Using the Kalman Filter to track Human Interactive Motion — Modelling and Initialization of the Kalman Filter for Translational Motion —

Markus Kohler *

Abstract

Based on an example of translational motion, this report shows how to model and initialize the Kalman Filter. Basic rules about physical motion are introduced to point out, that the well-known laws of physical motion are a mere approximation. Hence, motion of non-constant velocity or acceleration is modelled by additional use of white noise. Special attention is drawn to the matrix initialization for use in the Kalman Filter, as, in general, papers and books do not give any hint on this; thus inducing the impression that initializing is not important and may be arbitrary. For unknown matrices many users of the Kalman Filter choose the unity matrix. Sometimes it works, sometimes it does not. In order to close this gap, initialization is shown on the example of human interactive motion. In contrast to measuring instruments with documented measurement errors in manuals, the errors generated by vision-based sensoring must be estimated carefully. Of course, the described methods may be adapted to other circumstances.

Keywords
motion, tracking, Kalman Filter.

*Universität Dortmund, Informatik VII, D-44221 Dortmund, Germany
e-mail: kohler@ls7.informatik.uni-dortmund.de

1 Introduction

Computer vision has found growing interest during the last few years. One reason may be the continuously falling expenses of hardware for image grabbing and processing. If image processing deals with movable objects, it is a good choice to take the Kalman Filter for predicting the motion.

This report emphasizes on the example of translational motion, how the Kalman Filter should be modelled and how it should be initialized. Basic rules about physical motion are introduced. The well-known laws of physical motion are an approximation only. They only hold for motion with constant velocity or constant acceleration. If the motion, however, comes from human interaction, there will be continuous change in acceleration. Hence, motion with non-constant velocity or acceleration is modelled by using white noise. A special interest is to show how the matrices of the Kalman Filter should be determined and initialized.

This report only considers translational motion. The motion, that shall be tracked and predicted, is the motion of human body parts like it appears in gesture recognition systems [ZYK95, SKZ95, ARG97, Koh96, Koh97], for example. That kind of motion is neither of constant velocity nor constant acceleration. The Kalman Filter allows to model the acceleration or its derivative as white noise, which makes sense for human motion. In this case the Kalman Filter is an ideal predictor. It operates using the Maximum Likelihood estimation which yields in better results than least square methods [Bro83, page 242–245]. The efficiency of the prediction is shown in the ARGUS system [ARG97, Koh96, Koh97]. There the ideal motion is modelled with constant velocity and the arbitrary acceleration is considered as white noise. This keeps the system small, i. e. the matrices have small dimension. Further, as image recognition is used in the ARGUS system the measurement directly comes from a vision system. This is a two dimensional measurement and heavily depends on segmentation and the overall brightness of the surrounding. You cannot get any error boundaries from manuals but only from estimations.

There are some considerable reasons to use prediction in computer vision:

1. The time consuming and uncertain identification of the objects may be dropped due to tracking and prediction. Once the objects are identified, the prediction tells where they will appear in the next frames. As long as the objects do not collide and the frame rate is high enough, the predictor will keep right.

2. Using prediction may considerably reduce computational cost. If the predictor tells the position of the objects, only that certain parts of the image must be processed. There is no need to scan the whole frame. Hence, tracking minimizes the number of pixel operations.

3. If prediction is used to track an object, there is a predicted area in the next frame where the object should come to lie. Only that small area will be processed. Noise and other objects outside that area do not influence or even disturb the processing.

4. If the predicted area is segmented, less colours or grey-levels occur in the histograms. The histograms are more balanced and more bi-modal.

5. Tracking allows local segmentation. The threshold of a histogram based segmentation adapts easily when the object moves to an area of different illumination.

Thus tracking and prediction of motion speeds up the image processing, improves the independence of varying illumination and avoids most reactions on noise.

Generally, prediction works as described here briefly: For reasons of computational cost, a few designated points are tracked and predicted. Of course, it is possible to track all the boundary points of an object and even more. But generally, the center of gravity or some other designated points will be tracked. If several points of one object are tracked and the modelling of the motion allows to track each single point separately, this should be done to minimize computational cost. Let us consider the prediction for an object's single point (e. g. the center of gravity of an object). After a certain initialization and in every step the predictor tells a new position of that certain point. Either in a surrounding of that prediction, the point must be searched directly or the whole object should be searched in a surrounding around the prediction, that must be large enough to cover the whole object, even if the prediction might differ from the object's real position. Thus, this prediction allows to limit the search space. Only a surrounding of the point or the surrounding of the object must be considered instead of the whole possible space. Now the object or the object's point must be detected and measured. This real measurement is introduced into the Kalman Recursion and influences the blending factor (Kalman Matrix). The Kalman Matrix "decides" whether the measurement will dominate the estimate or the estimate will be weighted more in the next step.

As the literature keeps silent about initializing the Kalman Filter, there is a long discussion about this subject here. It is rather important to mention that if either the matrix $\mathbf{Q}_k$ or $\mathbf{R}_k$ (see Section 2.4) are disarranged once, they keep disarranged, will not be updated and the Kalman Process might give bad predictions. The matrix $\mathbf{P}_k$, however, and the state vector, will be updated during the recursion. Their initialization is not as important as $\mathbf{Q}_k$ and $\mathbf{R}_k$.

The next Section shows the details about tracking with the Kalman Filter. It is split into the Subsection 2.1, informing about basics in physics, Subsection 2.2, explaining distributions and the Fourier Transform of a white noise spectral density function, Subsection 2.3, modelling the translational motion with linear equations according to the requirements of the Kalman Filter, Subsection 2.4, suggesting how to initialize the covariance matrices and Subsection 2.5 showing the Kalman Recursion.

2 Tracking and Predicting Motion

The *Discrete Kalman Filter* [Bro83] is useful to predict positions of points in the two dimensional image. We start to model a motion of a single point with the Kalman Filter. To derive an appropriate model for the Kalman Filter the physical background is mentioned first. A physical motion can be described by a position vector $\mathbf{s}(t)$ that depends on the time t. The position vector may have from one to three dimensions depending whether the motion is along a straight line, in a plane or in a three dimensional space. As motion prediction is applied on images, the vector

used here will have two dimensions. At this state, however, it is irrelevant. Just remember that $\mathbf{s}(t)$ might be a vector or a scalar. The fact, that $\mathbf{s}(t)$ might be a vector, is marked by bold letters. All matrices and vectors are also marked bold and matrices are, in contrary to vectors, in capitals. All equations should be regarded carefully, as matrices might imply tensor products. Random variables are always indicated by X or by $\mathbf{X}$, if the random variable has vector type. Additional indices show up, what the random variable represents. The index a is commonly used for acceleration, v for the velocity, s for the position and e for error. A second index may show the co-ordinate x, y.

