Learning an Outlier-Robust Kalman Filter

Jo-Anne Ting[1], Evangelos Theodorou[1] and Stefan Schaal[1,2]

[1] University of Southern California, Los Angeles, CA 90089
[2] ATR Computational Neuroscience Laboratories, Kyoto, Japan
{joanneti, etheodor, sschaal}@usc.edu

Abstract. We introduce a modified Kalman filter that performs robust, real-time outlier detection, without the need for manual parameter tuning by the user. Systems that rely on high quality sensory data (for instance, robotic systems) can be sensitive to data containing outliers. The standard Kalman filter is not robust to outliers, and other variations of the Kalman filter have been proposed to overcome this issue. However, these methods may require manual parameter tuning, use of heuristics or complicated parameter estimation procedures. Our Kalman filter uses a weighted least squares-like approach by introducing weights for each data sample. A data sample with a smaller weight has a weaker contribution when estimating the current time step's state. Using an incremental variational Expectation-Maximization framework, we learn the weights and system dynamics. We evaluate our Kalman filter algorithm on data from a robotic dog.

1 Introduction

Systems that rely on high quality sensory data are often sensitive to data containing outliers. While data from sensors such as potentiometers and optical encoders are easily interpretable in their noise characteristics, other sensors such as visual systems, GPS devices and sonar sensors often provide measurements populated with outliers. As a result, robust, reliable detection and removal of outliers is essential in order to process these kinds of data. For example, in the application domain of robotics, legged locomotion is vulnerable to sensory data of poor quality, since one undetected outlier can disturb the balance controller to the point that the robot loses stability.

An outlier is generally defined as an observation that "lies outside some overall pattern of distribution" [1]. Outliers may originate from sensor noise (producing values that fall outside a valid range), from temporary sensor failures, or from unanticipated disturbances in the environment (e.g., a brief change of lighting conditions for a visual sensor). Note that some prior knowledge about the observed data's properties must be known. Otherwise, it is impossible to discern if a data sample that lies some distance away from the data cloud is truly an outlier or simply part of the data's structure.

For real-time applications, storing data samples may not be a viable option due to the high frequency of sensory data and insufficient memory resources. In this scenario, sensor data are made available one at a time and must be discarded once they have been observed. Hence, techniques that require access to the entire set of data samples, such as the Kalman smoother are not applicable. Instead, the Kalman filter [2] is a more suitable method, since it assumes that only data samples up to the current time step have been observed.

The Kalman filter is a widely used tool for estimating the state of a dynamic system, given noisy measurement data. It is the optimal *linear* estimator for linear Gaussian systems, giving the minimum mean squared error [3]. Using state estimates, the filter can also estimate what the corresponding (output) data are. However, the performance of the Kalman filter degrades when the observed data contains outliers.

To address this, previous work has tried to make the Kalman filter more robust to outliers by addressing the sensitivity of the squared error criterion to outliers [4] non-Gaussian, heavy-tailed distributions for random variables (e.g., [5]) or for observation and state noise, e.g., [6]. However, the resulting estimation of parameters may be quite complicated for systems with transient disturbances, and these filters may be more difficult to implement. Other approaches use resampling techniques or numerical integration, e.g., [7], that are not suitable for real-time applications.

Yet another class of methods uses a weighted least squares approach, as done in robust least squares [8], where each data sample is assigned a weight that indicates its contribution to the hidden state estimate at each time step, e.g., [9]. These methods model the weights as some heuristic function of the data (e.g., the Huber function [8]) and often require manual tuning of threshold parameters for optimal performance. Using incorrect or inaccurate estimates for the weights may lead to deteriorated performance, so special care is necessary with these techniques.

In this paper, we are interested in making the Kalman filter more robust to the outliers in the observations (i.e. the filter should identify and eliminate possible outliers as it tracks observed data). Identifying outliers in the state is a different problem, left for another paper. We introduce a modified Kalman filter that can detect outliers in the observed data without the need for manual parameter tuning or use of heuristic methods. For ease of analytical computation, we assume Gaussian distributions for variables and states. We illustrate the performance of this robust Kalman filter on robotic data, comparing it with other robust Kalman filter methods and demonstrating its effectiveness at detecting outliers in the observations.

