Stochastic Filtering for Motion Trajectory in Image Sequences Using a Monte Carlo Filter with Estimation of Hyper-Parameters

Naoyuki Ichimura

AIST02-J-00002-3

Technical Report of Information Technology Research Institute
National Institute of Advanced Industrial Science and Technology
AIST Tsukuba Central 2, 1-1-1 Umezono, Tsukuba, Ibaraki 305-8568, Japan

Abstract

False matching due to errors in feature extraction and changes in illumination between frames may occur in feature tracking in image sequences. False matching leads to outliers in feature motion trajectory. One way of reducing the effect of outliers is stochastic filtering using a state space model for motion trajectory. Hyper-parameters in the state space model, e.g., variances of noise distributions, must be determined appropriately to control tracking motion and outlier rejection properly. Likelihood can be used to estimate hyper-parameters, but it is difficult to apply online tracking due to computational cost. To estimate hyper-parameters online, we include hyper-parameters in state vector and estimate feature coordinates and hyper-parameters simultaneously. A Monte Carlo filter is used in state estimation, because adding hyper-parameters to state vector makes state space model nonlinear. Experimental results using synthetic and real data show that the proposed method can estimate appropriate hyper-parameters for tracking motion and reducing the effect of outliers.

Contents

Chapter 1

Introduction

Feature tracking is essential to image sequence analysis in computer vision. False matching due to errors in feature extraction and changes in illumination between frames may occur in feature tracking. False matching leads to outliers in the feature motion trajectory. The effect of outliers must be reduced to yield good data for subsequent processing such as shape from motion[1][2].

One way of reducing this effect is stochastic filtering using a state space model for motion trajectory, as typified by the Kalman filter[3]–[7]. Since state estimation result based on stochastic filtering varies with hyper-parameters, e.g., variances of noise distributions, in state space model, appropriate hyper-parameters should be used to carry out state estimation properly. Likelihood $p\left(\boldsymbol{y}_{\,k} \mid Y_{k-1}\right)$, $k = 1, 2, \ldots, t$ of time series of measurement $Y_t = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_t\}$ can be used to estimate hyper-parameters, but it is difficult to apply tracking due to computational cost; numerical search like coarse-to-fine method for grid of hyper-parameters is necessary. Stochastic property of motion generally varies with time and lag is not allowed in tracking, thus online estimation of hyper-parameters is quite important to feature tracking.

The Multiple-Model adaptive filter[8] can be used for online estimation of hyper-parameters. Multiple models with different hyper-parameters are prepared, and state estimation results of these models are combined based on likelihoods of models for current measurement to use appropriate hyper-parameters. Since all combinations of hyper-parameters should be given in this method, if appropriate hyper-parameters are not in the given scope, the filter can not yield good state estimation results. In addition, a lot of models may be necessary when many hyper-parameters are needed and/or quantization of them must be fine to obtain good accuracy; it reduces efficiency of algorithm.

In this paper, we include hyper-parameters in state vector and estimate feature coordinates and hyper-parameters simultaneously online. Using this simultaneous estimation, we can adjust hyper-parameters to feature motion automatically. Since adding hyper-parameters to state vector makes state space model nonlinear, we need state estimation method for nonlinear model. The sequential Monte Carlo (SMC) method used for nonlinear/non-Gaussian state space models has been proposed recently[9]–[12], so the Monte Carlo filter (MCF)[10], a type of the SMC, is applied to state estimation of our model. Online estimation of hyper-parameters by adding them to state vector was proposed in 1970's, but the extended Kalman filter which needs linear approximation of model and assumption of Gaussian distribution for state and noises could not yield good state estimation results[13][14]. The approximation and assumption are not needed for the MCF.

The MCF allows us to use the non-Gaussian observation noise distribution which is useful to reduce the effect of outliers[15][16]. Using heavy-tailed non-Gaussian distribution, e.g., Cauchy distribution, we can represent both observation noise with high probability and outliers with low probability. Thus state estimation for motion trajectory with outliers can be done.

Adaptive filtering for motion trajectory with time-dependent statistical property can be realized using online estimation of hyper-parameters and non-Gaussian noise distribution mentioned above. Experiments using synthetic and real data are shown to discuss the usefulness of the proposed method.

Chapter 2

State Space Model for Motion Trajectory

2.1 Notation and prior models

A state space model is represented as follows:

$$\boldsymbol{x}_t = \boldsymbol{F}\boldsymbol{x}_{t-1} + \boldsymbol{G}\boldsymbol{v}_t \tag{2.1}$$

$$\boldsymbol{y}_t = \boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{w}_t \tag{2.2}$$

where Eq. (2.1) is the state transition equation, $\boldsymbol{x}_t$ is the state vector, and $\boldsymbol{v}_t$ is the system noise vector. Matrix $\boldsymbol{F}$ and $\boldsymbol{G}$ are system matrices. Equation (2.2) is the observation equation, $\boldsymbol{y}_t$ is the observation vector, and $\boldsymbol{w}_t$ is the observation noise vector. Matrix $\boldsymbol{H}$ is an observation matrix.