In the following section basic rules about physics are introduced and there is a first hint, how the motion will be modelled. This leads to the white noise, discussed in the succeeding section. The section Discrete Time Model of the Kalman Filter will explain how to model and calculate the recursion. A special interest is to show how the matrices should be initialized. Generally papers and books keep silent about initialization. This gives the impression that initializing is not important and may be arbitrary. Users of the Kalman Filter often choose the unity matrix for the unknown matrices. Often it works, but sometimes it does not. On behalf of closing this gap, one way of initialization is shown in Section 2.4 for a human translational motion. Of course, this must be adapted to other circumstances.

2.1 Physics

Let $\mathbf{s}(t)$ denote the time variant position of a single moving point. Derivating $\mathbf{s}(t)$ yields the Taylor series

$$\mathbf{s}(t) = \mathbf{s}(0) + \dot{\mathbf{s}}(0)t + \frac{1}{2}\ddot{\mathbf{s}}(0)t^2 + \dots \tag{1}$$

where $\dot{\mathbf{s}}(0)$ indicates the first, $\ddot{\mathbf{s}}(0)$ the second derivative of $\mathbf{s}$ at initial time $t = 0$ etc. The Taylor series is generally reduced to

$$\mathbf{s}(t) = \mathbf{s}(0) + \dot{\mathbf{s}}(0)t + \frac{1}{2} \cdot \ddot{\mathbf{s}}(0) \cdot t^2$$

in physics. It is well known that $\dot{\mathbf{s}}(0) = \mathbf{v}(0)$ is the velocity at time $t = 0$ and $\ddot{\mathbf{s}}(0) = \mathbf{a}(0)$ is the acceleration at the initial time. However, this is a rather improper notation of a translational motion. It only holds for very small t, due to generally bad convergence of Taylor series, if $\mathbf{v}(0)$ or $\mathbf{a}(0)$ are not constant. If the acceleration is constant all time, the third derivative of $\mathbf{s}(t)$ will be zero. What does it mean, that the acceleration is constant? It means, that the force F, moving the object of mass m, is constant, because $F = m \cdot a$, with acceleration a. This may be true for many vehicles and missiles. But considering human motion, forces keep changing continuously. Only if the acceleration is constant, the well-know law follows

$$\mathbf{s}(t) = \mathbf{s}(0) + \mathbf{v}(0) \cdot t + \frac{1}{2} \cdot \mathbf{a}(0) \cdot t^2 \ . \tag{2}$$

If velocity is constant, $\ddot{\mathbf{s}}(t) = \dot{\mathbf{v}}(t) = \mathbf{a}(t)$ is zero and Equation (2) may be written as

$$\mathbf{s}(t) = \mathbf{s}(0) + \mathbf{v}(0) \cdot t \ . \tag{3}$$

Probably motion of technical systems can be considered to have constant velocity or constant acceleration after some initial time. However, no real object can start with acceleration zero and end in a constant positive acceleration without continuously changing the acceleration. Especially, human motion will scarcely be of constant velocity or acceleration.

Thus the motion is considered to be the superposition of an ideal basic motion with, for example, constant velocity and white noise. The white noise illustrates the acceleration that is (highly) time varying. Such a model needs to be written down with differential equations. That is done in section 2.3. Before, the expression "white noise" will be explained.

2.2 White Noise

White noise is defined to be a stationary random process having a constant spectral density function. The term "white" is an obvious carryover from optics where white light is light containing all visible frequencies. Let $X_a(t)$ be a random variable that describes the acceleration, that shall have the characteristics of white noise (random variables will be denoted by a bold or non-bold capital X with an index). The corresponding autocorrelation function of $X_a(t)$ is defined as

$$R_a(t_1, t_2) = E[X_a(t_1)X_a(t_2)] \, .$$

We may assume that the autocorrelation of $X_a(t)$ is independent from the exact time value (t_1 or t_2) but only depends on the time difference $\tau = t_2 - t_1$. This is meant by the term "stationary". If we denote t_1 as just t and t_2 as $t + \tau$, it can be written as

$$R_a(\tau) = E[X_a(t)X_a(t + \tau)] \, .$$

In [Bro83, chapter 2.9] it is shown that

$$R_a(\tau) = a\delta(\tau) \tag{4}$$

for a stationary random process having the density function $S_a(s) = a$ (here, $\delta(\tau)$ is the Dirac delta impulse, a denotes the white noise spectral amplitude). If $a = 1$, we say it is unity white noise. From Fourier theory and theory about distributions [BSG+96, page 414] it is well known, that $S_a(s) = \mathcal{F}\{R_a\}$, $\mathcal{F}\{\cdot\}$ denoting the Fourier transform. Besides, it seems to be intuitive that a random variable of white noise is not autocorrelated for $\tau \neq 0$, hence, $R_a(\tau) = 0$ for all $\tau \neq 0$. Considering a motionless object that starts to move in an arbitrary way and stops again after a time within a closed 3D space, the vector integral of the acceleration over that time interval is zero. Hence, the motion is of zero mean acceleration (any acceleration must be compensated by a negative acceleration). Thus subsampling $X_a(t)$ at discrete time points will deliver a time-wise uncorrelated zero-mean sequence (i. e. a discrete process with zero autocorrelation function except for $\tau = 0$). This is a requirement for the discrete model [Bro83, page 189]. The knowledge about white noise will be used in a later step.

