射。这里需要明白真实状态是隐含的，无法被直接观察。在这个例子里，观测是通过加速度计进行的（需要注意一点，陀螺仪的读数在这里不能称之为观测，而是系统运行的控制输入），H 定义为：

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

The noise of the measurement have to be Gaussian distributed as well with a zero mean and R as the covariance:

观测的误差必须是高斯分布的，并且均值为零方差为 R：

$$\boldsymbol{v}_k \sim N\ (0, \boldsymbol{R})$$

But as R is not a matrix the measurement noise is just equal to the variance of the measurement, since the covariance of the same variable is equal to the variance. See this page for more information.

这里的 R 不是矩阵而只是观测的方差，因为单个随机数的协方差其实就是方差。

Now we can define R as so:

下面定义 R：

$$\boldsymbol{R} = E\begin{bmatrix} v_k & {v_k}^T \end{bmatrix} = var(v_k)$$

More information about covariance can be found on Wikipedia and in my assignment.
We will assume that the measurement noise is the same and does not depend on the time k:

更多协方差的知识可以阅读相关的 Wikipedia 和我的作业。我们假定方差不是随时间变化的，所以：

$$var(v_k) = var(v)$$

Note that if you set the measurement noise variance $var(v)$too high the filter will respond really slowly as it is trusting new measurements less, but if it is too small the value might overshoot and be noisy since we trust the accelerometer measurements too much.

注意到如果$var(v)$被设置为过大则系统对加速度计的响应就会很慢，反之则可能会抖动的非常厉害。

So to round up you have to find the the process noise variances Q_θ and $Q_{\dot{\theta}_b}$ and the measurement variance of the measurement noise $var(v)$. There are multiple ways to find them, but it is out of the aspect of this article.

总而言之，需要找到合理的过程误差Q_θ和$Q_{\dot{\theta}_b}$以及观测的方差$var(v)$。有许多种获得它们的方法，但这不在本文的范围之内。

The Kalman filter equations

Okay now to the equations we will use to estimate the true state of the system at time k $\hat{x}_k$. Some clever guys came up with equations found below to estimate the state of the system.
The equations can be written more compact, but I prefer to have them stretched out, so it is easier to implement and understand the different steps.

好的，现在我们将介绍用于估计系统在 k 时刻状态$\hat{x}_k$的公式。一些聪明人推导出了下面一些公式来预测系统的当前状态。这些公式本可以写得更加紧凑一些，但我喜欢慢慢把它们推导出来，这样便于大家实现和理解这些步骤。

Predict

预测

In the first two equations we will try to predict the current state and the error covariance matrix at time k. First the filter will try to estimate the current state based on all the previous states and the gyro measurement:

前两个要介绍的公式的作用是预测时间 k 时的当前状态以及错误协方差矩阵。首先，滤波系统根据前一时刻的系统状态和陀螺仪读数来计算当前状态：

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}\hat{x}_{k-1|k-1} + \boldsymbol{B}\dot{\theta}_k$$

That is also why it is called a control input, since we use it as an extra input to estimate the state at the current time k called the a priori state $\hat{x}_{k|k-1}$ as described in the beginning of the article.
这就是为什么它被称为控制输入，因为我们用它作为额外输入来估计 k 时刻的系统状态，这个估计记作$\hat{x}_{k|k-1}$。

The next thing is that we will try to estimate the a priori error covariance matrix $P_{k|k-1}$ based on the previous error covariance matrix $P_{k-1|k-1}$, which is defined as:
我们尝试上一时刻（k-1）估计的错误协方差矩阵$P_{k-1|k-1}$来预测先验错误协方差矩阵$P_{k|k-1}$，定义为：