Stochastic filtering based on the state space model is used to estimate state vector $\boldsymbol{x}_t$ using time series of measurement $Y_t = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t\}$, i.e., to calculate conditional probability density function $p(\boldsymbol{x}_t \mid Y_t)$.

The observation vector

$$\boldsymbol{y}_t = [x(t), y(t)]^T \tag{2.3}$$

represents coordinates of features in images. The following smoothness prior models, i.e., constant velocity model, are used for motion trajectory:

$$x(t+1) = 2x(t) - x(t-1) \tag{2.4}$$

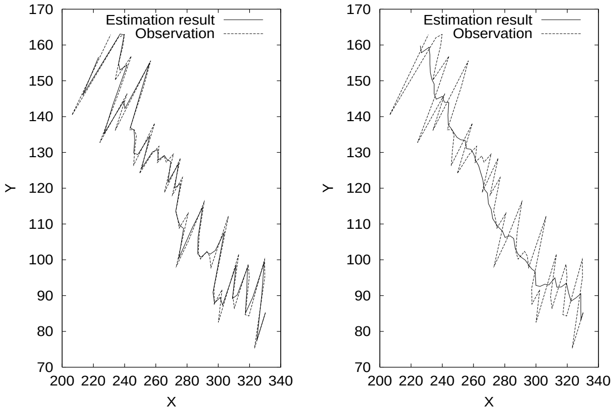
$$y(t+1) = 2y(t) - y(t-1) \tag{2.5}$$

This assumption means that the change in velocity of feature between 2 adjacent frames is small and the velocity varies gradually in image sequence.

2.2 Noise distribution and hyper-parameters

Variables in system noise vector $\boldsymbol{v}_t$ are independent, and their distributions are represented as $q(v; m_q, \tau^2)$, where m_q and τ are parameters of location and scale. The multivariate form of system noise distribution is denoted as $q_v(\boldsymbol{v}_t; \boldsymbol{m}_{vq}, \boldsymbol{T})$. Variables in observation noise vector $\boldsymbol{w}_t$ are independent, and their distributions are represented as $r(w; m_r, \sigma^2)$, where m_r and σ are parameters of location and scale. The multivariate form of observation noise distribution is denoted as $r_v(\boldsymbol{w}_t; \boldsymbol{m}_{vr}, \boldsymbol{\Sigma})$. Since we assume the mean of noise is zero, the entities in $\boldsymbol{m}_{vq}$ and $\boldsymbol{m}_{vr}$ are zero.

Parameters τ^2 and σ^2 in system noise and observation noise distribution are called hyper-parameters, because their role is to control state estimation. Figure 2.1 shows an example of the effect of hyper-parameters for state estimation. The prior models of Eq.(2.4) and (2.5) and Gaussian distribution for



(a) (b)

Figure 2.1: An example of the effect of hyper-parameters for state estimation. (a) $\tau^2 = 1.0$ and $\sigma^2 = 1.0$. (b) $\tau^2 = 0.01$ and $\sigma^2 = 50.0$.

system and observation noise were used. State estimation was done by the Kalman filter. The hyper-parameters, $(\tau^2, \sigma^2) = (1.0, 1.0)$ (Fig.2.1 (a)) and $(0.01, 50.0)$ (Fig.2.1 (b)), were used for the same observation. As we can see from Fig.2.1(a) and (b), the state estimation results are completely different. This example shows the importance of hyper-parameters in state estimation.

Likelihood $p\left(\boldsymbol{y}_k \mid Y_{k-1}\right)$, $k = 1, 2, \ldots, t$, $Y_0 = \phi$ of time series of measurement can be used to determine hyper-parameters. It is difficult, however, to apply feature tracking due to computational cost. Since lag is not allowed in feature tracking and stochastic property of motion generally varies with time, online estimation of hyper-parameters is needed.

2.3 State space model to estimate hyper-parameters

From the prior models of Eq.(2.4) and (2.5), $x\left(t\right), y\left(t\right), x\left(t-1\right)$ and $y\left(t-1\right)$ are included in the state vector. And hyper-parameters are also included in the state vector to estimate them online as follows:

$$\boldsymbol{x}_t = \left[x_s\left(t\right), y_s\left(t\right), x_s\left(t-1\right), y_s\left(t-1\right), \log\tau^2\left(t\right), \log\sigma^2\left(t\right)\right]^T \tag{2.6}$$

The logarithm of hyper-parameters is used to preserve scale parameters as positive. Since hyper-parameters are in the state vector, they are estimated online; both feature coordinates and hyper-parameters are estimated simultaneously to adjust the hyper-parameters to feature motion automatically.