2.3 Discrete Time Model of the Kalman Filter

As mentioned above the Taylor series of Equation (1) approximates the motion at most in a rather small surrounding of $t = 0$, if velocity or acceleration is not constant. A better way to describe motion is the usage of differential equations. Therefore phase variables are introduced:

$$\mathbf{p}_1 = \mathbf{s}(t), \mathbf{p}_2 = \dot{\mathbf{s}}(t), \mathbf{p}_3 = \ddot{\mathbf{s}}(t), \ldots \tag{5}$$

As we model the acceleration by the random variable, it is $\mathbf{p}_3 = \ddot{\mathbf{s}}(t) = \mathbf{X}_a(t)$. Let $\mathbf{X}_a(t)$ be a white noise process with maximum acceleration a (for all components of $\mathbf{X}_a(t)$; $\mathbf{X}_a(t)$ is a random vector containing a random acceleration for each dimension. This simplification is possible because any human motion either vertical or horizontal will have same characteristics), then $S_a(s) = a$. Knowing that $\dot{\mathbf{p}}_1 = \mathbf{p}_2, \dot{\mathbf{p}}_2 = \mathbf{p}_3, \ldots$ yields

$$\begin{pmatrix} \dot{\mathbf{p}}_1 \\ \dot{\mathbf{p}}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \end{pmatrix} + \mathbf{I} \begin{pmatrix} 0 \\ \mathbf{X}_a(t) \end{pmatrix} . \tag{6}$$

We define $\mathbf{w}(t) := (0, \mathbf{X}_a(t))^T$ and $\mathbf{I}$ is the identity matrix. The endpoint values are $\mathbf{p}_1(0) = \mathbf{s}(0), \mathbf{p}_2(0) = \mathbf{v}(0)$. Experiments with the human computer interaction system ARGUS [ARG97, Koh96, Koh97] showed, that modelling motion with constant velocity and considering acceleration as white noise (Equation (6)) is sufficient.

According [Bro83, page 189] the solution $\mathbf{x}(t)$ of Equation (6) at time t_{k+1} can be written as

$$\mathbf{x}(t_{k+1}) = \mathbf{\Phi}(t_{k+1}, t_k)\mathbf{x}(t_k) + \int_{t_k}^{t^{k+1}} \mathbf{\Phi}(t_{k+1}, \tau)\mathbf{G}(\tau)\mathbf{w}(\tau)d\tau \ ,$$

while $\mathbf{G}(\tau) = \mathbf{I}$. The vector $\mathbf{x}(\cdot) = (\mathbf{p}_1, \mathbf{p}_2)^T$ is composed of the actual position and the velocity (Equation (5)). An abbreviated notation using $\mathbf{x}_k := \mathbf{x}(t_k), \mathbf{\Phi}_k := \mathbf{\Phi}(t_{k+1}, t_k)$ and

$$\mathbf{w}_k := \int_{t_k}^{t^{k+1}} \mathbf{\Phi}(t_{k+1}, \tau)\mathbf{G}(\tau)\mathbf{w}(\tau)d\tau \tag{7}$$

provides

$$\mathbf{x}_{k+1} = \mathbf{\Phi}_k\mathbf{x}_k + \mathbf{w}_k \ . \tag{8}$$

As $\mathbf{x}(\cdot)$ consists of the position and velocity, $\mathbf{\Phi}_k$ is the matrix that relates $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$ in absence of a forcing function, i. e. it describes how a new position at time t_{k+1} depends on the previous position and the velocity at time t_k, and how the new velocity at time t_{k+1} relates to the previous velocity (in our case velocity is constant and, of course, independent from the position).

The vector $\mathbf{x}_k = (\mathbf{s}_k, \mathbf{v}_k)^T$ is called the *process state vector* at time t_k with the estimated position $\mathbf{s}_k := \mathbf{s}(t_k)$ and the estimated velocity $\mathbf{v}_k := \mathbf{v}(t_k)$. Generally $\mathbf{x}_k$ is a tensor vector. The *transition matrix* $\mathbf{\Phi}_k$ describes the motion of the object. With $\Delta t := t_{k+1} - t_k$ this yields the matrix

$$\mathbf{\Phi}_k = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} .$$

This especially means that

$$\mathbf{\Phi}(t_1, t_2) = \begin{pmatrix} 1 & t_1 - t_2 \\ 0 & 1 \end{pmatrix} . \tag{9}$$

For further calculations $\mathbf{w}_k$ will be evaluated now to

$$\begin{aligned} \mathbf{w}_k &= \int_{t_k}^{t^{k+1}} \begin{pmatrix} 1 & t_{k+1} - \tau \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ \mathbf{X}_a(\tau) \end{pmatrix} d\tau \\ &= \begin{pmatrix} \int_{t_k}^{t^{k+1}} (t_{k+1} - \tau)\mathbf{X}_a(\tau)\, d\tau \\ \int_{t_k}^{t^{k+1}} \mathbf{X}_a(\tau)\, d\tau \end{pmatrix} \end{aligned}$$

The following definition will be used later

$$\begin{pmatrix} \mathbf{w}_{s,k} \\ \mathbf{w}_{v,k} \end{pmatrix} := \begin{pmatrix} \int_{t_k}^{t^{k+1}} (t_{k+1} - \tau)\mathbf{X}_a(\tau)\, d\tau \\ \int_{t_k}^{t^{k+1}} \mathbf{X}_a(\tau)\, d\tau \end{pmatrix} \tag{10}$$

Of course, none of the integrals $\mathbf{w}_{s,k}$ or $\mathbf{w}_{v,k}$ can be determined as $\mathbf{X}_a(t)$ is a random variable of time t. Only statistical moments like the expected value or the variance may be determined by the distribution of $\mathbf{X}_a(t)$.

Until this point we did a lot of modelling. Now look at Equation (8): The vector $\mathbf{x}_k$ consists of the position $\mathbf{s}_k$ at time t_k in its first component and the velocity $\mathbf{v}_k$ at time t_k in its second component. The multiplication with $\mathbf{\Phi}_k$ tells that the new position, that is the first component of $\mathbf{x}_{k+1}$ without noise, arises from $\mathbf{s}_k + \Delta t \cdot \mathbf{v}_k$. This is nothing else than Equation (3) for constant velocity during the time interval $[t_k, t_{k+1}]$. The new velocity, that is the second component of the vector $\mathbf{x}_{k+1}$ is constant and identical to $\mathbf{v}_k$, if the white noise is not regarded. Indeed, $\mathbf{\Phi}_k$ just relates the position and velocity at time t_k to the new position and velocity at time t_{k+1} without regarding the white noise $\mathbf{w}_k$ of the random process.

This was the main part of modelling the Kalman Filter. For the random process a model must be introduced, what we did until now. During the recursion of the Kalman Filter each prediction will be followed by a measurement. Entering this measurement into the recursion, it will influence the next prediction. This is best described in [dP67]. So the Kalman Filter "learns" the data included by the measurement.