2 Outlier Detection in the Kalman Filter

Let us assume we have data observed over N time steps, $\{\mathbf{z}_k\}_{k=1}^{N}$, and the corresponding hidden states as $\{\boldsymbol{\theta}_k\}_{k=1}^{N}$ (where $\boldsymbol{\theta}_k \in \Re^{d_2 \times 1}, \mathbf{z}_k \in \Re^{d_1 \times 1}$). The Kalman filter system equations are:

$$\begin{aligned} \mathbf{z}_k &= \mathbf{C}\boldsymbol{\theta}_k + \mathbf{v}_k \\ \boldsymbol{\theta}_k &= \mathbf{A}\boldsymbol{\theta}_{k-1} + \mathbf{s}_k \end{aligned} \tag{1}$$

where $\mathbf{C} \in \Re^{d_1 \times d_2}$ is the observation matrix, $\mathbf{A} \in \Re^{d_2 \times d_2}$ is the state transition matrix, $\mathbf{v}_k \in \Re^{d_1 \times 1}$ is the observation noise at time step k, and $\mathbf{s}_k \in \Re^{d_2 \times 1}$ is the state noise at time step k. We assume $\mathbf{v}_k \sim \text{Normal}\left(0, \mathbf{R}\right)$, $\mathbf{s}_k \sim \text{Normal}\left(0, \mathbf{Q}\right)$, where $\mathbf{R} \in \Re^{d_1 \times d_1}$ and $\mathbf{Q} \in \Re^{d_2 \times d_2}$ are diagonal covariance matrices (with vectors $\mathbf{r}$ and $\mathbf{q}$ on their diagonals) for the observation and state noise, respectively. The corresponding filter propagation and update equations are, for $k = 1, .., N$:

$$\begin{aligned}
&\textbf{Propagation:} \\
\boldsymbol{\theta}'_k &= \mathbf{A} \left\langle \boldsymbol{\theta}_{k-1} \right\rangle && (2) \\
\boldsymbol{\Sigma}'_k &= \mathbf{A} \boldsymbol{\Sigma}_{k-1} \mathbf{A}^T + \mathbf{Q} && (3) \\
&\textbf{Update:} \\
\mathbf{S}'_k &= \left(\mathbf{C} \boldsymbol{\Sigma}'_k \mathbf{C}^T + \mathbf{R} \right)^{-1} && (4) \\
K'_k &= \boldsymbol{\Sigma}'_k \mathbf{C}^T \mathbf{S}'_k && (5) \\
\left\langle \boldsymbol{\theta}_k \right\rangle &= \boldsymbol{\theta}'_k + K'_k \left(\mathbf{z}_k - \mathbf{C} \boldsymbol{\theta}'_k \right) && (6) \\
\boldsymbol{\Sigma}_k &= \left(\mathbf{I} - K'_k \mathbf{C} \right) \boldsymbol{\Sigma}'_k && (7)
\end{aligned}$$

where $\left\langle \boldsymbol{\theta}_k \right\rangle$[3] is the posterior mean vector of the state $\boldsymbol{\theta}_k$, $\boldsymbol{\Sigma}_k$ is the posterior covariance matrix of $\boldsymbol{\theta}_k$, and $\mathbf{S}'_k$ is the covariance matrix of the residual prediction error—all at time step k. The system dynamics ($\mathbf{C}$, $\mathbf{A}$, $\mathbf{R}$ and $\mathbf{Q}$) are unknown, and we can use a maximum likelihood framework to estimate these parameter values . Unfortunately, the standard Kalman filter is not robust to outliers.

2.1 Robust Kalman Filtering with Bayesian Weights

To overcome this limitation, we introduce a scalar weight w_k for each observed data sample $\mathbf{z}_k$ such that the variance of $\mathbf{z}_k$ is weighted with w_k, as done in [10]. [10] considers a weighted least squares regression model and assumes that the weights are known and given. We place a Gamma prior distribution over the the weights to ensure they remain positive, as done previously in [11]. Additionally, we learn estimates for the system dynamics at each time step. The prior distributions of our model are:

$$\begin{aligned}
\mathbf{z}_k | \boldsymbol{\theta}_k, w_k &\sim \text{Normal}\left(\mathbf{C} \boldsymbol{\theta}_k, \mathbf{R}/w_k \right) \\
\boldsymbol{\theta}_k | \boldsymbol{\theta}_{k-1} &\sim \text{Normal}\left(\mathbf{A} \boldsymbol{\theta}_{k-1}, \mathbf{Q} \right) \\
w_k &\sim \text{Gamma}\left(a_{w_k}, b_{w_k} \right)
\end{aligned} \qquad (8)$$