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}^T + \boldsymbol{Q}_k$$

This matrix is used to estimate how much we trust the current values of the estimated state. The smaller the more we trust the current estimated state. The principle of the equation above is actually pretty easy to understand, as it is pretty obvious that the error covariance will increase since we last updated the estimate of the state, therefore we multiplied the error covariance matrix by the state transition model F and the transpose of that F^T and add the current process noise $Q_k$at time k.
这个矩阵反映我们对估计结果的信任，矩阵对应的错误越小表示我们越信任当前估计的系统状态。上面等式的原理其实非常容易理解。非常明显，错误的协方差从上一轮更新后会增加，所以我们用状态转移矩阵 F 及其转置F^T乘以上一时刻的错误协方差矩阵，然后再加上 k-1 时刻到 k 时刻的系统运行误差Q_k。

The error covariance matrix P in our case is a 2×2 matrix:
错误协方差矩阵 P 在这里是 2×2 的矩阵：

$$\boldsymbol{P} = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}$$

Update
更新
The fist thing we will do is to compute the difference between the measurement z_k and the a priori state $x_{k|k-1}$, this is also called the innovation:
我们首先要做的是计算观测值z_k与先验状态$x_{k|k-1}$之间的差别，这也被称为**创新**：

$$\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - \boldsymbol{H}\hat{x}_{k|k-1}$$

The observation model H is used to map the a priori state $x_{k|k-1}$ into the observed space which is the measurement from the accelerometer, therefore the innovation is not a matrix
观测模型 H 用来将先验状态$x_{k|k-1}$映射到可观测空间，该空间对应于加速度计的测量值，所以创新不是一个矩阵

$$\tilde{\boldsymbol{y}}_k = \left[\tilde{\boldsymbol{y}}\right]_k$$

The next thing we will do is calculate whats called the innovation covariance:
下面要做的是计算创新的方差：

$$\boldsymbol{S}_k = \boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^T + \boldsymbol{R}$$

What it does is that it tries to predict how much we should trust the measurement based on the a priori error covariance matrix $P_{k|k-1}$ and the measurement covariance matrix R. The observation model H is used to map the a priori error covariance matrix $P_{k|k-1}$ into observed space.
这里做的是估计我们应该多信任观测值，这是根据先验错误协方差矩阵$P_{k|k-1}$和测量误差协方差矩阵 R 得到的。观测模型 H 用于将先验错误协方差矩阵$P_{k|k-1}$映射到可观测空间。

The bigger the value of the measurement noise the larger the value of S, this means that we do not trust the incoming measurement that much.
In this case S is not a matrix and is just written as:
测量误差越大则 S 的值就越大，这时候我们不能太相信观测值，这个例子里 S 不是一个矩阵，可写作：

$$\boldsymbol{S}_k = \left[\boldsymbol{S}\right]_k$$

The next step is to calculate the Kalman gain. The Kalman gain is used to to indicate how much we trust the innovation and is defined as:

下面要计算的是 Kalman 增益，它用于标志我们能够多么信任创新：

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}^T\boldsymbol{S}_k^{-1}$$

You can see that if we do not trust the innovation that much the innovation covariance S will be high and if we trust the estimate of the state then the error covariance matrix P will be small the Kalman gain will therefore be small and opposite if we trust the innovation but does not trust the estimation of the current state.

可以看到如果我们不相信创新的结果，那么这可能是由于 S 比较大；如果我们信任状态估计的结果，那么意味着 P 是比较小的；Kalman 增益较小意味着我们更加信任状态估计的结果，反之则更加信任观测值。

If you take a deeper look you can see that the transpose of the observation model H is used to map the state of the error covariance matrix P into observed space. We then compare the error covariance matrix by multiplying with the inverse of the innovation covariance S.

仔细观察会发现观测模型 H 的转置是用来将状态错误协方差矩阵 P 映射到可观测空间，然后我们再比较估计错误协方差矩阵和创新错误协方差矩阵的大小。

This make sense as we will use the observation model H to extract data from the state error covariance and compare that with the current estimate of the innovation covariance.

