Chapter 7
Multi-sensor Integration

Under ideal conditions, the algorithms described above work perfectly. From the initial orientation plus the gyroscope data, they can first determine the orientation. And knowing orientation, they can cancel out the contribution from gravity, and—given the initial position and velocity—find the current position in space.

However, the analysis of real sensory signals is not quite that simple. Real sensory signals include a number of artifacts like offsets and drifts (Woodman 2007). And since offsets lead to velocity errors that grow linearly with time, and position errors grow quadratically (as described in Sect. 2.2.2), the analytical solutions rarely get applied directly.

But a wealth of algorithms exist, using different approaches to deal with gyroscope bias drift, inertial acceleration, and magnetic disturbances (e.g. Mahony et al. 2008; Savage 2006; Sabatini 2006; Roetenberg et al. 2007). Two main sensor fusion approaches have been proposed: stochastic filtering, often implemented in the form of an extended Kalman filter. And the so-called "complementary filtering" approaches, which fuse multiple noisy measurements from the gyroscopes, accelerometers, and magnetometers that have complementary spectral characteristics. For each measurement, the complementary filtering uses only the part of the signal frequency spectrum that contains useful information. (This is reflected in the name, *complementary filters*.) Unfortunately, due to the varying conventions used in the different publications, such as quaternions, Euler angles, rotation matrices, and rotations of objects versus rotations of coordinate systems, direct comparisons of the different approaches are often difficult.

This chapter first provides an introduction to working with uncertain data. After that the principle of Kalman filters is introduced. In the last section, an example of a complementary filter that has received much attention for the evaluation of IMU data is presented, the filter proposed by Madgwick et al. (2011).

T. Haslwanter, *3D Kinematics*, https://doi.org/10.1007/978-3-319-75277-8_7

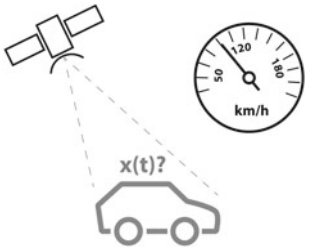

Fig. 7.1 How best to combine the information from a GPS and the speedometer, to obtain the optimal estimate of the current position?

7.1 Working with Uncertain Data

As[1] an example application, consider the problem of determining the precise location of a car (Fig. 7.1). The car can be equipped with a GPS unit that provides an estimate of the position within a few meters. The GPS estimate is likely to be noisy; readings "jump around" rapidly, though always remaining within a few meters of the real position. In addition, since the car is expected to follow the laws of physics, its position can also be estimated by integrating its velocity over time, determined by keeping track of wheel revolutions and the angle of the steering wheel. This is a technique known as "dead reckoning". Typically, the dead reckoning will provide a very smooth estimate of the car's position, but it will drift over time as small errors accumulate.

What makes this process particularly challenging is that neither the current position/velocity of the car nor the GPS measurement are 100% accurate. To prepare the mathematical ground for working with probabilities, the next section will start with an introduction on how *uncertain information* can be described mathematically.

7.1.1 Uncertain Data in One Dimension

Normal Distribution

For simplicity, we start out with a one-dimensional, uncertain piece of information: a one-dimensional position measurement. The measurement indicates a certain value μ, but also has an uncertainty σ (Fig. 7.2). The probability that the value of a measurement is correct is characterized by a so-called "probability distribution". In many cases, this probability distribution is well described by a "normal probability distribution", also called a "Gaussian probability distribution":

$$\mathcal{N}(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{7.1}$$

[1]This section is strongly based on the presentation http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/ by Timm Babb.

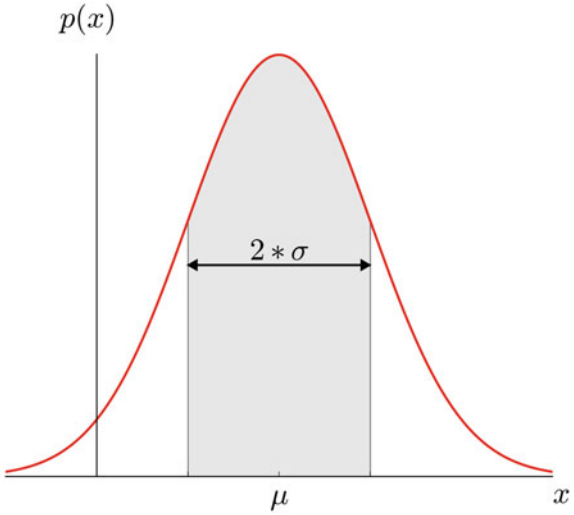

Fig. 7.2 Normal distribution, centered about μ, with a standard deviation of σ

where μ is the mean or expected value of the distribution, and σ is the standard deviation (σ^2 is the variance).