Since hyper-parameters are included in the state vector and time-dependent to follow feature motion, we define the system matrices, observation matrix, system noise vector, and observation noise vector of the state space model as follows:

$$\mathbf{F} = \begin{bmatrix} 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & 2 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.7}$$

$$\boldsymbol{v}_t = [v_x(t), v_y(t), v_{\tau^2}(t), v_{\sigma^2}(t)]^T \tag{2.8}$$

$$\boldsymbol{w}_t = [w_x(t), w_y(t)]^T \tag{2.9}$$

where v_{τ^2} and v_{σ^2} are variables for system noise of hyper-parameters. The location and scale in multivariate form of system noise distribution $q_v\left(\boldsymbol{v}_t; \boldsymbol{m}_{vq}, \boldsymbol{T}\right)$ are $\boldsymbol{m}_{vq} = [0, 0, 0, 0]^T$ and $\boldsymbol{T} = \mathrm{diag}\left(\tau^2, \tau^2, \nu^2, \xi^2\right)$. The location and scale in multivariate form of observation noise distribution are $\boldsymbol{m}_{\ vr} = [0, 0]^T$ and $\boldsymbol{\Sigma} = \mathrm{diag}\left(\sigma^2, \sigma^2\right)$.

ν^2 and ξ^2 are called hyper-hyper-parameters, because these parameters govern change in hyper-parameters.

Chapter 3

State Estimation by Monte Carlo Filter

3.1 Monte Carlo filter algorithm

Adding hyper-parameters to state vector makes the state space model nonlinear, since system and observation noise vectors are the function of hyper-parameters in the state vector. We need state estimation method for such a nonlinear model, because the Kalman filter commonly used assumes a linear/Gaussian model.

The sequential Monte Carlo method (SMC) was proposed for nonlinear/non-Gaussian models[9]-[12]. The Monte Calro filter (MCF)[10], a type of the SMC, is applied to state estimation of the proposed model. The procedure of the MCF is briefly described below.

In the MCF, probability distributions used in state estimation are approximated by m particles, i.e., m realizations from distributions, as follows:

$$\text{prediction} : p\left(\boldsymbol{x}_t \mid Y_{t-1}\right), \quad \left\{\boldsymbol{p}_t^{(1)}, \ldots, \boldsymbol{p}_t^{(m)}\right\}$$
$$\text{filter} : p\left(\boldsymbol{x}_t \mid Y_t\right), \quad \left\{\boldsymbol{f}_t^{(1)}, \ldots, \boldsymbol{f}_t^{(m)}\right\}$$

System noise distribution is approximated as follows:

$$\text{system noise} : q_v\left(\boldsymbol{v}_t ; \boldsymbol{m}_{vq},\, \boldsymbol{T}\right), \quad \left\{\boldsymbol{v}_t^{(1)}, \ldots, \boldsymbol{v}_t^{(m)}\right\}$$

State estimation is carried out by repeating the following one-step-ahead prediction and filter process:

[MCF algorithm]

[Step 1: generation of particles of initial distribution]

Generate m particles, i.e., m random vectors, from an initial distribution of state $p_0\left(\boldsymbol{x}\right)$. These particles $\boldsymbol{f}_0^{(i)}$ $(i = 1, \ldots, m)$ are regarded as initial filter distribution.

[Step 2: filtering] Repeat next steps.

[Step 2-1: generation of particles for system noise] Generate m particles $\boldsymbol{v}_t^{(i)} \sim q_v\left(\boldsymbol{v}_t ; \boldsymbol{m}_{vq}, \boldsymbol{T}\right)$, $i = 1, \ldots, m$.

[Step 2-2: one-step-ahead prediction] Compute particles, $\boldsymbol{p}_t^{(i)}$, representing the prediction distribution $p\left(\boldsymbol{x}_t \mid Y_{t-1}\right)$ using the state transition equation:

$$\boldsymbol{p}_t^{(i)} = \boldsymbol{F}\boldsymbol{f}_{t-1}^{(i)} + \boldsymbol{G}\boldsymbol{v}_t^{(i)},\ i = 1, \ldots, m \tag{3.1}$$

$\left[\text{Step 2-3: calculation of likelihood of } \boldsymbol{p}_t^{(i)}\right]$ Calculate the likelihood $\alpha_t^{(i)}$ of particle $\boldsymbol{p}_t^{(i)}$ using measurement vector $\boldsymbol{y}_t$ and observation noise distribution as follows:

$$\alpha_t^{(i)} \quad = \quad r_v\left(\boldsymbol{y}_t - \boldsymbol{H}\boldsymbol{p}_t^{(i)} ; \boldsymbol{m}_{vr}, \boldsymbol{\Sigma}\right),\ i = 1, \ldots, m \tag{3.2}$$

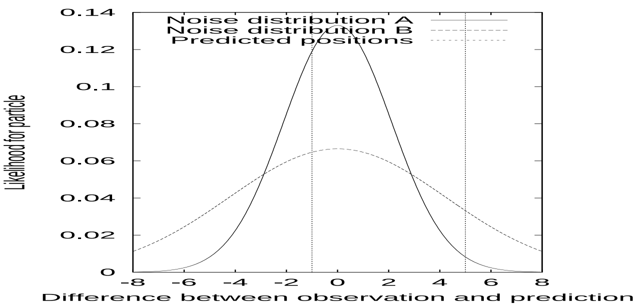


Figure 3.1: An example of observation noise distributions with different hyper-parameters.