We can treat this problem as an Expectation-Minimization-like (EM) learning problem [12, 13] and maximize the log likelihood $\log p(\boldsymbol{\theta}_{1:N})$. Due to analytical issues, we only have access to a lower bound of this measure. This lower bound is based on an expected value of the "complete" data likelihood $\left\langle \log p\left(\boldsymbol{\theta}_{1:N}, \mathbf{z}_{1:N}, \mathbf{w} \right) \right\rangle$, formulated over all variables of the learning problem. Since we are considering a real-time problem, we will have observed only data samples $\mathbf{z}_{1:k}$ at time step k. Consequently, we should consider the log evidence

[3] Note that $\langle\rangle$ denotes the expectation operator

of only the data samples observed to date, i.e., $\log p\left(\boldsymbol{\theta}_{1:k}, \mathbf{z}_{1:k}, \mathbf{w}_{1:k}\right)$, when estimating the posterior distributions of random variables at time step k.

The expectation of the complete data likelihood should be taken with respect to the true posterior distribution of all hidden variables $Q\left(\mathbf{w}, \boldsymbol{\theta}\right)$. Since this is an analytically intractable expression, we use a technique from variational calculus to construct a lower bound and make a factorial approximation of the true posterior as follows: $Q\left(\mathbf{w}, \boldsymbol{\theta}\right) = \prod_{i=1}^{N} Q\left(w_i\right) \prod_{i=1}^{N} Q\left(\boldsymbol{\theta}_i | \boldsymbol{\theta}_{i-1}\right) Q(\boldsymbol{\theta}_0)$ (e.g., [13]). This factorization of $\boldsymbol{\theta}$ conserves the Markov property that Kalman filters, by definition, have and makes the resulting posterior distributions over hidden variables analytically tractable. The factorial approximation was chosen purposely so that $Q(w_k)$ is independent from $Q(\boldsymbol{\theta}_k)$; performing joint inference of w_k and $\boldsymbol{\theta}_k$ does not make sense in the context of our generative model. The final EM update equations for time step k are:

E-step:

$$\boldsymbol{\Sigma}_k = \left(\langle w_k \rangle \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{C}_k + \mathbf{Q}_k^{-1} \right)^{-1} \tag{9}$$

$$\langle \boldsymbol{\theta}_k \rangle = \boldsymbol{\Sigma}_k \left(\mathbf{Q}_k^{-1} \mathbf{A}_k \langle \boldsymbol{\theta}_{k-1} \rangle + \langle w_k \rangle \mathbf{C}_k^T \mathbf{R}_k^{-1} \mathbf{z}_k \right) \tag{10}$$

$$\langle w_k \rangle = \frac{a_{w_k,0} + \frac{1}{2}}{b_{w_k,0} + \left\langle \left(\mathbf{z}_k - \mathbf{C}_k \boldsymbol{\theta}_k\right)^T \mathbf{R}_k^{-1} \left(\mathbf{z}_k - \mathbf{C}_k \boldsymbol{\theta}_k\right) \right\rangle} \tag{11}$$

M-step:

$$\mathbf{C}_k = \left(\sum_{i=1}^{k} \langle w_i \rangle \mathbf{z}_i \langle \boldsymbol{\theta}_i \rangle^T \right) \left(\sum_{i=1}^{k} \langle w_i \rangle \left\langle \boldsymbol{\theta}_i \boldsymbol{\theta}_i^T \right\rangle \right)^{-1} \tag{12}$$

$$\mathbf{A}_k = \left(\sum_{i=1}^{k} \langle \boldsymbol{\theta}_i \rangle \langle \boldsymbol{\theta}_{i-1} \rangle^T \right) \left(\sum_{i=1}^{k} \left\langle \boldsymbol{\theta}_{i-1} \boldsymbol{\theta}_{i-1}^T \right\rangle \right)^{-1} \tag{13}$$