The measurement of the random process occurs in time at discrete points in accordance with the linear relationship

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{e}_k \tag{11}$$

where $\mathbf{z}_k$ is the *vector measurement* and $\mathbf{H}_k$ gives the ideal (noiseless) connection between the measurement and the state vector at time t_k. Hence, the Kalman

Recursion considers that the internal state vector of the model may not directly be measured. To explain this, think about a 3D motion that is observed by a video camera. If we want to measure the real 3D motion and if we accordingly modelled the Kalman Filter with a state vector containing three component position and velocity, we only get 2D measurement data (screen coordinates) of the projected position. In this case $\mathbf{H}_k$ will be the perspective projection matrix that extracts the first component of $\mathbf{x}_k$, which is the 3D position, and projects it to 2D screen coordinates. Furthermore, in Equation (11) $\mathbf{e}_k$ is the *measurement error* and also should have white characteristic.

If people decide to model the Kalman Filter and hence the state vector with the position and velocity, they sometimes try to extract both position and velocity from the measurement. Tracking on the screen, however, allows to measure the position or its projection only. It makes no sense to calculate the velocity out of the difference in position, because this is the task of the Kalman Filter. Therefore, the matrix $\mathbf{H}_k$ of Equation (11) links the position or its projection to the state vector. Mostly, $\mathbf{H}_k$ is no square matrix. In order to track the 2D motion directly on the screen, the state vector will be modelled to consist of a 2D position and a 2D velocity. Therefore, $\mathbf{H}_k$ shall only extract the position (first component) from the state vector which is done by

$$\mathbf{H}_k = \begin{pmatrix} 1 & 0 \end{pmatrix} .$$

Coming into details, the process state vector holds a 2D position and a 2D velocity modelling the position and velocity in screen coordinates. Thus, using $s_{k,x}, s_{k,y}, v_{k,x}$ and $v_{k,y}$ for the x- and y-position resp. velocity (note they are all scalar) of the process state vector, Equation (8) takes the form

$$\begin{pmatrix} s_{k+1,x} \\ s_{k+1,y} \\ v_{k+1,x} \\ v_{k+1,y} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s_{k,x} \\ s_{k,y} \\ v_{k,x} \\ v_{k,y} \end{pmatrix} + \mathbf{w}_k \tag{12}$$

Consider Equation (8) and (12) carefully to understand the tensor products that were applied. The tensor vector is

$$\mathbf{x}_k = \begin{pmatrix} \mathbf{s}_k \\ \mathbf{v}_k \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} s_{k,x} \\ s_{k,y} \end{pmatrix} \\ \begin{pmatrix} v_{k,x} \\ v_{k,y} \end{pmatrix} \end{pmatrix} \cong \begin{pmatrix} s_{k,x} \\ s_{k,y} \\ v_{k,x} \\ v_{k,y} \end{pmatrix} .$$

Thus, the tensor matrix product has the form

$$\begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \begin{pmatrix} s_{k,x} \\ s_{k,y} \end{pmatrix} \\ \begin{pmatrix} v_{k,x} \\ v_{k,y} \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} s_{k,x} + \Delta t\ v_{k,x} \\ s_{k,y} + \Delta t\ v_{k,y} \end{pmatrix} \\ \begin{pmatrix} v_{k,x} \\ v_{k,y} \end{pmatrix} \end{pmatrix}$$

$$\cong \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s_{k,x} \\ s_{k,y} \\ v_{k,x} \\ v_{k,y} \end{pmatrix}$$

Regard, that $\mathbf{w}_k$ has the same dimension as the process state vector and is also a tensor vector, i. e. $\mathbf{w}_k = (\mathbf{w}_{s,k}, \mathbf{w}_{v,k})^T$ as defined in Equation (10). Further, denoting the measurements of the x and y positions by $\mathbf{z}_k = (s^m_{k,x}, s^m_{k,y})^T$ and the x, y measurement errors by $\mathbf{e}_k = (e_{k,x}, e_{k,y})^T$ Equation (11) constitutes

$$\begin{pmatrix} s^m_{k,x} \\ s^m_{k,y} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} s_{k,x} \\ s_{k,y} \\ v_{k,x} \\ v_{k,y} \end{pmatrix} + \begin{pmatrix} e_{k,x} \\ e_{k,y} \end{pmatrix} . \tag{13}$$

Physically, forces that induce acceleration and velocities do not influence each other, if they are perpendicular (orthogonal). Any translational force, acceleration and velocity in 2D or 3D space is a superposition of two resp. three orthogonal forces, accelerations or velocities. Hence, every two x and y components of Equation (12) and Equation (13) are uncorrelated and the linear systems may be split up into two independent systems:

$$\begin{pmatrix} s_{k+1,x} \\ v_{k+1,x} \end{pmatrix} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} s_{k,x} \\ v_{k,x} \end{pmatrix} + \begin{pmatrix} w_{s,k,x} \\ w_{v,k,x} \end{pmatrix}$$

and

$$\begin{pmatrix} s_{k+1,y} \\ v_{k+1,y} \end{pmatrix} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} s_{k,y} \\ v_{k,y} \end{pmatrix} + \begin{pmatrix} w_{s,k,y} \\ w_{v,k,y} \end{pmatrix}$$

whereas the linear system for the measurement takes the form

$$\begin{pmatrix} s^m_{k,x} \end{pmatrix} = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} s_{k,x} \\ v_{k,x} \end{pmatrix} + e_{k,x}$$

and

$$\begin{pmatrix} s^m_{k,y} \end{pmatrix} = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} s_{k,y} \\ v_{k,y} \end{pmatrix} + e_{k,y}$$

This holds for $\mathbf{w}_{s,k} := (w_{s,k,x}, w_{s,k,y})^T$ and $\mathbf{w}_{v,k} := (w_{v,k,x}, w_{v,k,y})^T$. Thus, for two dimensional prediction two separate Kalman Recursions are evaluated. If several objects in the plane are tracked it is highly recommended to split up the system, because it is then more time efficient as matrix inversion and operations take less time.

As we now understood the Kalman Filter, we are able to calculate the covariance matrices on which the Kalman Recursion is based.