Combination of Two Normal Distributions

How can one obtain the best estimate for the position of the car if one has two different measurements, in our example the prediction from the dead reckoning and the GPS measurement? Luckily, the product of two Gaussians is again a Gaussian (Fig. 7.3). Let μ_i be the best guess of the measurement i, and σ_i the corresponding standard deviation. Then, the combined probability distribution can then be obtained with

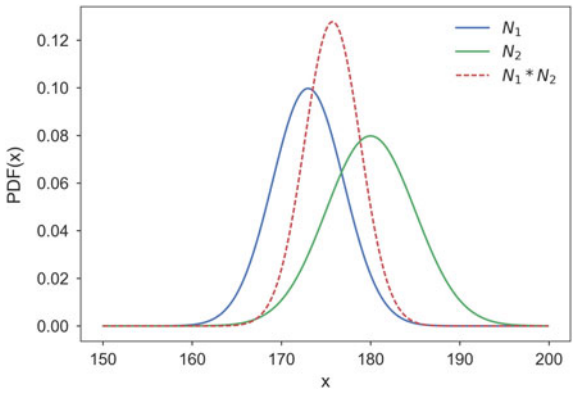

Fig. 7.3 The product of two normal probability distributions (solid lines) is again normally distributed (dotted line)

$$\mathcal{N}(x \mid \mu_0, \sigma_0) \cdot \mathcal{N}(x \mid \mu_1, \sigma_1) \sim \mathcal{N}(x \mid \mu', \sigma') \tag{7.2}$$

Substituting Eq. (7.1) into Eq. (7.2), and normalizing the resulting distribution gives

$$\mu' = \mu_0 + \frac{\sigma_0^2(\mu_1 - \mu_0)}{\sigma_0^2 + \sigma_1^2} \tag{7.3}$$

$$\sigma'^2 = \sigma_0^2 - \frac{\sigma_0^4}{\sigma_0^2 + \sigma_1^2}. \tag{7.4}$$

With

$$\mathbf{k} = \frac{\sigma_0^2}{\sigma_0^2 + \sigma_1^2} \tag{7.5}$$

we obtain

$$\begin{aligned} \mu' &= \mu_0 + \mathbf{k}(\mu_1 - \mu_0) \\ \sigma'^2 &= (1 - \mathbf{k})\sigma_0^2. \end{aligned} \tag{7.6}$$

The variable **k** in Eq. (7.6) corresponds to the "Kalman Gain" of the Kalman filter described in the next section, and the combination of two probability distributions is equivalent to the action *Update* in Fig. 7.8, since the information from one system is combined with that from another system.

7.1.2 Uncertain Data in Multiple Dimensions

In practice systems often have more than one dimension, i.e., they require more than one parameter to characterize the current state of the system. For the car example here, Fig. 7.4 shows 500 (hypothetical) position and velocity measurements. Each of these parameters is normally distributed, as shown by the corresponding histogram.

Plotting not the individual points, but the *probability* to find a given position/velocity measurement gives the corresponding two-dimensional Gaussian probability distribution (see Fig. 7.5).

In Fig. 7.5, position and velocity are "uncorrelated", which means that the state of one variable (e.g. position) tells us nothing about what the other (e.g. velocity) might be. The example in Fig. 7.6 shows something more interesting. There position and velocity are "correlated": the likelihood of observing a particular position depends on the current velocity.

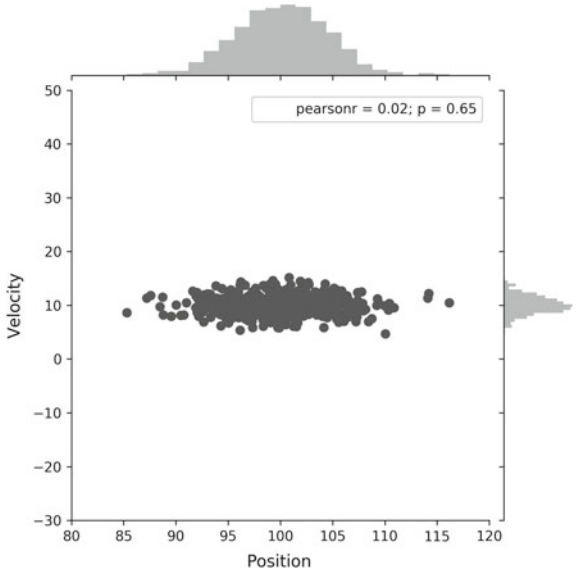

Fig. 7.4 500 samples from uncorrelated position and velocity measurements. The projections on the top and on the right show the corresponding sample histograms

