TR-M-0046

Predictive Filtering for Robust 2D Feature Tracking in Real-time

チラズ ベルアブデカデ-ル
Chiraz BenAbdelkader

高橋 和彦
Kazuhiko TAKAHASHI

大谷 淳
Jun OHYA

1999.8.31

ATR 知能映像通信研究所

Predictive Filtering for Robust 2D Feature Tracking in Real-time

Chiraz BenAbdelkader, Kazuhiko Takahashi, and Jun Ohya

I. Introduction

Recent years have seen increased interest in human motion analysis, due to its importance in various real-world applications, such as in advanced man-machine interface systems, visual communications (e.g. tele-conferencing) and virtual reality. Real-time 3D pose estimation of humans in video is a basic part of many of these applications. The estimated pose data can be used to drive (animate) avatars in a visual communication environment, for example.

A system currently under development at ATR-MIC addresses this problem based on trinocular CCD cameras. This system, depicted below in Figure 1, is part of a cyberspace communication environment, wherein people can communicate with each other remotely by controlling 3D CG avatars in a common cyberspace. First, a silhouette of the human body is obtained via background differencing and other lo-level morphological operations. The feature extraction module then locates certain salient features of the human silhouette, namely the centroid, head, feet, and hands. At most three[1] 2D points (one from each view) are thus obtained for each body part. Finally, the 3D position is computed for each body part using the best two of its corresponding three 2D feature points, based on the triangulation method. The view selection module decides which two of the three points to use.

The 2D feature extraction method used here is based solely on heuristic analysis of the silhouette contour. One obvious limitation of this approach is its sensitivity to segmentation errors. In real-life applications, where the segmentation process is often error prone due to the presence of clutter in the background, this can be a problem. Furthermore, contour analysis cannot handle self-occlusions of the human body. When body parts partially or totally occlude one another (in certain postures), it may not be possible to distinguish them just by looking at the silhouette. Previous computer vision work suggests two ways in which feature extraction can be made more robust:

[1]The number is less than three if feature extraction fails on one or more of the views, such as when the feature is occluded.

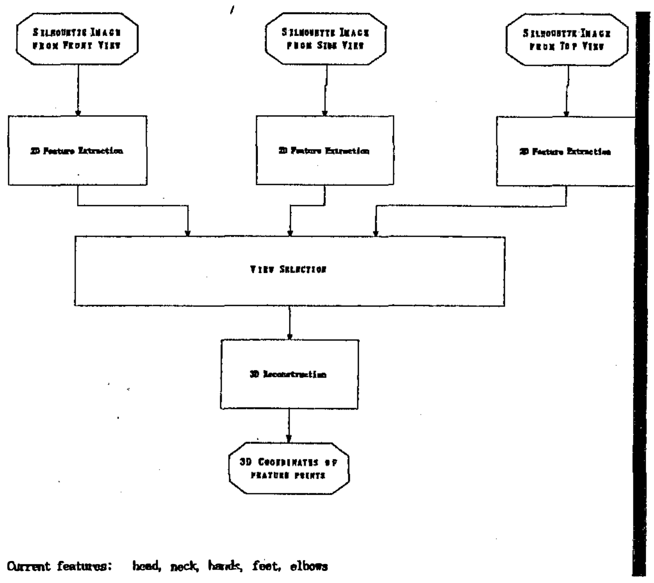

Silhouette Image from Front View
Silhouette Image from Side View
Silhouette Image from Top View
2D Feature Extraction
2D Feature Extraction
2D Feature Extraction
View Selection
3D Reconstruction
3D Coordinates of feature points
Current features: head, neck, hands, feet, elbows

Figure 1 – 3D Pose Estimation System Architecture

1. Use of Frame-to-frame 2D tracking. Here, by assuming that human body motion is smooth (continuous), each feature (body part) can be constrained to lie within a window not 'too far' from its location in the previous frame.
2. Use of apriori knowledge about the structure of the human body, and the physical constraints on the positions and movement of the body parts. While this approach is more promising, it requires massive computational power.