2.4 Determining Error Covariance Matrices

The Kalman Filter only works correctly, if

$$E[\mathbf{w}_k \mathbf{w}_i^T] = \begin{cases} \mathbf{Q}_k & i = k \\ 0 & i \neq k \end{cases} \tag{14}$$

$$E[\mathbf{e}_k \mathbf{e}_i^T] = \begin{cases} \mathbf{R}_k & i = k \\ 0 & i \neq k \end{cases}$$

$$E[\mathbf{w}_k \mathbf{e}_i^T] = 0 \quad \forall i, k \, . \tag{15}$$

Because $\mathbf{X}_a(t)$ (the random variable describing the acceleration) has a zero autocorrelation for two different time-steps, it is obvious that $E[\mathbf{w}_k \mathbf{w}_i^T] = 0$ for $i \neq k$. The sequence consisting of $\mathbf{e}_k$ is also a white (white noise), and hence, two samples at different times are uncorrelated. This means a measurement error at a time t_k should not have any influence on a later measurement. Further, an acceleration at time t_k should not influence a later acceleration. Both is true, as the user will just act without thinking what acceleration he did before.

In the last Section 2.5 about the Kalman Recursion we will see that there are three matrices that enter the Kalman Recursion. It is $\mathbf{Q}_k, \mathbf{R}_k$ and $\mathbf{P}_k$. Further an initial estimate of the state vector is required. As both, $\mathbf{P}_k$ and the state vector, will be updated during the recursion, they are less important than $\mathbf{Q}_k$ and $\mathbf{R}_k$. If either the matrix $\mathbf{Q}_k$ or $\mathbf{R}_k$ are disarranged, they keep disarranged, will not be updated and the Kalman Process might not give any good prediction.

2.4.1 Covariance Matrix of the Measurement Error

We start to determine the covariance matrix $\mathbf{R}_k$. Let $X_{e,x}(t)$ and $X_{e,y}(t)$ denote the random variable that describes the measurement error. Hence, $e_{k,x} = X_{e,x}(t_k)$ and $e_{k,y} = X_{e,y}(t_k)$ for all k. The error measurement covariance is

$$\begin{aligned} \mathbf{R}_k &= E[(X_{e,x}(t_k), X_{e,y}(t_k))^T (X_{e,x}(t_k), X_{e,y}(t_k))] \\ &= \begin{pmatrix} E[X_{e,x}(t_k) X_{e,x}(t_k)] & E[X_{e,x}(t_k) X_{e,y}(t_k)] \\ E[X_{e,x}(t_k) X_{e,y}(t_k)] & E[X_{e,y}(t_k) X_{e,y}(t_k)] \end{pmatrix} \end{aligned}$$

The random variables $X_{e,x}(t)$ and $X_{e,y}(t)$ are zero-mean and uncorrelated, i. e. that there is no dependence between an occuring error of the x and y measurement. Obviously, integrating the product $X_{e,x}(t) X_{e,y}(t)$ over some time interval positive and negative values will accumulate to zero. As the characteristics of the measurement do not change, this is true for any large enough time interval and thus $E[X_{e,x}(t_k) X_{e,y}(t_k)] = 0$. The expression $E[X_{e,x}(t_k) X_{e,x}(t_k)]$ is the square of the standard deviation of the random variable $X_{e,x}(t)$. Measurement errors will certainly occur in any physical system. Sometimes they are known and sometimes they must be estimated. Measuring the voltage of a device will depend on an error of the measurement equipment. The measurement error is usually reported in the manual as a certain percentage of the voltage. Using a vision based measurement with cameras segmentation is necessary to extract the measurable features. Usually, segmentation and thus the measurement will heavily depend on the average brightness. Shadows can throw certain pixels to the background. E. g. trying to recognize skin colour some pixel of the skin coloured object might wrongly disappear and some background might be added to the object. The object's boundary will increase or decrease. Generally, if the smallest diameter of the object appears in the image of a certain thickness, then the measurement error should keep below that value. Here, we are going to create a complete filter model for the gesture recognition example. Because gesture recognition is driven near the lower resolution boundary to achieve a bigger interaction area [Koh97, Section Assembly of ARGUS in a Home Environment], it is a prerequisite, that the measurement error should be below one pixel. Assuming that the measurement error appears to be an error of plus and minus one

pixel with the same probability, we get the square of the standard deviation

$$E[X_{e,x}(t_k)X_{e,x}(t_k)] = \frac{1}{3}\cdot((-1)^2+0^2+(+1)^2) = \frac{2}{3}\ .$$

The same holds for $E[X_{e,y}(t_k)X_{e,y}(t_k)] = 2/3$. Now the matrix can be written as

$$\mathbf{R}_k = \frac{2}{3}\cdot\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \forall k$$

More general, if the square standard deviation of the x resp. y measurement error is $E[X_{e,x}(t_k)X_{e,x}(t_k)] = \sigma^2_{e,x}$ and $E[X_{e,y}(t_k)X_{e,y}(t_k)] = \sigma^2_{e,y}$ and the x and y measurement are completely independent, the matrix is

$$\mathbf{R}_k = \begin{pmatrix} \sigma^2_{e,x} & 0 \\ 0 & \sigma^2_{e,y} \end{pmatrix} \quad \forall k\ .$$

This seems to be a good assumption for the measurement error covariance. Notice that this matrix is related on the unit "pixel", because measurement is done in pixels. The matrix $\mathbf{R}_0 = \mathbf{R}_k$ is the covariance matrix of $\mathbf{e}_k$ and is constant over the whole time. The time independence of the measurement error is often given in physical data measuring.

2.4.2 Covariance Matrix $\mathbf{Q}_k$ of the White Acceleration

Next the initial estimate of $\mathbf{Q}_k$ is described. According Equation (7) and (14) we get

$$\begin{aligned}
\mathbf{Q}_k &= E[\mathbf{w}_k\mathbf{w}_k^T] \\
&= E\left\{\left[\int_{t_k}^{t^{k+1}} \mathbf{\Phi}(t_{k+1},u)\mathbf{G}(u)\mathbf{w}(u)du\right]\left[\int_{t_k}^{t^{k+1}} \mathbf{\Phi}(t_{k+1},v)\mathbf{G}(v)\mathbf{w}(v)dv\right]^T\right\} \\
&= \int_{t_k}^{t^{k+1}}\int_{t_k}^{t^{k+1}} \mathbf{\Phi}(t_{k+1},u)\mathbf{G}(u)E[\mathbf{w}(u)\mathbf{w}(v)^T]\mathbf{G}(v)^T\,\mathbf{\Phi}(t_{k+1},v)^T du\ dv \qquad (16)
\end{aligned}$$