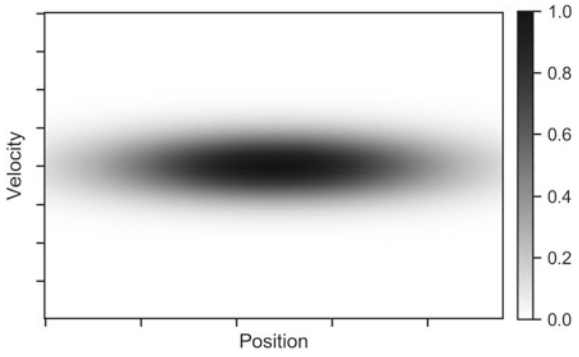

Fig. 7.5 Probability distribution of two uncorrelated variables. The colorbar on the right side shows the scale how likely it is to obtain a measurement at any given location

This kind of situation might arise if, for example, the estimate of a new position is based on an old one. If the velocity was high, the car probably moved farther, so the new position will be more distant. If the car drove slowly, it did not get as far.

This kind of relationship is really important to keep track of, because it provides more information: one measurement contains information about what the other could be. This correlation is captured by the so-called "covariance matrix" $\mathbf{\Sigma}$. Each element Σ_{ij} of the matrix quantifies the degree of correlation between the i^{th} state variable

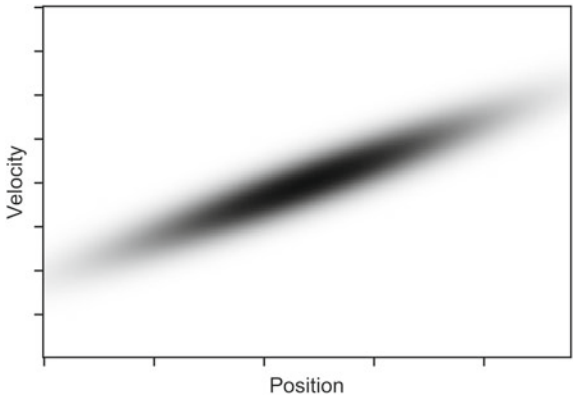

Fig. 7.6 Probability distribution of two correlated variables

and the j^{th} state variable. (Note that the covariance matrix is symmetric, which means that it does not matter if the indices i and j are exchanged.)

To make the notation more concise, let p_k be the position at time t_k, and v_k the corresponding velocity. The "state vector" describing the object is now given by the vector $\mathbf{x}$, defined as

$$\mathbf{x}_k = \begin{pmatrix} \text{position} \\ \text{velocity} \end{pmatrix}(t_k) = \begin{pmatrix} p_k \\ v_k \end{pmatrix}. \tag{7.7}$$

And the corresponding covariance matrix is

$$\mathbf{\Sigma}_k = \begin{bmatrix} \Sigma_{pp} \Sigma_{pv} \\ \Sigma_{vp} \Sigma_{vv} \end{bmatrix}. \tag{7.8}$$

The diagonal elements of the covariance matrix correspond to the *variance* of the position and the velocity, respectively. And the off-diagonal elements quantify the *correlation* between the two parameters.

The combination of uncertain multidimensional measurements with uncertain state expectations requires matrix versions of Eqs. (7.5) and (7.6).

If $\mathbf{\Sigma}$ is the covariance matrix of a Gaussian blob, and the vector $\boldsymbol{\mu}$ its mean along each axis, then the Kalman gain matrix $\mathbf{K}$ is

$$\mathbf{K} = \mathbf{\Sigma}_0 \cdot (\mathbf{\Sigma}_0 + \mathbf{\Sigma}_1)^{-1}, \tag{7.9}$$

and the new mean $\boldsymbol{\mu}'$ and the new covariance matrix Σ' are (Fig. 7.7)

$$\begin{aligned} \boldsymbol{\mu}' &= \boldsymbol{\mu}_0 + \mathbf{K} \cdot (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) \\ \mathbf{\Sigma}' &= (1 - \mathbf{K}) \cdot \mathbf{\Sigma}_0. \end{aligned} \tag{7.10}$$

The subscripts in Eqs. (7.9) and (7.10) refer to the first and second set of measurements.