这里的物理含义是通过观测模型 H 把数据从状态错误协方差中提取出来，然后跟当前创新协方差矩阵去比较。

Note that if you do not know the state at startup you can set the error covariance matrix like so:

$$\boldsymbol{P} = \begin{bmatrix} L & 0 \\ 0 & L \end{bmatrix}$$

Where L represent a large number.

如果你不知道起始状态下的错误协方差矩阵，那么可以把它定义如下，其中 L 表示一个较大的值。

For my balancing robot I know the starting angle and I find the bias of the gyro at startup by calibrating, so I assume that the state will be known at startup, so I initialize the error covariance matrix like so:

对于我的平衡机器人来说，我是知道起始角度的，而偏斜的初始值我是通过校准来得到，所以我已知所有的初始状态，那么错误协方差矩阵定义为：

$$\boldsymbol{P} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Take a look at my calibration routine for more information.

有兴趣可以看看我的校准方法。

In this case the Kalman gain is a 2×1 matrix:

这样的情况下，Kalman 增益是一个 2×1 的矩阵：

$$\boldsymbol{K} = \begin{bmatrix} K_0 \\ K_1 \end{bmatrix}$$

Now we can update the a posteriori estimate of the current state:

现在我们可以更新后验状态：

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\,\tilde{\boldsymbol{y}}_k$$

This is done by adding the a priori state $\hat{x}_{k|k-1}$ with the Kalman gain multiplied by the innovation $\tilde{y}_k$.

这里只是将先验状态$\hat{x}_{k|k-1}$与 Kalman 增益为因子的创新$\tilde{y}_k$相乘。

Remember that the innovation $\tilde{y}_k$ is the difference between the measurement z_k and the estimated priori state $\hat{x}_{k|k-1}$, so the innovation can both be positive and negative.
记住创新$\tilde{y}_k$是观测值z_k和估计先验状态$\hat{x}_{k|k-1}$的差，所以创新本身可以是正或负。

A little simplified the equation can be understood as we simply correct the estimate of the a priori state $\hat{x}_{k|k-1}$, that was calculated using the previous state and the gyro measurement, with the measurement – in this case the accelerometer.
这个式子很好理解，因为我们只是用加速度计的观测值来修正通过上一时刻估计的状态和陀螺仪得到的先验状态$\hat{x}_{k|k-1}$。

The last thing we will do is update the a posteriori error covariance matrix:
最后我们要做的就是更新后验的错误协方差矩阵：

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_{k|k-1}$$

Where I is called the identity matrix and is defined as:
其中 I 称为单位矩阵，定义为：

$$\boldsymbol{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

What the filter is doing is that it is basically self-correcting the error covariance matrix based on how much we corrected the estimate. This make sense as we corrected the state based the a priori error covariance matrix $P_{k|k-1}$, but also the innovation covariance S_k.
滤波系统做的是基于前错误协方差矩阵$P_{k|k-1}$来修正当前的错误协方差矩阵，需要注意的是 Kalman 增益本身是考虑了创新的协方差矩阵S_k的。

Implementing the filter

实现滤波

In this section I will use the equation from above to implement the filter into a simple c++ code that can be used for balancing robots, quad-copters and other applications where you need to compute the angle, bias or rate.
在这节，我会用前面介绍的那些式子来用 c++代码实现一个简单的滤波器，它可以用于平衡机器人、四轴飞行器和一些其它的应用，只要你需要考虑角度、偏斜和速率之类的。

In case you want the code next to you, it can be found at github:
https://github.com/TKJElectronics/KalmanFilter.
如果你需要这些代码，可以在 github 上找到。

I will simply write the equations at the top of each step and then simplify them after that I will write how it is can be done i C and finally I will link to calculations done in Wolfram Alpha in the bottom of each step, as I used them to do the calculation.