After Equation (6) the vector $\mathbf{w}(t) = (0, \mathbf{X}_a(t))^T$ is defined to be a 2×1 tensor vector, which yields

$$\begin{aligned}
E[\mathbf{w}(u)\mathbf{w}(v)^T] &= E\left\{\begin{pmatrix} 0\cdot\mathbf{I} & 0\cdot\mathbf{I} \\ 0\cdot\mathbf{I} & \mathbf{X}_a(u)\mathbf{X}_a(v)^T \end{pmatrix}\right\} \\
&= \begin{pmatrix} 0\cdot\mathbf{I} & 0\cdot\mathbf{I} \\ 0\cdot\mathbf{I} & E[\mathbf{X}_a(u)\mathbf{X}_a(v)^T] \end{pmatrix}
\end{aligned}$$

For the reason of superposition principle, the x and y random acceleration $X_{a,x}(\cdot), X_{a,y}(\cdot)$ are orthogonal and, hence, independent. In the special case of tracking a 2D movement (see Section 2.2 and Equation (4))

$$\begin{aligned}
E[\mathbf{X}_a(u)\mathbf{X}_a(v)^T] &= \begin{pmatrix} E[X_{a,x}(u)X_{a,x}(v)] & E[X_{a,x}(u)X_{a,y}(v)] \\ E[X_{a,y}(u)X_{a,x}(v)] & E[X_{a,y}(u)X_{a,y}(v)] \end{pmatrix} \\
&= \begin{pmatrix} a^2\delta(u-v) & 0 \\ 0 & a^2\delta(u-v) \end{pmatrix} \\
&= a^2\delta(u-v)\cdot\mathbf{I}\ .
\end{aligned}$$

Hence, with $\mathbf{G}(u) = \mathbf{G}(v) = \mathbf{I}$ we conclude

$$\begin{aligned}\mathbf{G}(u)E[\mathbf{w}(u)\mathbf{w}(v)^T]\mathbf{G}(v)^T &= \begin{pmatrix} 0\cdot\mathbf{I} & 0\cdot\mathbf{I} \\ 0\cdot\mathbf{I} & a^2\delta(u-v)\cdot\mathbf{I} \end{pmatrix} \\ &= a^2\delta(u-v)\cdot\begin{pmatrix} 0\cdot\mathbf{I} & 0\cdot\mathbf{I} \\ 0\cdot\mathbf{I} & \mathbf{I} \end{pmatrix} \qquad (17)\end{aligned}$$

where $\mathbf{I}$ is the 2×2 identity matrix. The integration of Equation (16) under consideration of (17) may be written as

$$\mathbf{Q}_k = \int_{t_k}^{t^{k+1}} \int_{t_k}^{t^{k+1}} a^2\delta(u-v) \begin{pmatrix} \mathbf{I}(t_{k+1}-u)(t_{k+1}-v) & \mathbf{I}(t_{k+1}-u) \\ \mathbf{I}(t_{k+1}-v) & \mathbf{I} \end{pmatrix} du\, dv$$

Four integrals $I_1, \ldots I_4$ representing the four components of the matrix must be solved. Therefore, theory about distributions is needed [BBNSR87], especially

$$\int_{-\infty}^{\infty} f(t)\delta(t)dt = \int_{a}^{b} f(t)\delta(t)dt = \begin{cases} f(0) & \text{for } a < 0 \text{ and } b > 0 \\ 0 & \text{for } a, b < 0 \text{ or } a, b > 0\ . \end{cases}$$

Hence, only the solutions are presented here:

$$I_1 := \int_{t_k}^{t^{k+1}} \int_{t_k}^{t^{k+1}} a^2\delta(u-v)\,\mathbf{I}\,du\,dv = a^2 \cdot \Delta t \cdot \mathbf{I} \cdot s \qquad (18)$$

$$I_2 := \int_{t_k}^{t^{k+1}} \int_{t_k}^{t^{k+1}} a^2\delta(u-v)\,\mathbf{I}\,(t_{k+1}-u)du\,dv \qquad (19)$$

$$= \frac{a^2}{2} \cdot (\Delta t)^2 \cdot \mathbf{I} \cdot s \qquad (20)$$

$$I_3 = I_2 \qquad (21)$$

$$I_4 := \int_{t_k}^{t^{k+1}} \int_{t_k}^{t^{k+1}} a^2\delta(u-v)\,\mathbf{I}\,(t_{k+1}-u)(t_{k+1}-v)du\,dv$$

$$= \frac{a^2}{3} \cdot (\Delta t)^3 \cdot \mathbf{I} \cdot s \qquad (22)$$

where $\Delta t = t_{k+1} - t_k$ as previously defined. Finally,

$$\mathbf{Q}_k = \frac{a^2\Delta t}{6} \begin{pmatrix} 2\mathbf{I}(\Delta t)^2 & 3\mathbf{I}\Delta t \\ 3\mathbf{I}\Delta t & 6\mathbf{I} \end{pmatrix} s$$

while a is the spectral amplitude of the white noise (s is the unit seconds). As the acceleration is identified as white noise, a is the amplitude of the "accelerating frequencies". This matrix may be applied for any filter models with translational motion of constant velocity and random acceleration.

According to experiments in the ARGUS project, we estimate the acceleration to

$$a < 11\frac{m}{s^2} \approx 1225\frac{pixel}{s^2}$$

for fast human movements (the assumption was, that the motion mainly takes place in the maximum interaction area of 3,45 m).

At this point it is worth to remark that Δt may be measured in seconds or in frames (that means one time unit is the time interval needed to process one frame completely). So we get three more applicable accelerations

$$\begin{aligned} a = &\quad 49\tfrac{pixel}{frame^2} && \text{for a frame rate of 5 frames per second} \\ a = &\quad 12\tfrac{pixel}{frame^2} && \text{for a frame rate of 10 frames per second} \\ a = &\quad 5\tfrac{pixel}{frame^2} && \text{for a frame rate of 15 frames per second} \end{aligned}$$

2.4.3 Covariance Matrix $\mathbf{P}_0^-$ of the Estimation Error

Before determining the covariance matrix $\mathbf{P}_0^-$, we need some further notation. We assume at this point that we have an initial estimate of the process at some point in time t_k, and that this estimate is based on all of our knowledge about the process prior to t_k. This prior (or *a priori*) estimate will be denoted as $\hat{\mathbf{x}}_k^-$ where the "hat" denotes estimate, and the "super minus" is a reminder that this is our best estimate prior to assimilating the measurement at t_k. The estimation error is then $\mathbf{x}_k - \hat{\mathbf{x}}_k^-$ and the associated error covariance matrix

$$\mathbf{P}_k^- = E[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^T]$$

The vector $\mathbf{x}_k$ is the ideal process state vector. As the real measurement of data is erroneous and the Kalman Process can only make estimates from errorness data the ideal process state vector generally differs from the estimated state vector, that is updated by the recursion. The covariance of the deviation of the unknown ideal process state vector and the calculated process state vector is held in the matrix $\mathbf{P}_k^-$.