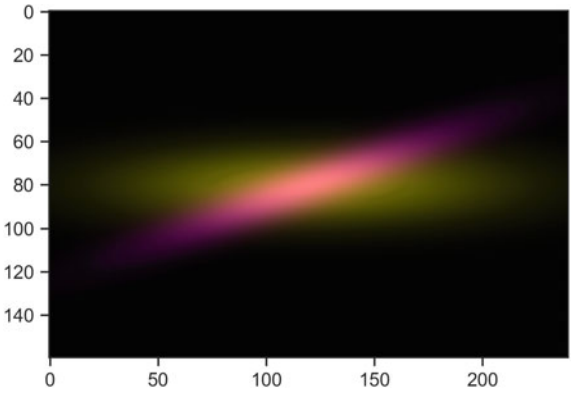

Fig. 7.7 The combination from the information from two higher dimensional probability distributions reduces the uncertainties in our estimates: the bright area indicates where both measurement_0 (magenta) and measurement_1 (yellow) agree

7.2 Kalman Filter

7.2.1 Idea Behind Kalman Filters

Kalman[2] filtering is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone, by using Bayesian inference and estimating a joint probability distribution over the variables for each time frame. The elegant feature of the Kalman filter is that the uncertainty in the data is taken into consideration, and the maximum amount of knowledge is extracted from the given information. The filter is named after Rudolf E. Kálmán (1930–2016), one of the primary developers of its theory.

The Kalman filter has numerous applications in technology. A common application is for guidance, navigation, and control of vehicles, particularly aircraft and spacecraft. Furthermore, the Kalman filter is a widely applied concept in time series analysis used in fields such as signal processing and econometrics. Kalman filters are also one of the main topics in the field of robotic motion planning and control, and they are sometimes included in trajectory optimization. In neuroscience, the Kalman filter has found use in modeling the central nervous system's control of movement. Due to the time delay between issuing motor commands and receiving sensory feedback, use of the Kalman filter provides the needed model for making estimates of the current state of the motor system and issuing updated commands (Wolpert and Ghahramani 2000).

The algorithm works in a two-step process (see Fig. 7.8). In the *Prediction* step, the Kalman filter produces estimates of the current state variables, along with their

[2]This section is taken from https://en.wikipedia.org/wiki/Kalman_filter

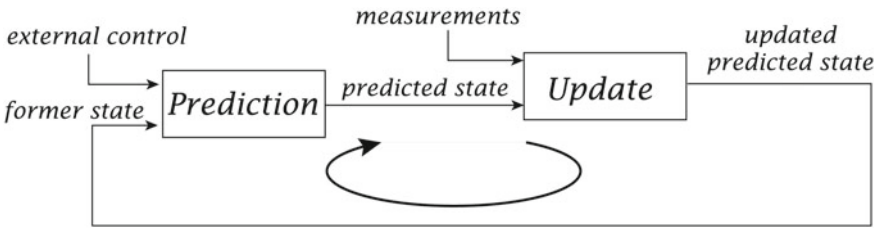

Fig. 7.8 Simplified diagram describing the iterative update of a system

uncertainties. Once the outcome of the next measurement (necessarily corrupted with some amount of error, including random noise) is observed, the state estimates are combined with the measurements in an *Update* step using a weighted average, with more weight being given to estimates with higher certainty. The algorithm is recursive. It can run in real time, using only the present input measurements and the previously calculated state and its uncertainty matrix; no additional past information is required.

The Kalman filter does not require any assumption that the errors are Gaussian. However, the filter yields the exact conditional probability estimate in the special case when all errors are Gaussian distributed.

For Kalman filters, the underlying model is a Bayesian model similar to a "hidden Markov model", but where the state space of the latent variables is continuous and where all latent and observed variables have Gaussian distributions.

Example Application

In the example with the car on a highway, the parameters (position/velocity) describe the current state of this system, and are sampled at equal time increments Δt. Knowing the *former state* of the system (position/velocity), and the *external control* (position of the accelerator pedal), one can calculate the *predicted state* of the system Δt seconds later (Fig. 7.8). Since the car may have encountered, e.g., a steep, bad section of the road, the *predicted state* will not match up exactly with the *measurements* (e.g., the position/velocity information from the GPS signals). In order to get the best possible estimate of the current position, these two pieces of information are integrated in the *Update*, to get the *updated predicted state*. This is the new starting point, and the process begins all over again.