[Step 2-4: calculation of filter distribution] Calculate particles $\boldsymbol{f}_t^{(i)}$ of filter distribution by resampling particles $\boldsymbol{p}_t^{(i)}$ in accordance with the following probabilities:

$$Pr\left(\boldsymbol{f}_t^{(i)} = \boldsymbol{p}_t^{(i)}\right) = \frac{\alpha_t^{(i)}}{\alpha_t^{(1)} + \ldots + \alpha_t^{(m)}},\ i = 1, \ldots, m \tag{3.3}$$

3.2 Intuition for adaptation of hyper-parameters by the MCF

The likelihoods of particles used in resampling in Step 2-4 are determined by the difference between measurement and predicted position, and observation noise distribution, i.e., $r_v\left(\boldsymbol{y}_t - \boldsymbol{H}\boldsymbol{p}_t^{(i)}; \boldsymbol{m}_{vr}, \boldsymbol{\Sigma}\right)$ in Eq.(3.2). The particle with high likelihood survives the resampling, thus hyper-parameters in such particle (state vector) are chosen as good one for adapting filter to feature motion.

Figure 3.1 shows an example of two observation noise distributions with different hyper-parameters. If the difference between measurement and predicted position is -1, observation noise distribution A gives higher likelihood than one given by observation noise distribution B. So the hyper-parameters of distribution A is chosen with high probability. If the difference between measurement and predicted position is 5, the hyper-parameters of distribution B survives with high probability. Hyper-parameters are adapted to feature motion through such a selection mechanism in the MCF.

Chapter 4

Non-Gaussian Observation Noise Distribution for Outlier Rejection

The MCF allows us to use the heavy-tailed non-Gaussian observation noise distribution which is useful to reduce the effect of outliers[15][16]. The following Cauchy distribution $C\left(0, \sigma^2\right)$ is selected for observation noise:

$$r\left(w; 0, \sigma^2\right) = \frac{\sigma}{\pi\left\{w^2 + \sigma^2\right\}} \tag{4.1}$$

where σ is scale parameter. It is heavy-tailed (Fig.4.1) and represents both observation noise with high probability and outliers with low probability. State estimation for motion trajectory with outliers, therefore, can be realized.

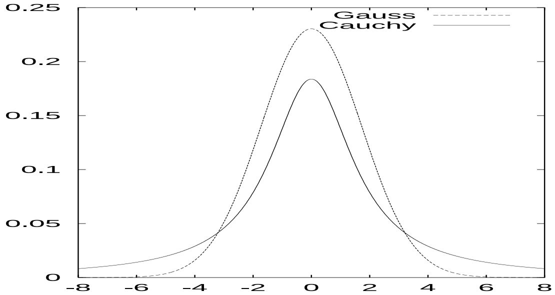


Figure 4.1: Gaussian distribution and Cauchy distribution; location: 0, scale: 3.

Chapter 5

Experimental Results

Experiments using synthetic and real data are done to discuss the usefulness of the proposed method.

5.1 Synthetic data: problem for controlling the tradeoff

First, the problem for controlling the tradeoff between tracking abrupt motion changes and outlier rejection is used to show the adaptation ability of the proposed filter.

Using heavy-tailed non-Gaussian distribution, we can realize state estimation for motion trajectory with outliers as mentioned in Chapter 4. A problem remains, however: the delay in tracking abrupt changes in feature motion (Fig.5.1). The purpose of this experiment is to control the tradeoff between outlier rejection and tracking abrupt motion changes. Tradeoff is controlled by changing hyper-parameters, i.e., scale parameters of observation noise distribution and system noise distribution. The larger scale parameter of system noise distribution, for example, enables higher tracking speed. Thus controlling tradeoff is equivalent to adjusting hyper-parameters based on feature motion.

Synthetic data shown in Fig.5.2 was used where the solid line represents the true trajectory and the dotted line represents observation with noise and outliers. This data contains 3 outliers ($t = 15, 30$ and 75) and an abrupt change in motion ($t = 50$). The Cauchy distribution $C\ \left(0, \tau^2\right)$ was also used for system noise to represent both smooth motion that may occur frequently and abrupt changes in motion that may occur infrequently.

To evaluate the best performance of the proposed filter, log-likelihood was used to determine hyper-hyper-parameters ν^2 and ξ^2[17]. Log-likelihood for the proposed filter can be computed using approximation as follows[10]:

$$l\left(\nu^2, \xi^2\right) \cong \sum_{t=1}^{N} \log\left(\sum_{i=1}^{m} \alpha_t^{(i)}\right) - N \log m \tag{5.1}$$

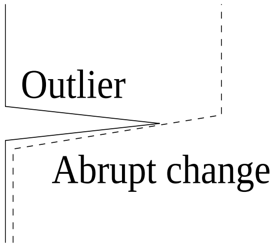


Figure 5.1: Tradeoff between outlier rejection and tracking abrupt changes in motion.