As $\mathbf{P}_k^-$ will be updated during the Kalman Recursion it is sufficient to determine $\mathbf{P}_0^-$. Generally, this is a difficult task. We can only refer to a special application, that is the ARGUS system [ARG97, Koh96, Koh97]. Even there, it is still rather difficult to get an estimation for the initial value of $\mathbf{P}_0^-$.

In the Kalman Model described by Section 2.3, the matrix $\mathbf{P}_0^-$ takes the form

$$\mathbf{P}_0^- = \begin{pmatrix} \sigma_{s,s}^2 & \sigma_{s,v}^2 \\ \sigma_{s,v}^2 & \sigma_{v,v}^2 \end{pmatrix} .$$

The value $\sigma_{s,s}$ is the standard deviation of the position estimation error. Further, $\sigma_{s,v}^2 = E[(\mathbf{s}_k - \hat{\mathbf{s}}_k^-)(\mathbf{v}_k - \hat{\mathbf{v}}_k^-)^T]$, because $\mathbf{x}_k = (\mathbf{s}_k, \mathbf{v}_k)^T$. Over some time $E[\mathbf{s}_k - \hat{\mathbf{s}}_k^-] = E[\mathbf{v}_k - \hat{\mathbf{v}}_k^-] = 0$. We assume, that the position and velocity estimation errors are independent and uncorrelated, even if there is no soundness reason. This leads to $\sigma_{s,v} = 0$. The standard deviation of the velocity estimation error $\sigma_{v,v}$ and $\sigma_{s,s}$ are discussed later.

As the tracking process in ARGUS starts as soon as motion is detected, the position of the motion (its center of gravity) may be taken as a first estimate for the position component of the process state vector $\hat{\mathbf{x}}_0^-$. There is no good way to determine the velocity component of $\hat{\mathbf{x}}_0^-$. There is only one thing that we know about the velocity

of an object, which is detected by means of motion recognition: The velocity is not zero, otherwise it would not be detected by motion recognition.

Motion recognition is done by difference images. Thus a fast moving object appears as two domains in the difference image. If we assume that the action mostly takes place such that the hands appear near the resolution boundary, the maximal distance between the domains is approximately

$$s = 25\ pixel \quad \text{at a frame rate of 5 frames per second,} \tag{23}$$
$$s = 6\ pixel \quad \text{at a frame rate of 10 frames per second,} \tag{24}$$
$$s = 3\ pixel \quad \text{at a frame rate of 15 frames per second,} \tag{25}$$

and the maximal velocity will be

$$v = 49\frac{pixel}{frame} \quad \text{at a frame rate of 5 frames per second,} \tag{26}$$
$$v = 12\frac{pixel}{frame} \quad \text{at a frame rate of 10 frames per second,} \tag{27}$$
$$v = 5\frac{pixel}{frame} \quad \text{at a frame rate of 15 frames per second.} \tag{28}$$

If two domains appear and, as well, if the two domains collapse, the true center of gravity of the moving object will have at most half of the maximal distance from the estimated center of gravity. The estimated center of gravity can only be set to the center of gravity of the two domains[1]. Therefore the maximal error in the position component (first component) of $\mathbf{x}_k - \hat{\mathbf{x}}_k^-$ will be $s/2$ with one of the values of s of Equation (23)–(25). Because the user does a lot more slow than fast motion, we assume that the distances are distributed according to the Gauss distribution with standard deviation $2\sigma_{s,s} = s/2$ (this means that approximately 1–6 % of the motion will be that fast, that the first estimation error is $s/2$).

Because it is impossible to measure velocity of objects with difference images and because the mean of the velocity is zero, we set the initial velocity of the process state vector $\hat{\mathbf{x}}_0^-$ to zero, even if it is very obvious that it is not zero. Hence, there is an error in that first estimation and we get a positive standard deviation $\sigma_{v,v}$. We take the same Gauss distribution that yields $2\sigma_{v,v} = v/2$. Again there is no obvious way to do this. However, it is not worth to spend too much time on it, as the Kalman Recursion will immediately update the value.

Finally, we take the covariance matrix of the estimation error as

$$\mathbf{P}_0^- = \frac{1}{16} \cdot \begin{pmatrix} s^2 & 0 \\ 0 & v^2 \end{pmatrix} .$$

The initial value of the process state vector is set to

$$\hat{\mathbf{x}}_0^- = \begin{pmatrix} \mathbf{s}_0^- \\ 0 \end{pmatrix}$$

where $\mathbf{s}_0^-$ is the center of gravity of the (two) domains of an object detected by the motion recognition.

[1] Generally it is not possible to find the exact position of the moving object from the difference image.

2.5 The Kalman Recursion

In the Kalman Recursion special symbols like the "super minus" and the "hat" are used, that are explained at the beginning of Section 2.4.3. The matrices $\mathbf{P}_0^-, \mathbf{R}_0$ and $\mathbf{Q}_0$ must be initialized previously as well as the $\hat{\mathbf{x}}_0^-$. It is important to find good values for $\mathbf{R}_k$ and $\mathbf{Q}_k$. If the values of $\hat{\mathbf{x}}_0^-$ or $\mathbf{P}_0^-$ are incorrect they will generally converge towards the expected values. Next the *Kalman Recursion* is evaluated

1. Compute gain (blending factor)

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1}$$

   The matrix $\mathbf{K}_k$ acts like a blending factor in the update estimate $\hat{\mathbf{x}}_k^-$ (see item 2.). If the measurement error is "large" (unreliable measurement), then the covariance matrix $\mathbf{R}_k$ influences $\mathbf{K}_k$ such, that some components of $\mathbf{K}_k$ get small. Thus, the influence of the measurement $\mathbf{z}_k$ in the update estimate gets small. This is discussed later.