Step 1:

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}\hat{x}_{k-1|k-1} + \boldsymbol{B}\dot{\theta}_k$$
$$\begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \dot{\theta}_k$$
$$= \begin{bmatrix} \theta - \dot{\theta}_b\Delta t \\ \dot{\theta}_b \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \dot{\theta}_k$$

$$= \begin{bmatrix} \theta - \dot{\theta}_b \Delta t + \dot{\theta} \Delta t \\ \dot{\theta}_b \end{bmatrix}$$
$$= \begin{bmatrix} \theta + \Delta t(\dot{\theta} - \dot{\theta}_b) \\ \dot{\theta}_b \end{bmatrix}$$

As you can see the a priori estimate of the angle is $\hat{\theta}_{k|k-1}$ is equal to the estimate of the previous state $\hat{\theta}_{k-1|k-1}$ plus the unbiased rate times the delta time Δt.
Since we can not directly measure the bias the estimate of the a priori bias is just equal to the previous one.
你可以看到上面式子里的角度的先验估计$\hat{\theta}_{k|k-1}$等于上次估计的状态$\hat{\theta}_{k-1|k-1}$加上去掉偏斜的角速度乘以时间变化Δt。

This can be written in C like so:

```
rate = newRate - bias;
angle += dt * rate;
```

Note that I calculate the unbiased rate, so it can be be used by the user as well.
这可以写成下面的 C 代码：
这里我计算的去偏斜的速率，它可以被其他人类似采用。

Wolfram Alpha links:
Eq. 1.1

Step 2:

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}^T + \boldsymbol{Q}_k$$
$$\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k-1|k-1} \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t$$
$$= \begin{bmatrix} P_{00} - \Delta t P_{10} & P_{01} - \Delta t P_{11} \\ P_{10} & P_{11} \end{bmatrix}_{k-1|k-1} \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t$$
$$= \begin{bmatrix} P_{00} - \Delta t P_{10} - \Delta t(P_{01} - \Delta t P_{11}) & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t$$
$$= \begin{bmatrix} P_{00} - \Delta t P_{10} - \Delta t(P_{01} - \Delta t P_{11}) + Q_\theta \Delta t & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} + Q_{\dot{\theta}_b} \Delta t \end{bmatrix}$$
$$= \begin{bmatrix} P_{00} + \Delta t(\Delta t P_{11} - P_{01} - P_{10} + Q_\theta) & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} + Q_{\dot{\theta}_b} \Delta t \end{bmatrix}$$

The equations above can be written in C like so:
这些等式写作 C 代码为：

```
P[0][0] += dt * (dt*P[1][1] - P[0][1] - P[1][0] + Q_angle);
P[0][1] -= dt * P[1][1];
P[1][0] -= dt * P[1][1];
P[1][1] += Q_gyroBias * dt;
```

Note that this is the part of the code that there was an error in in the original code that I used.
这是我之前犯过错误的地方。

Wolfram Alpha links:
Eq. 2.1
Eq. 2.2
Eq. 2.3
Eq. 2.4

Step 3:

$$\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - \boldsymbol{H}\hat{x}_{k|k-1}$$

$= \boldsymbol{z}_k - \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1}$
$= \boldsymbol{z}_k - \theta_{k|k-1}$
The innovation can be calculated in C like so:

```
y = newAngle - angle;
```

Wolfram Alpha links:
Eq. 3.1

Step 4:

$\boldsymbol{S}_k = \boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^T + \boldsymbol{R}$
$= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \boldsymbol{R}$
$= P_{00k|k-1} + \boldsymbol{R}$
$= P_{00k|k-1} + var(v)$
Again the C code is pretty simple:

```
S = P[0][0] + R_angle;
```

Wolfram Alpha links:
Eq. 4.1

Step 5:

$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}^T\boldsymbol{S}_k^{-1}$
$\begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \boldsymbol{S}_k^{-1}$
$= \begin{bmatrix} P_{00} \\ P_{10} \end{bmatrix}_{k|k-1} \boldsymbol{S}_k^{-1}$
$= \frac{\begin{bmatrix} P_{00} \\ P_{10} \end{bmatrix}_{k|k-1}}{\boldsymbol{S}_k}$
Note that in other cases S can be a matrix and you can not just simply divide P by S. Instead you have to calculate the inverse of the matrix. See the following page for more information on how to do so.
注意在某些情况下 S 可能是一个矩阵，所以不能直接除。这时候你需要先计算矩阵的逆，然后再相乘。
The C implementation looks like this:

```
K[0] = P[0][0] / S;
K[1] = P[1][0] / S;
```

Wolfram Alpha links:
Eq. 5.1

Step 6:

$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\, \tilde{\boldsymbol{y}}_k$
$\begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k} = \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} + \begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k \tilde{\boldsymbol{y}}_k$
$= \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} + \begin{bmatrix} K_0\, \tilde{\boldsymbol{y}} \\ K_1\, \tilde{\boldsymbol{y}} \end{bmatrix}_k$
Yet again the equation end up pretty short, and can be written as so in C:

```
angle += K[0] * y;
bias += K[1] * y;
```

Step 7:

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H})\boldsymbol{P}_{k|k-1}$$

$$\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k} = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k \begin{bmatrix} 1 & 0 \end{bmatrix} \right) \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1}$$

$$= \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} K_0 & 0 \\ K_1 & 0 \end{bmatrix}_k \right) \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1}$$

$$= \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} - \begin{bmatrix} K_0 P_{00} & K_0 P_{01} \\ K_1 P_{00} & K_1 P_{01} \end{bmatrix}$$

Remember that we decrease the error covariance matrix again, since the error of the estimate of the state has been decreased.

注意我们其实降低了错误的协方差矩阵，这是由于我们降低了状态估计的错误。

The C code looks like this:

```
P[0][0] -= K[0] * P[0][0];
P[0][1] -= K[0] * P[0][1];
P[1][0] -= K[1] * P[0][0];
P[1][1] -= K[1] * P[0][1];
```

Wolfram Alpha links:
Eq. 7.1
Eq. 7.2
Eq. 7.3

Note that I have found that the following variances works perfectly for most IMUs:

我发现下面的变量设定适合大多数的 IMU 传感器：

```
const double Q_angle = 0.001;
const double Q_gyroBias = 0.003;
const double R_angle = 0.03;
```

Remember that it’s very important to set the target angle at startup if you need to use the output at startup. For more information, see the calibration routine for my balancing robot.

注意如果你想一开始就用滤波系统的输出，那你必须设定一个相对准确的初始值。

In case you missed it here is the library I wrote a library that can be used by any microcontroller that supports floating math. The source code can be found at github:
https://github.com/TKJElectronics/KalmanFilter.

如果你担心这里的代码遗漏了什么细节，可以看看我的 github，这里有适合大多数支持浮点的微处理器的代码。

If you prefer a video explanation about the Kalman filter, I recommend the following video series:
http://www.youtube.com/watch?v=FkCT_LV9Syk.

如果你喜欢视频介绍，我推荐下面的这个。

Note that you can not use the library if you need to represent something in a full 3D orientations, as euler angles suffer from what is called Gimbal lock you will need to use Quaternions to do that, but that is a whole nother story. For now take a look at the following page.

注意这些代码不能用于描述 3D 的全状态，因为欧拉角存在 Gimbal lock 问题，所以你最好采用例如四项式的描述方法，但这不在本文讨论范围。

This is all for know, I hope that you will find i helpful, if you do or have any questions fell free to

post a comment below – it supports LaTeX syntax as well, if you need to write equations.
If you spot any errors please let me know as well.
这是我所知道的一切，如果你有什么问题，请留言。留言支持所有 LaTex 的语法，你可以用来输入公式。
如果你发现本文的任何问题，欢迎跟我联系。