$$r_{km} = \frac{1}{k} \sum_{i=1}^{k} \langle w_i \rangle \left\langle (\mathbf{z}_{im} - \mathbf{C}_k(m,:)\boldsymbol{\theta}_i)^2 \right\rangle \tag{14}$$

$$q_{kn} = \frac{1}{k} \sum_{i=1}^{k} \left\langle (\boldsymbol{\theta}_{in} - \mathbf{A}_k(n,:)\boldsymbol{\theta}_{i-1})^2 \right\rangle \tag{15}$$

where $m = 1, .., d_1$, $n = 1, .., d_2$; r_{km} is the mth coefficient of the vector $\mathbf{r}_k$; q_{kn} is the nth coefficient of the vector $\mathbf{q}_k$; $\mathbf{C}_k(m,:)$ is the mth row of the matrix $\mathbf{C}_k$; $\mathbf{A}_k(n,:)$ is the nth row of the matrix $\mathbf{A}_k$; and $a_{w_k,0}$ and $b_{w_k,0}$ are prior scale parameters for the weight w_k. Equations (9) to (15) should be computed once for each time step k (e.g., [14]) when the data sample $\mathbf{z}_k$ becomes available.

Since storing sensor data is not possible in real-time applications, (12) to (15)—which require access to all observed data samples up to time step k—need to be re-written using only values observed, calculated or used in the current time step k. We can do this by collecting sufficient statistics in (12) to (15) and rewriting them as:

$$\mathbf{C}_k = \mathrm{sum}_k^{\mathbf{wz}\boldsymbol{\theta}^T} \left(\mathrm{sum}_k^{\mathbf{w}\boldsymbol{\theta}\boldsymbol{\theta}^T} \right)^{-1} \tag{16}$$

$$\mathbf{A}_k = \mathrm{sum}_k^{\boldsymbol{\theta}\boldsymbol{\theta}'} \left(\mathrm{sum}_k^{\boldsymbol{\theta}'\boldsymbol{\theta}'} \right)^{-1} \tag{17}$$

$$r_{km} = \frac{1}{k} \left[\mathrm{sum}_{km}^{\mathbf{w}zz} - 2\mathbf{C}_k(m,:)\mathrm{sum}_{km}^{\mathbf{w}z\boldsymbol{\theta}} + \mathrm{diag}\left\{ \mathbf{C}_k(m,:)\mathrm{sum}_k^{\mathbf{w}\boldsymbol{\theta}\boldsymbol{\theta}^T} \mathbf{C}_k(m,:)^T \right\} \right] \tag{18}$$

$$q_{kn} = \frac{1}{k} \left[\mathrm{sum}_{kn}^{\theta^2} - 2\mathbf{A}_k(n,:)\mathrm{sum}_{kn}^{\boldsymbol{\theta}\boldsymbol{\theta}'} + \mathrm{diag}\left\{ \mathbf{A}_k(n,:)\mathrm{sum}_k^{\boldsymbol{\theta}'\boldsymbol{\theta}'} \mathbf{A}_k(n,:)^T \right\} \right] \tag{19}$$

where $m = 1, .., d_1$, $n = 1, .., d_2$, and the sufficient statistics are:

$$\mathrm{sum}_k^{\mathbf{wz}\boldsymbol{\theta}^T} = \langle w_k \rangle \mathbf{z}_k \langle \boldsymbol{\theta}_k \rangle^T + \mathrm{sum}_{k-1}^{\mathbf{wz}\boldsymbol{\theta}^T} \qquad \mathrm{sum}_k^{\mathbf{w}\boldsymbol{\theta}\boldsymbol{\theta}^T} = \langle w_k \rangle \left\langle \boldsymbol{\theta}_k \boldsymbol{\theta}_k^T \right\rangle + \mathrm{sum}_{k-1}^{\mathbf{w}\boldsymbol{\theta}\boldsymbol{\theta}^T}$$