In this example, the Kalman filter can be thought of as operating in two distinct phases: *Prediction* and *Update*. In the prediction phase, the car's old position will be modified according to the physical laws of motion (the dynamic or "state transition" model) plus any changes produced by the accelerator pedal and steering wheel. Not only will a new position estimate be calculated, but a new covariance will be calculated as well, providing information about the uncertainty of the car's position. Perhaps the covariance is proportional to the speed of the car because we are more uncertain about the accuracy of the dead reckoning position estimate at high speeds but very certain about the position estimate when moving slowly. Next, in the update

phase, a measurement of the car's position is taken from the GPS unit. Along with this measurement comes some amount of uncertainty, and its covariance relative to that of the prediction from the previous phase determines how much the new measurement will affect the updated prediction. Ideally, if the dead reckoning estimates tend to drift away from the real position, the GPS measurement should pull the position estimate back toward the real position but not disturb it to the point of becoming rapidly changing and noisy.

7.2.2 State Predictions

How can state predictions, corresponding to the box *Prediction* in Fig. 7.8, be implemented? In the following first the equations without *external control* will be considered, and then external input will be added. In order to facilitate the overview, and the correspondence between the equations and Figs. 7.9 and 7.10, the elements in the following equations will use the same colors as the corresponding elements in the figures.

Without External Control

In the example the current position/velocity is known at time t_{k-1} (Fig. 7.9, left). If one wants to know the best estimate for the position/velocity at time t_k (Fig. 7.9, right), one can write that down as

$$\begin{aligned} p_k &= p_{k-1} + \Delta t * v_{k-1} \\ v_k &= v_{k-1}. \end{aligned} \tag{7.11}$$

Using vector and matrix notation, this can be written as

$$\mathbf{x}_k = \mathbf{F}_k \cdot \mathbf{x}_{k-1}. \tag{7.12}$$

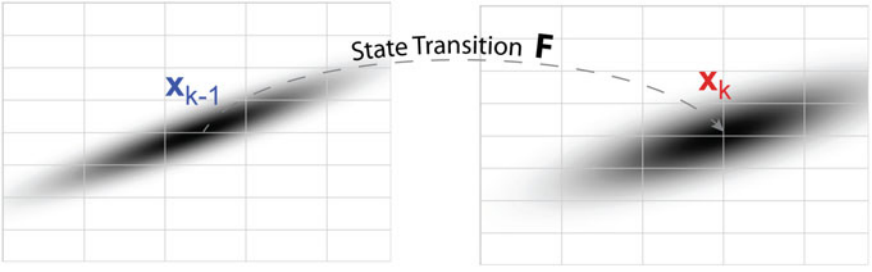

Fig. 7.9 Knowledge of the previous state (left, $\mathbf{x}_{k-1}$) and the state-transition matrix $\mathbf{F}$ allows calculation of the new state (right, $\mathbf{x}_k$)

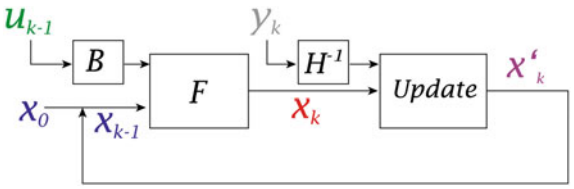

Fig. 7.10 Simplified scheme describing the iterative update of a system, taking external factors $\mathbf{u}_{k-1}$ and measurements $\mathbf{y}_k$ into consideration

Borrowing the terminology from the theory of control systems

- $\mathbf{x}_k$ is the *state* at time k, containing all the parameters required to describe the current state of system. (In our example, these are position and velocity, and $\mathbf{x}_0$ provides the initial state vector.)

$$\mathbf{x}_k = \begin{pmatrix} p_k \\ v_k \end{pmatrix}.$$

- $\mathbf{F}$ is the *state transition model*, in our case given by the matrix

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

  which is applied to the previous state $\mathbf{x}_{k-1}$ in order to get the estimated new state $\mathbf{x}_k$.
- In the context of Kalman filters, the covariance matrix for the state $\mathbf{x}_{k-1}$ is typically indicated with $\mathbf{P}_{k-1}$

$$\mathbf{P} = \begin{bmatrix} \sigma_p^2 & \sigma_{pv} \\ \sigma_{vp} & \sigma_v^2 \end{bmatrix}.$$

If one knows how the *state transition* affects each individual point (Fig. 7.9), one can also calculate how the probability distribution develops. Elementary linear algebra gives

$$\mathbf{P}_k = \mathbf{F}_k \cdot \mathbf{P}_{k-1} \cdot \mathbf{F}_k^T. \tag{7.13}$$

With External Control

If accelerator or brakes are activated, the car will no longer move smoothly forward, but also undergo an acceleration a. In that case, the new position/velocity are given by

$$\begin{aligned} p_k &= p_{k-1} + \Delta t * v_{k-1} + \frac{1}{2} * a * \Delta t^2 \\ v_k &= v_{k-1} + a * \Delta t. \end{aligned} \tag{7.14}$$