2. Update estimate

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k \cdot \hat{\mathbf{x}}_k^-)$$

   Remember that $\mathbf{z}_k$ often has a different dimension than $\mathbf{x}_k$ and $\hat{\mathbf{x}}_k^-$ (generally less dimension). Thus, $\mathbf{z}_k - \mathbf{H}_k \cdot \hat{\mathbf{x}}_k^-$ is the difference of the real measurement and the measurable components of the process state vector. Further, the matrix $\mathbf{K}_k$ must have the dimension of $\mathbf{H}_k^T$.

3. Update *error covariance matrix*

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_k^-$$

4. Project ahead

$$\hat{\mathbf{x}}_{k+1}^- = \mathbf{\Phi}_k \hat{\mathbf{x}}_k$$

5. This yields the best estimate of the *error covariance matrix*

$$\mathbf{P}_{k+1}^- = \mathbf{\Phi}_k \mathbf{P}_k \mathbf{\Phi}_k^T + \mathbf{Q}_k \ .$$

The Kalman Recursion is documented in [Bro83, pages 195–200]. It is worth to shortly discuss the blending factor and the update estimate: Let us assume that the motion is rather inaccurate, i. e. the white noise of the model (the acceleration) and the estimation error dominate the measurement error[2], then it is $\|\mathbf{R}_k\| << \|\mathbf{Q}_k\|$ and $\|\mathbf{R}_k\| << \|\mathbf{P}_k^-\|$. To shortly explain, what happens, assume at that very point that $\mathbf{R}_k = 0$, and that we are able to measure all the components of the process state vector. Then $\mathbf{z}_k$ has the same dimension as $\mathbf{x}_k$ and $\mathbf{H}_k = \mathbf{I}$. In this case it is

$$\mathbf{K}_k = \mathbf{I}$$

[2] This means that the motion was not modelled all right. We had better taken the acceleration into the process state vector instead of considering it as white noise.

and the update estimate 2. results in

$$\begin{aligned} \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k \cdot \hat{\mathbf{x}}_k^-) \\ &= \mathbf{z}_k \ . \end{aligned} \tag{29}$$

It needs further detailed analysis (for the one dimensional case see [dP67]). But it seems obvious by this example that if the measurement error is considerably small compared to the estimation errors of the Kalman Filter, the process state vector $\hat{\mathbf{x}}_k^-$ adopts the value of the measurement.

On the other hand, if the measurement error dominates the white noise of the model (the acceleration) and the estimation error, then it is $\|\mathbf{R}_k\| >> \|\mathbf{Q}_k\|$ and thus $\|\mathbf{R}_k\| >> \|\mathbf{P}_k^-\|$. To make it easier to understand let us now assume that $\mathbf{P}_k^- = 0$. Then the Kalman Matrix is

$$\mathbf{K}_k = 0$$

and the update $\hat{\mathbf{x}}_k$ is set to $\hat{\mathbf{x}}_k^-$.

Considering a single point that is predicted with the Kalman Recursion, we get a best estimate for the next position of the point. Due to the estimation error, a surrounding of that prediction must be searched. The extension of that search environment is best determined by experiments. The larger it is, the slower will be the search. The smaller it is, the more will the measurement fail, because it does not find the predicted point in that environment. If several points of an object are tracked and predicted, also a search environment must be set around the predicted object.

3 Conclusion

This report was inspired by the fact, that the literature keeps silent about how to initialize the Kalman Filter. Most practically oriented books show how to model the Filter, but there was no book, that described the initialization on a practical example. Theoretically based books like [Bro83] are not easy to read and understand. However, they give theoretical hints, how the matrices could be initialized. This report shows the modelling of the Kalman Filter for a rather often appearing pure translational motion. Further its intention is to give a guide on the example of ARGUS [Koh97], how to solve the initialization problem and what is important about initializing the covariance matrices. Basic knowledge like the dependence of the physical laws for motion on the Taylor series was discussed as well as the white noise and its spectral density function.

References

[ARG97] Projektgruppe 277: ARGUS. Ein ergonomisches Dialogsystem zur Steuerung von technischen Systemen in Wohnbereichen mittels Gestenerkennung — Abschlußbericht. Technical report, Informatik VII, University of Dortmund, 1997.

[BBNSR87] Babovsky, Beth, Neunzert, and Schulz-Reese. *Fourieranalysis*. Teubner Verlag, 1987.

[Bro83] Robert G. Brown. *Introduction to random signal analysis and Kalman filtering*. Wiley, New York [u.a.], 1983.

[BSG+96] I. N. Bronstein, K. A. Semendjajew, G. Grosche, V. Ziegler, and D. Ziegler. *Teubner-Taschenbuch der Mathematik*, volume 1. B. G. Teubner, Leipzig, 1996.

[dP67] Roger M. du Plessis. Poor Man's Explanation of Kalman Filtering or how I stopped worrying and learned to love Matrix Inversion. Technical report, Autonetics Division, Rockwell International, 3370 Miraloma Avenue, Anaheim, California 92803, June 1967.

[Koh96] Markus Kohler. Vision Based Remote Control in Intelligent Home Environments. In B. Girod, H. Niemann, and H.-P. Seidel, editors, *3D Image Analysis and Synthesis'96*, pages 147–154, University of Erlangen-Nuremberg/Germany, November 18–19 1996. infix Verlag. ISBN 3-89601-000-X; see also *Ein ergonomisches Dialogsystem zur Steuerung von technischen Systemen in Wohnbereichen mittels Gestenerkennung — Abschlußbericht* at Dep. f. CG/University of Dortmund.

[Koh97] Markus Kohler. Technical details and ergonomical aspects of gesture recognition applied in intelligent home environments. Technical Report 638, Informatik VII, University of Dortmund, January 1997.

[SKZ95] Michael Stark, Markus Kohler, and P. G. ZYKLOP. Video based gesture recognition for human computer interaction. Technical Report 593, Informatik VII, University of Dortmund, November 1995. More details in Projektgruppe 247: ZYKLOP — Visueller Mensch-Rechner-Dialog — Abschlußbericht (1995).

[ZYK95] Projektgruppe 247: ZYKLOP. Visueller Mensch-Rechner-Dialog — Abschlußbericht. Technical report, Informatik VII, University of Dortmund, 1995.