$$\begin{aligned} \mathrm{sum}_k^{\boldsymbol{\theta}\boldsymbol{\theta}'} &= \langle \boldsymbol{\theta}_k \rangle \langle \boldsymbol{\theta}_{k-1} \rangle^T + \mathrm{sum}_{k-1}^{\boldsymbol{\theta}\boldsymbol{\theta}'} & \mathrm{sum}_k^{\boldsymbol{\theta}'\boldsymbol{\theta}'} &= \left\langle \boldsymbol{\theta}_{k-1} \boldsymbol{\theta}_{k-1}^T \right\rangle + \mathrm{sum}_{k-1}^{\boldsymbol{\theta}'\boldsymbol{\theta}'} \\ \mathrm{sum}_{km}^{\mathbf{w}zz} &= \langle w_k \rangle z_{km}^2 + \mathrm{sum}_{k-1}^{\mathbf{w}zz} & \mathrm{sum}_{km}^{\mathbf{w}z\boldsymbol{\theta}} &= \langle w_k \rangle z_{km} \boldsymbol{\theta}_k + \mathrm{sum}_{k-1,m}^{\mathbf{w}z\boldsymbol{\theta}} \\ \mathrm{sum}_{kn}^{\theta^2} &= \left\langle \theta_{kn}^2 \right\rangle + \mathrm{sum}_{k-1,n}^{\theta^2} & \mathrm{sum}_{kn}^{\theta\boldsymbol{\theta}'} &= \langle \theta_{kn} \rangle \langle \boldsymbol{\theta}_{k-1} \rangle + \mathrm{sum}_{kn}^{\theta\boldsymbol{\theta}'} \end{aligned}$$

A few remarks should be made regarding the initialization of priors used in (9) to (11), (16) to (19). In particular, the prior scale parameters $a_{w_k,0}$ and $b_{w_k,0}$ should be selected so that the weights $\langle w_k \rangle$ are 1 with some confidence, i.e., the algorithm starts by assuming most data samples are inliers. We set $a_{w_k,0} = 1$ and $b_{w_k,0} = 1$ so that $\langle w_k \rangle$ has a prior mean of $a_{w_k,0}/b_{w_k,0} = 1$ with a variance of $a_{w_k,0}/b_{w_k,0}^2 = 1$. This set of values is generally valid for any data set and/or application and does not need to be modified, unless the user has good reason to insert strong biases towards particular parameter values. Since prior knowledge about the observed data's properties must be known in order to distinguish if a data sample is an outlier or part of the data's structure, this Bayesian approach provides a natural framework to incorporate this information.

Secondly, the algorithm is relatively insensitive to the the initialization of $\mathbf{A}$ and $\mathbf{C}$ and will always converge to the same final solution, regardless of these values. For our experiments, we initialize $\mathbf{C} = \mathbf{A} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. The initial values of $\mathbf{R}$ and $\mathbf{Q}$ should be set based on the user's initial estimate of how noisy the observed data is (e.g., $\mathbf{R} = \mathbf{Q} = 0.01\mathbf{I}$ for noisy data, $\mathbf{R} = \mathbf{Q} = 10^{-4}\mathbf{I}$ for less noisy data).

2.2 Relationship to the Kalman Filter

If we substitute (2) and (3) into (4) to (7), we reach recursive expressions for $\langle \boldsymbol{\theta}_k \rangle$ and $\boldsymbol{\Sigma}_k$, proving that our model is a variant of the Kalman filter. By applying this sequence of algebraic manipulations in reverse order to (9) and (10), we arrive at the following:

Propagation:

$$\boldsymbol{\theta}'_k = \mathbf{A}_k \langle \boldsymbol{\theta}_{k-1} \rangle \tag{20}$$

$$\boldsymbol{\Sigma}'_k = \mathbf{Q}_k \tag{21}$$

Update:

$$\mathbf{S}'_k = \left(\mathbf{C}_k \boldsymbol{\Sigma}'_k \mathbf{C}_k^T + \frac{1}{\langle w_k \rangle} \mathbf{R}_k \right)^{-1} \tag{22}$$

$$K'_k = \boldsymbol{\Sigma}'_k \mathbf{C}_k^T \mathbf{S}'_k \tag{23}$$

$$\langle \boldsymbol{\theta}_k \rangle = \boldsymbol{\theta}'_k + K'_k \left(\mathbf{z}_k - \mathbf{C}_k \boldsymbol{\theta}'_k \right) \tag{24}$$

$$\boldsymbol{\Sigma}_k = \left(\mathbf{I} - K'_k \mathbf{C}_k \right) \boldsymbol{\Sigma}'_k \tag{25}$$