Writing these equations in matrix form one obtains

$$\begin{aligned} \mathbf{x}_k &= \mathbf{F}_k \cdot \mathbf{x}_{k-1} + \begin{pmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{pmatrix} * a \\ &= \mathbf{F}_k \cdot \mathbf{x}_{k-1} + \mathbf{B}_k \cdot \mathbf{u}_k \end{aligned} \tag{7.15}$$

where:

- $\mathbf{B}$ is called the *control matrix*
- the *control vector* $\mathbf{u}_k$ (here a simple scalar a) characterizes the external input.

Also the external control is not 100% precise, but also has some variability. Denoting this external variability with $\mathbf{Q}$, Eq. (7.13) turns into

$$\mathbf{P}_k = \mathbf{F}_k \cdot \mathbf{P}_{k-1} \cdot \mathbf{F}_k^T + \mathbf{Q}_k. \tag{7.16}$$

In words, with the knowledge of the covariance matrix for the previous state vector, $\mathbf{P}_{k-1}$, the state transition matrix $\mathbf{F}_k$, and the variability in external control, $\mathbf{Q}_k$, the covariance matrix for the new state can be calculated.

To conclude the update cycle, one final step has to be considered: external *measurements* (see Fig. 7.10, $\mathbf{y}_k$):

- $\mathbf{y}_k$, the observation at time k, is combined with the estimated state $\mathbf{x}_k$ to form a new estimate, $\mathbf{x}'_k$.
- With this new best estimate, the whole process is then repeated.

The next section shows how external measurements are included in the equations.

7.2.3 Measurements and Kalman Equations

Several sensors might provide information about the state of the system. For the time being it does not matter what they measure; perhaps one reads position and the other reads velocity. Each sensor says something indirect about the state. In other words, the sensors operate on a state and produce a set of readings. In the context of Kalman filters, it is typically assumed that the expected sensor signal is related to the state estimate $\mathbf{x}_k$ through a linear transformation

$$\mathbf{sensor} = \mathbf{H}_k \cdot \mathbf{x}_k. \tag{7.17}$$

And the uncertainty in the state estimate $\mathbf{P}$ propagates into an uncertainty in the sensor space via

$$\Sigma_{\text{expected}}^{\text{sensor}} = \mathbf{H}_k \cdot \mathbf{P}_k \cdot \mathbf{H}_k^T. \tag{7.18}$$

In Kalman equations, the mean sensor signal is typically labelled $\mathbf{z}_{k-1}$, and the *sensor noise* (i.e. the covariance of the sensor readings) with $\mathbf{R}_{k-1}$.

Putting It All Together

We have two distributions: the predicted measurement with

$$(\mu_0, \mathbf{\Sigma}_0) = (\mathbf{H}_k \cdot \mathbf{x}_k, \mathbf{H}_k \cdot \mathbf{P}_k \cdot \mathbf{H}_k^T)$$

and the observed measurement with

$$(\mu_1, \mathbf{\Sigma}_1) = (\mathbf{z}_k, \mathbf{R}_k).$$

Plugging these into Eq. (7.10) to find their overlap, we get

$$\begin{aligned} \mathbf{H}_k \cdot \mathbf{x}'_k &= \mathbf{H}_k \cdot \mathbf{x}_k + \mathbf{K} \cdot (\mathbf{z}_k - \mathbf{H}_k \cdot \mathbf{x}_k) \\ \mathbf{H}_k \cdot \mathbf{P}'_k \cdot \mathbf{H}_k^T &= (1 - \mathbf{K}) \cdot \mathbf{H}_k \cdot \mathbf{P}_k \cdot \mathbf{H}_k^T \end{aligned}. \tag{7.19}$$

And from Eq. (7.9), the Kalman gain is

$$\mathbf{K} = \mathbf{H}_k \cdot \mathbf{P}_k \cdot \mathbf{H}_k^T \cdot \left(\mathbf{H}_k \cdot \mathbf{P}_k \cdot \mathbf{H}_k^T + \mathbf{R}_k\right)^{-1}. \tag{7.20}$$

We can eliminate $\mathbf{H}_k$ off the front of every term in Eqs. (7.19) and (7.20) (note that one is hiding inside $\mathbf{K}$), and an $\mathbf{H}_k^T$ off the end of all terms in the equation for $\mathbf{P}'$ (Fig. 7.11).

$$\begin{aligned} \mathbf{x}'_k &= \mathbf{x}_k + \mathbf{K}' \cdot (\mathbf{z}_k - \mathbf{H}_k \cdot \mathbf{x}_k) \\ \mathbf{P}'_k &= \mathbf{P}_k - \mathbf{K}' \cdot \mathbf{H}_k \cdot \mathbf{P}_k \end{aligned} \tag{7.21}$$

$$\mathbf{K}' = \mathbf{P}_k \cdot \mathbf{H}_k^T \cdot (\mathbf{H}_k \cdot \mathbf{P}_k \cdot \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \tag{7.22}$$

... giving us the complete equations for the *Update* step.