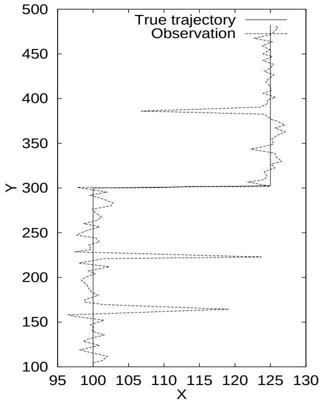


Figure 5.2: Synthetic data used in the experiment.

Table 5.1: Hyper-hyper-parameters determined for the proposed filter, hyper-parameters determined for Kalman fitler, and mean squared error between estimated and true trajectories of both filters.

Proposed filter			Kalman fitler		
ν^2	ξ^2	MSE	τ^2	σ^2	MSE
0.006	0.034	0.118	0.20	8.5	0.269

where $\alpha_t^{(i)}$ is likelihood calculated in Step 2-3 of the MCF algorithm and N is the length of the sequence of measurement Y_t. A coarse-to-fine method was used to determine hyper-hyper-parameters; first, coarse grid $\{1, 2, \ldots, 20\} \times \{1, 2, \ldots, 20\}$ was used as the candidate of parameters and then a more detailed grid was selected based on the maximum log-likelihood in the coarse grid. Parameters determined for synthetic data are shown in Table 5.1.

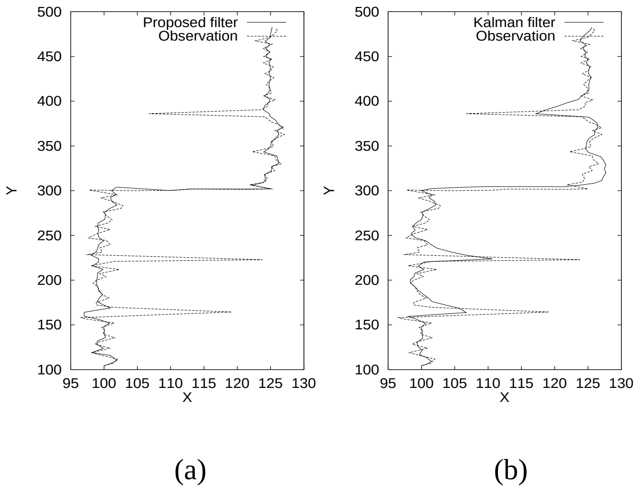
The number of particles in the MCF, m, was 10,000. The initial distribution for coordinates of features was Gaussian with mean vector $[x(1), y(1), x(1), y(1)]^T$ and covariance matrix $\operatorname{diag}(10, 10, 10, 10)$. The initial distribution for hyper-parameters was uniform distribution in $[-8, 8]$. Estimated coordinates of features were obtained from the mode of 2D distribution of $x_s(t)$ and $y_s(t)$ computed from particles $\boldsymbol{f}_t^{(i)}$ using a Parzen estimator[18]. The estimate of each hyper-parameter is also obtained using 1D distribution.

The proposed filter was compared to the Kalman filter for linear/Gaussian model which was obtained by removing hyper-parameters from state vector and using Gaussian distribution for both system and observation noise. Log-likelihood was also used to determine hyper-parameters τ^2 and σ^2 for the linear/Gaussian model to compare the best performance. Log-likelihood for the model is represented as follows:

$$l_{LGM}\left(\tau^2, \sigma^2\right) = -\frac{1}{2}\left\{nN \log 2\pi + \sum_{t=1}^{N} \log\left|\boldsymbol{V}_{t|t-1}\right| + \sum_{t=1}^{N}\left(\boldsymbol{y}_t - \boldsymbol{m}_{t|t-1}\right)^T \boldsymbol{V}_{t|t-1}^{-1}\left(\boldsymbol{y}_t - \boldsymbol{m}_{t|t-1}\right)\right\} \tag{5.2}$$

where n is the number of dimensions of the state vector and $\boldsymbol{m}_{t|t-1}$ and $\boldsymbol{V}_{t|t-1}$ are the mean vector and covariance matrix of prediction distribution calculated in the Kalman filter. Determined parameters are shown in Table 5.1.

Figure 5.3 and 5.4 show the estimated trajectories and errors of coordinates. The effect of outliers



(a) (b)

Figure 5.3: Filtering results for synthetic data. (a) proposed filter. (b) Kalman filter.