Close examination of the above equations show that (9) and (10) in the Bayesian model correspond to standard Kalman filter equations, with modified expressions for $\boldsymbol{\Sigma}'_k$ and $\mathbf{S}'_k$ and time-varying system dynamics. $\boldsymbol{\Sigma}'_k$ is no longer *explicitly* dependent on $\boldsymbol{\Sigma}_{k-1}$, since $\boldsymbol{\Sigma}_{k-1}$ does not appear in (21). However, the current state's covariance $\boldsymbol{\Sigma}_k$ is still dependent on the previous state's covariance $\boldsymbol{\Sigma}_{k-1}$ (through parameters K' and $\mathbf{C}_k$).

Additionally, the term $\mathbf{R}_k$ in $\mathbf{S}'_k$ is now weighted. Equation (11) reveals that if the prediction error in $\mathbf{z}_k$ is so large that it dominates the denominator, then the weight $\langle w_k \rangle$ of that data sample will be very small. If $\mathbf{z}_k$ has a very small weight $\langle w_k \rangle$, then $\mathbf{S}'_k$, the posterior covariance of the residual prediction error, will be very small, leading to a very small Kalman gain K'_k. In short, the influence of the data sample $\mathbf{z}_k$ will be downweighted when predicting $\boldsymbol{\theta}_k$, the hidden state at time step k. The resulting Bayesian algorithm has a computational complexity on the same order as that of a standard Kalman filter, since matrix inversions are still needed, as in the standard Kalman filter. In comparison to other Kalman filters that use heuristics or require more involved computation/implementation, this outlier-robust Kalman filter is principled and easy to implement.

3 Experimental Results

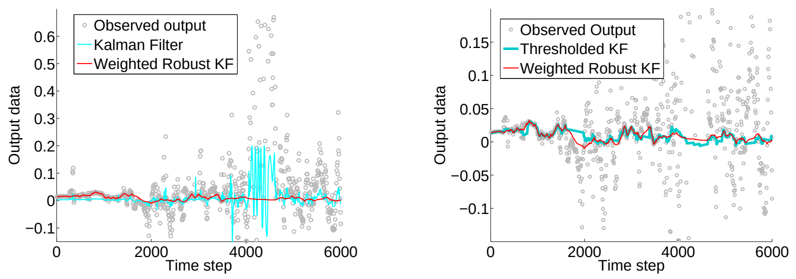
We evaluated our weighted robust Kalman filter on data collected from a a robotic dog, LittleDog, manufactured by Boston Dynamics Inc. (Cambridge, MA), and compared it with two other filters. We omitted the filter of [9], since we had difficulty implementing it. Instead, we used a hand-tuned thresholded Kalman filter to serve as a baseline comparison. The two other filters consist of the standard Kalman filter and a Kalman filter where outliers are determined by thresholding on the Mahalanobis distance. If the Mahalanobis distance exceeds a certain threshold value, the associated data sample is considered an outlier and ignored. If we have a priori access to the entire data set and are able to *manually hand-tune* this threshold value accordingly, the thresholded Kalman filter gives *near-optimal* performance. Recall that we are interested in the Kalman filter's prediction of the observed data and detection of outliers in the observations. Estimation of the system dynamics for the purpose of parameter identification is a different problem, and more details can be found in [15].

3.1 LittleDog Robot

We evaluated all filters on a 12 degree-of-freedom robotic dog, LittleDog, shown in Fig. 1. The robot dog has two sources that measure its orientation: a motion capture (MOCAP) system and an on-board inertia measurement unit (IMU). Both provide a quaternion q of the robot's orientation: q_{MOCAP} from the MOCAP and q_{IMU} from the IMU. q_{IMU} drifts over time, since the IMU cannot provide stable orientation estimation but its signal is clean. In contrast, q_{MOCAP} has outliers and noise, but no drift. We would like to estimate the offset between q_{MOCAP} and q_{IMU}, and this offset is

Fig. 1. LittleDog

a *noisy slowly drifting signal containing outliers*. Depending on the quality of estimate desired, we can estimate it with a straight line, as done in [11]. Alternatively, if we want to estimate the signal more accurately, we can use the proposed outlier-robust Kalman filter to track it. For optimal performance, we manually tuned **C**, **A**, **R** and **Q** for the standard Kalman filter—a tricky and time-consuming process. The system dynamics of the thresholded Kalman filter were learnt using a maximum likelihood framework. Its threshold parameter was manually tuned for best performance on this data set.