And that's it! $\mathbf{x}'$ is the new best estimate, and can be fed (along with $\mathbf{P}'_k$) back into another round of *Prediction* and *Update*.

These equations represent any linear system accurately. And for an implementation, of all the math above only Eqs. (7.13) and (7.16), (7.21), and (7.22) are required.

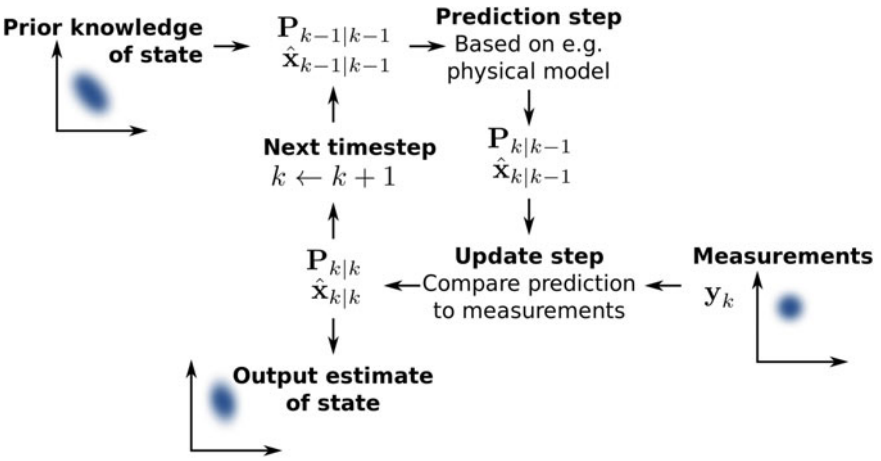

Fig. 7.11 The Kalman filter keeps track of the estimated state of the system and the variance or uncertainty of the estimate. The estimate is updated using a state transition model and measurements. $\mathbf{x}_{k|k-1}$ denotes the estimate of the system's state at time step k before the kth measurement $\mathbf{y}_k$ has been taken into account; $\mathbf{P}_{k|k-1}$ is the corresponding uncertainty (from Wikipedia)

7.2.4 Kalman Filters with Quaternions

For nonlinear systems the math gets more complicated, and methods like *extended Kalman filters* and *unscented Kalman filters* have to be used. These work in principle by linearizing the predictions and measurements about their mean. These extensions are particularly important for 3-D kinematics, since there the underlying algorithms are clearly nonlinear: for example, expressed with quaternions, the combination of two rotations requires a cross product calculation (Eq. 4.6).

The quaternion-based extended Kalman filter developed by Yun and Bachman for human body motion tracking (Yun and Bachmann 2006) is implemented in the `scikit-kinematics` package `imus`.

7.3 Complementary Filters

The "complementary filter" is a somewhat different, simple estimation technique that was developed in the flight control industry to combine measurements (Higgins 1975). This filter is actually a steady-state Kalman filter for a certain class of filtering problems. It does not consider any statistical description of the signal, but instead considers how x and y, two noisy measurements of some signal z, can be used to produce an estimate of the signal, $\hat{z}$, if the filter characteristics of the two measurements complement each other (see Fig. 7.12).

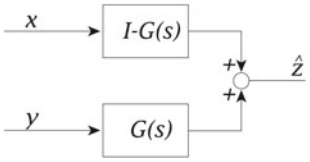

Fig. 7.12 In a "complementary filter", the filter characteristics of two measurements of a signal z, here labeled x and y, complement each other. Thus the filter outputs can be combined to achieve a better estimate of the original signal

An example of a *complementary filter* that takes the kinematic properties of 3-D orientation into consideration is the approach described by (Madgwick et al. 2011). That approach makes use of the fact that human movements cannot contain linear accelerations lasting more than a few seconds. This allows the construction of analysis algorithms with advantages over Kalman filter approaches.