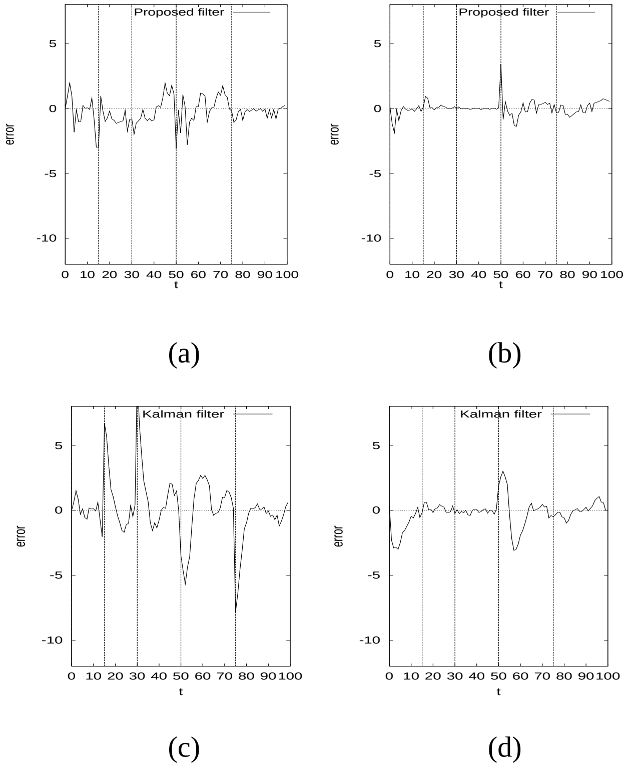
was clearly reduced by the proposed filter (Fig.5.3 (a) and Fig.5.4 (a),(b)), while it was appeared in the results of the Kalman filter (Fig.5.3 (b) and Fig.5.4 (c),(d)). The tradeoff between tracking abrupt changes and outlier rejection was controlled properly by the proposed filter, because the scale parameter of system noise distribution, τ^2, was increased very rapidly at $t = 50$ (Fig.5.5). Since the larger scale parameter enables higher tracking speed, errors in coordinates of the proposed filter were smaller than that of the Kalman filter around $t = 50$ (Fig.5.4). Table 5.1 shows the mean squared error between estimated and true trajectories; the proposed filter was more accurate than the Kalman filter.

5.2 Synthetic data: problem for improving accuracy of motion and shape

The effect of filtering for improving the accuracy of motion and shape estimation is shown. The factorization based method[1] which can eliminate the projection error due to affine approximation of perspective projection was used to estimate motion and shape from motion trajectories of feature points. 62 feature points on sphere and motion trajectoies of them obtained from synthetic motion were used. True trajectoies and observation with Gaussian noise and outliers are shown in Fig.5.6 (a) and (b). The variance of the Gaussian noise for each trajectory was changed randomly using uniform distribution in [1:16] and outliers obtained from unifrom distribution in [-20:20] were added to the Gaussian noise with 6% probabilty. Since each trajectory has different statistical property and the property varies with time due to outliers, different hyper-parameters for each trajectory and time are needed. Using the proposed fileter, we can estimate hyper-parameters for each trajectory online. A large amount of computation was necessary to estimate hyper-hyper-parameters based on the likelihood Eq.(5.1) for every trajectory, thus the same hyper-hyper-parameters used in 5.1, ν^2=0.006 and ξ^2=0.034, were used instead. Other conditions in 5.1, e.g., the number of particles, were also used.

Although some noise components remained, the effect of noise and outliers was reduced from the observation by the proposed filter (Fig.5.6 (b) and (c)): Fig.5.6 (d) shows typical results of the filter for the trajectories selected from Fig.5.6 (c) .

The rotation angles of sphere recovered from observation had increasing error and some bumpy changes (Fig.5.7 (a)), while such error and changes were reduced in the rotation angles recovered from estimated trajectoies (Fig.5.7 (b)). Multi-frame algorithm like the factorization based method used here reduces the effect of noise by *averaging*. If the mean of noise components is not close to zero, offset remains after averaging and it affects the result of motion and shape estimation. Offset is caused by



(a) (b)

(c) (d)

Figure 5.4: Error in coordinates for synthetic data. Abscissa represents time. (a),(b): error in x and y coordinates for the proposed filter. (c),(d): error in x and y coordinates for the Kalman filter. Vertical lines show when outliers (t=15,30,75) and abrupt change in motion (t=50) occur.

outliers and the proposed filter can reduce its effect, thus the errors of motion are reduced as shown in Fig.5.7. Shape error was measured by $\|\boldsymbol{S}_{true} - \boldsymbol{S}_{estimated}\| \,/\, \|\boldsymbol{S}_{true}\|$ where $\boldsymbol{S}$ is shape matrix. The error of shape reconstructed from the observation was 5.8% while one reconstructed from the estimated trajectoies was 2.5%.

5.3 Real data: online estimation of hyper-parameters for eye tracker

In results for online implementation, the proposed method was incorporated into an eye tracker. The template of the right eye was stored and used to track the eye in the image sequence (Fig.5.8): exhaustive search was used to detect the eye position in the first frame and a local search based on the eye position in the previous frame was used in successive frames. An image processing board with hardware for computing normalized correlations was used in template matching. The proposed filter was applied to estimate hyper-parameters online.

Hyper-parameters were properly varied based on changes in motion: large scale parameter of system noise distribution was used for large velocities and small one selected for the small velocities (Fig.5.9(a) and (b)). This result shows the adaptation ability of the proposed filter for the real online system. The frame rate was about 10 [frame/s] on PC with Pentium III 500MHz.