(a) Predicted data for the standard and weighted robust KFs

(b) Predicted data for the thresholded and weighted robust KFs

Fig. 2. Observed vs. predicted data from LittleDog robot shown for all Kalman filters (KF), over 6000 samples

Figure 2 shows the offset data (in gray circles) between q_{MOCAP} and q_{IMU} for one of the four quaternion coefficients, collected over 6000 data samples, at 1 sample/time step. Figure 2(a) shows that the standard Kalman filter fails to detect outliers occurring between the 4000th and 5000th sample. Figure 2(b) shows that the thresholded Kalman filter does not react as violently as the standard Kalman filter to outliers and, in fact, appears to perform similarly to the weighted robust Kalman filter. This is to be expected, given we hand-tuned the threshold parameter for optimal performance.

4 Conclusions

We derived an outlier-robust Kalman filter by introducing weights for each data sample. This Kalman filter learns the weights and the system dynamics, without the need for any manual parameter tuning by the user, heuristics or sampling. It performs as well as a hand-tuned Kalman filter (that required prior knowledge of the data) on robotic data. It provides an easy-to-use competitive alternative for robust tracking of sensor data and offers a simple outlier detection mechanism that can potentially be applied to more complex, nonlinear filters.

Acknowledgments

This research was supported in part by National Science Foundation grants ECS-0325383, IIS-0312802, IIS-0082995, ECS-0326095, ANI-0224419, a NASA grant AC#98−516, an AFOSR grant on Intelligent Control, the ERATO Kawato Dynamic Brain Project funded by the Japanese Science and Technology Agency, and the ATR Computational Neuroscience Laboratories.

References

1. Moore, D.S., McCabe, G.P.: Introduction to the Practice of Statistics. W.H. Freeman & Company (March 1999)
2. Kalman, R.E.: A new approach to linear filtering and prediction problems. In Transactions of the ASME - Journal of Basic Engineering **183** (1960) 35–45
3. Morris, J.M.: The Kalman filter: A robust estimator for some classes of linear quadratic problems. IEEE Transactions on Information Theory **22** (1976) 526–534
4. Huber, P.J.: Robust estimation of a location parameter. Annals of Mathematical Statistics **35** (1964) 73–101
5. West, M.: Robust sequential approximate Bayesian estimation. Journal of the Royal Statistical Society, Series B **43** (1981) 157–166
6. Masreliez, C.: Approximate non-Gaussian filtering with linear state and observation relations. IEEE Transactions on Automatic Control **20** (1975) 107–110
7. Kitagawa, G., Gersch, W.: Smoothness priors analysis of time series. In: Lecture Notes in Statistics. Springer-Verlag (1996)
8. Huber, P.J.: Robust Statistics. Wiley (1973)
9. Chan, S.C., Zhang, Z.G., Tse, K.W.: A new robust Kalman filter algorithm under outliers and system uncertainties. In: IEEE International Symposium on Circuits and Systems. IEEE (2005) 4317–4320
10. Gelman, A., Carlin, J., Stern, H., Rubin, D.: Bayesian Data Analysis. Chapman and Hall (2000)
11. Ting, J., D'Souza, A., Schaal, S.: Automatic outlier detection: A Bayesian approach. In: IEEE International Conference on Robotics and Automation. (2007)
12. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of Royal Statistical Society. Series B **39**(1) (1977) 1–38
13. Ghahramani, Z., Beal, M.: Graphical models and variational methods. In Saad, D., Opper, M., eds.: Advanced Mean Field Methods - Theory and Practice. MIT Press (2000)
14. Neal, R.M., Hinton, G.E.: A view of the EM algorithm that justifies incremental, sparse, and other variants. In Jordan, M.I., ed.: Learning in Graphical Models. MIT Press (1999) 355–368
15. Ting, J., D'Souza, A., Schaal, S.: Bayesian regression with input noise for high dimensional data. In: Proceedings of the 23rd International Conference on Machine Learning, ACM (2006) 937–944