7.3.1 Gradient Descent Approach

The algorithm for sensor integration developed by Madgwick is computationally very efficient (Madgwick et al. 2011). It uses a quaternion representation, allowing accelerometer and magnetometer data to be used in a "gradient descent algorithm" to compute the direction of the gyroscope measurement error as a quaternion derivative. The algorithm achieves levels of accuracy matching that of the Kalman-based algorithms. Open-source implementations of this algorithm are available for C, C#, and Matlab,[3] and for Python.[4]

The idea behind the *gradient descent* method is illustrated in Fig. 7.13: on an "error-surface", walk in the steepest downward direction (="gradient") in order to get to bottom most quickly.

The smart algorithm by Madgwick uses the following assumptions:

- On average, gravity points downward. This provides the direction of the space-fixed z-axis. And the horizontal component of the local magnetic field can be taken as the direction of the x-axis.
- Knowing the 3-D angular velocity ω_{t-1}, one can use a modification of Eq. (5.13),

$$\frac{\mathrm{d}\tilde{\boldsymbol{q}}}{\mathrm{d}t} = \frac{1}{2}\tilde{\boldsymbol{\omega}} \circ \tilde{\boldsymbol{q}} \tag{7.23}$$

to get an estimate of the current orientation

$$\tilde{\boldsymbol{q}}_{\omega,t} \approx \tilde{\boldsymbol{q}}_{\omega,t-1} + \left(\frac{\mathrm{d}\tilde{\boldsymbol{q}}}{\mathrm{d}t}\right)_{\omega,t} * \Delta t. \tag{7.24}$$

[3] http://x-io.co.uk/open-source-imu-and-ahrs-algorithms

[4] http://work.thaslwanter.at/skinematics/html/imus.html

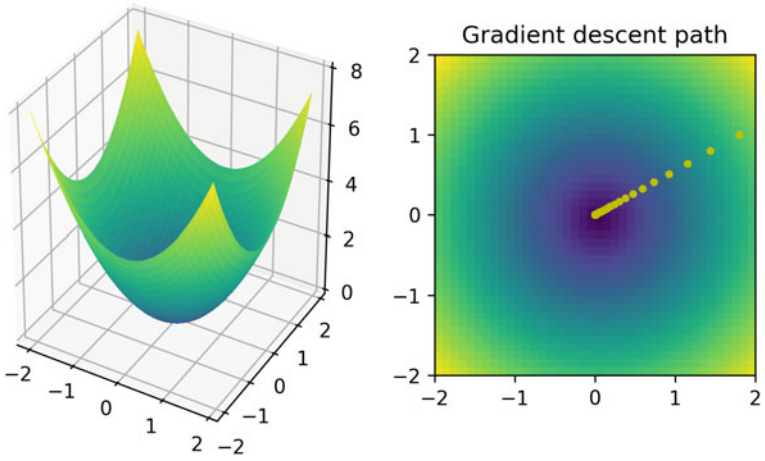

Fig. 7.13 (Left) Magnitude of deviation from gravity, forming a quadratic bowl. (Right) The yellow dots indicate the stepwise "gradient descent" to the bottom of a quadratic bowl, here starting at $(x, y) = (1.8/1.0)$

From the point of view of complementary filters, this provides the high-frequency input.
In Eq. (7.23), one has to be very careful with the sequence: Madgwick defines $\tilde{\boldsymbol{q}}$ to represent the orientation of the earth relative to the sensor, which results in the inversion of the sequence in Eq. (7.23)!

- Angular velocity sensors in IMUs typically show a substantial amount of drift and noise. In order to compensate for the resulting cumulative error, Madgwick combines this orientation estimate with a second estimate of orientation: interpreting the readout of the accelerometers as gravity, and knowing the direction of the local magnetic field, provides in combination a second orientation estimate, in addition to the one in Eq. (7.24). Here Madgwick proposes—based on investigations of the underlying kinematics—to perform *the first step of a gradient descent* into that direction, with a fixed magnitude which is set to compensate the typical gyroscope errors. Note that the step width for this step was adjusted in (Madgwick et al. 2011) to optimize the filter properties for the frequently used *XSens* sensor. From the point of view of complementary filters, this compensates for low-frequency drifts and errors.
- Since the local surroundings can have a significant effect on the local magnetic field, Madgwick's algorithm is modified such that this can only lead to errors in the heading direction, and no tilt errors.

The decision which filter is "best" for a given application depends on the specific application requirements. The filter by Madgwick was designed specifically for real-time implementations of human movement recordings. For other applications, for example, aerospace or in the automotive area, assumptions inherent in the approach by Madgwick may not hold.