We should give care to the lower bound of the scale parameter of system noise distribution in practical application. For example, if the object being tracked remains stationary for a long period, the scale parameter would decay to very small number and thus the system noise would decay to almost zero. If the object moves again, tracking may fail. So the lower bound of the scale parameter should be defined.

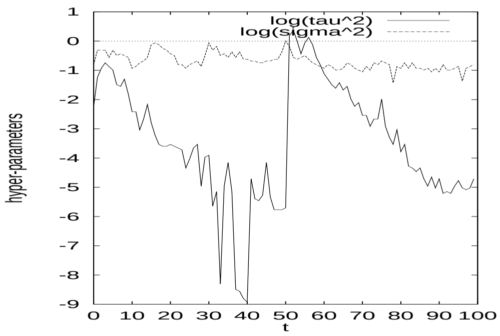
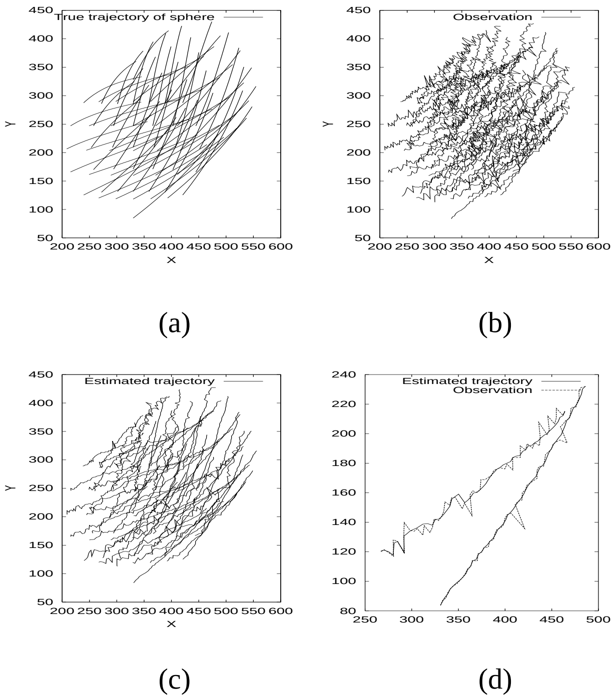


Figure 5.5: Changes in hyper-parameters, $\log 10\left(\tau^2\right)$ and $\log 10\left(\sigma^2\right)$, in the filtering.

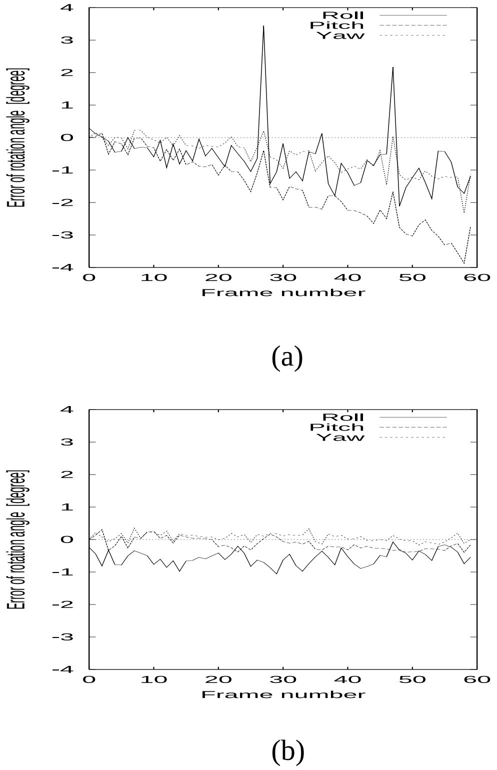
It is done easily because it is almost equivalent to define the smallest search region size for the local search: -3 was used in this experiment.



(a) (b)

(c) (d)

Figure 5.6: Motion trajectories obtained from 62 feature points on sphere. (a) True motion trajectories. (b) Observation with noise and outliers. (c) Estimated trajectories obtained from the proposed filter. (d) Typical results of the proposed filter selected from (c).



(a)

(b)

Figure 5.7: Errors in recovered rotation angles: roll, pitch and yaw. (a) Errors in rotation angles obtained from observation. (b) Errors in rotation angles obtained from estimated trajectoies.

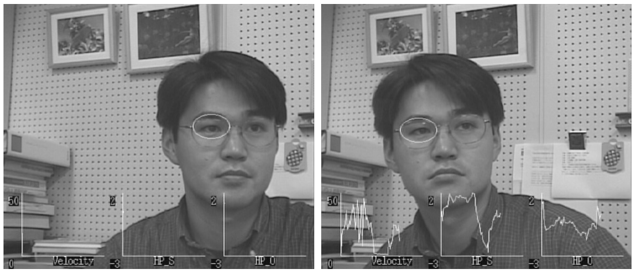
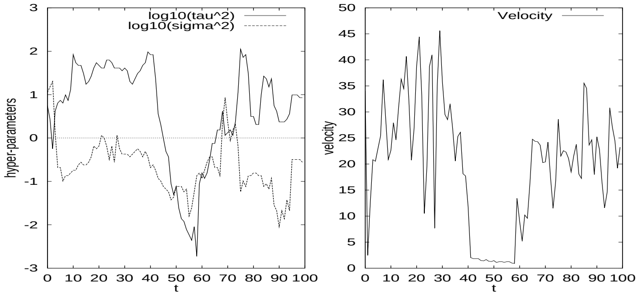


Figure 5.8: Snapshots of the eye tracking image sequence. The arc in pictures shows the estimated eye position obtained from the proposed filter.



(a) (b)

Figure 5.9: Changes in Hyper-parameters and velocity. (a) Changes in hyper-parameters. (b) Changes in feature velocity.

Chapter 6

Conclusions

Stochastic filtering for a motion trajectory with estimation of hyper-parameters has been proposed. Hyper-parameters governing state estimation were included in state vector and estimated simultaneously with feature coordinates. Experimental results for synthetic data and online implementation verified the usefulness of our proposed method.

One of the problems of the proposed filter is the effect of approximation of distribution by particles for state estimation. Estimation using particles sometime converges only on a number of varieties of particles; this phenomenon is called as “sample impoverishment”[19]. The algorithm for increasing varieties of particles should be incorporated to avoid the phenomenon and keep the adaptation ability of the filter especially for long image sequences. This is a projected work in our research.

Bibliography

[1] S. Christy and R. Horaud:“Euclidean reconstruction: from paraperspective to perspective,” Proc. 4th European Conf. Comp. Vis., Vol.2, pp. 129-140, 1996.

[2] C. J. Poelman and T. Kanade: “A paraperspective factorization method for shape and motion recovery,” IEEE Trans. Pattern Anal. & Mach. Intell., Vol.19, No.3, pp. 206-218, 1997

[3] J.S. Meditch: “Stochastic optimal linear estimation and control,” McGraw-Hill, 1969.

[4] E. D. Dickmanns and V. Graefe: “Applications of dynamic monocular machine vision,” Machine Vision and Applications, Vol.1, No.4, pp. 241-261, 1988.

[5] R. Deriche and O. Faugeras: “Tracking line segments,” Proc. 1st European Conf. Comp. Vis., pp. 259-268, 1990.

[6] E. D. Dickmanns and B. D. Mysliwetz: “Recursive 3-D road and relative ego-state recognition,” IEEE Trans. Pattern Anal. & Mach. Intell., Vol.14, No.2, pp. 199-213, 1992.

[7] L. Goncalves, E. D. Bernardo, E. Ursella and P. Perona: “Monocular tracking of the human arm in 3D,” Proc. 5th Int. Conf. Comp. Vis., pp. 764-770, 1995.

[8] Y. Bar-Shalom and T. E. Fortmann:“Tracking and data association,” Academic Press, 1988.

[9] N. J. Gordon, D. J. Salmond, and A. F. M. Smith: “Novel approach to nonlinear/non-Gaussian Bayesian state estimation,” IEE Proc. F, Vol. 140, No. 2, pp. 107-113, 1993.

[10] G.Kitagawa: “Monte Carlo filter and smoother for non-Gaussian nonlinear state space models,” J. Comp. and Graph. Stat., Vol. 5, No. 1, pp. 1-25, 1996.

[11] M. Isard and A. Blake: “Condensation – conditional density propagation for visual tracking,” Internat. J. Comp. Vis., Vol. 29, No. 1, pp. 5-28, 1998.

[12] J. S. Liu and R. Chen: “Sequential Monte Carlo methods for dynamic systems,” J. Amer. Statist. Assoc., Vol. 93, pp. 1032-1044, 1998.

[13] B. D. O. Anderson and J. B. Moore:“Optimal Filtering,” Prentice-Hall, 1979.

[14] G. Kitagawa: “A self-organizing state-space model,” J. Amer. Statist. Assoc., Vol. 93, pp. 1203-1215, 1998.

[15] C. J. Masreliez and R. D. Martin: “Robust Bayesian estimation for the linear model and robustifying the Kalman filter,” IEEE Trans. on Auto. Control, Vol. AC-22, No. 3, pp. 361-371, 1977.

[16] K. L. Lange, R. J. A. Little and J. M. G. Taylor:“Robust statistical modeling using the t distribution,” J. Amer. Statist. Assoc., Vol.84, No.408, pp. 881-896, 1989.

[17] H. Akaike: “Likelihood and the Bayes procedures,” Bayesian Stat., pp. 143-166, 1980.

[18] K. Fukunaga: “Introduction to Statistical Pattern Recognition,” Academic Press, 1972.

[19] O. King and D. A. Forsyth: “How does CONDENSATION behave with a finite number of samples?,” Proc. 6th European Conf. Comp. Vis., pp. 695-709, 2000.