This paper takes the first approach. We propose the dynamic model-based 2D feature tracking framework depicted in Figure 2. Here, heuristic contour analysis of the silhouette is still used as the primary feature extraction mechanism, due to its simplicity and fastness. However, an *Arbitration* module is subsequently used to judge the goodness of its results, and whether they are consistent with those of the previous frame. This 'arbitration' is done separately for each feature point. The Feature Search module then locates any feature point for which contour analysis is deemed to have failed. In this module, some appearance-based matching scheme is used to search for the feature point within a window centered at that point's predicted location in the image. Finally, the predictive filter determines the location of each feature point in the next image frame, based on its location in previous frame(s).

Currently, we have only applied this method for tracking hands, mainly because they are more prone to occlusion. However, the proposed framework is general enough to be extended to tracking other body parts.

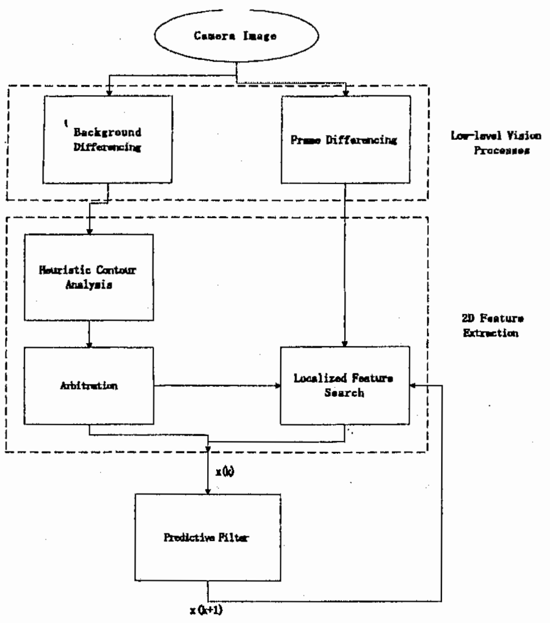


Figure 2 – 2D Tracking of Significant Points

The rest of the paper is organized as follows. In Sections 1, 2 and 3, we discuss the implementation of the method's three main modules, namely the Arbitration, Feature Search and Prediction modules. Section 4 gives the results obtained when testing the proposed method on video sequences in real time. Finally, Section 5 gives conclusions and future directions for further improvement.

II. Arbitration Module

2D feature extraction via contour analysis is typically done by tracing the boundary points of the silhouette, and identifying points of maximum curvature. Thus it is a fast and simple. However, this simplicity is what causes such a method to be unreliable. The two obvious limitations are that it is sensitive to segmentation errors, and does not work well when self-occlusion occurs. The Arbitration module essentially determines if the results of the contour analysis module are unreliable, based on the predicted position of the feature point (i.e. *contextual* knowledge), and *a priori* knowledge about the structure of the human body. In the case of the hand point, it checks if any of the following conditions are true:

- The point is outside the window centered at the hand point's predicted location in the image. The size (i.e. height and width) of the window are design parameters, that roughly correspond to the size of a typical hand.

- The lateral distance between the predicted hand point and the head point is less than some threshold, indicating that the hand is too close to the body, and may be totally or partially occluded by the body.

- The distance between the predicted hand point and the neck point is less than some threshold, indicating that the arm is bent (or bending) and may be confused with the elbow point in the contour analysis.

III. Feature Search Module

The goal here is to locate the feature point within a window in the image centered at the predicted location for that point. This can be done via some appearance-based scheme, that matches known properties of the feature point within the search window. This approach requires an appearance description of the each feature point.

In the case of the hand point, we first classify all pixels within the search window as skin or non-skin pixels, then take the average of the skin pixels as the desired point. This method obviously has its limitations; It assumes the hand is bare, and is not always reliable since any skin classification is sensitive to lighting and varies from person to person. A template matching scheme will generally not work any better, because the shape and appearance of the hand varies with its posture.

IV. Prediction Module

This module determines the location of a feature point in the next frame, based on its location in previous frames. To this end, we use a predictive filter, which has the effect of modeling (or approaximating) the trajectory of a feature point as an autoregressive process, i.e. a stochastic process whose current value is a linear combination of its previous values. The goal here is to minimize the prediction error, in order to increase the robustness of the Feature Search module.

A. Linear Predictive Filtering

Predictive filtering is a stochastic modeling tool for predicting future values of a discrete-time stochastic process $\mathbf{x}(k)$[2] as a linear function of past samples of that process, namely $\hat{\mathbf{x}}(k) = \sum_{i=1}^{M} w_i \mathbf{x}(k-i)$. Figure 3 shows the input and output of a typical predictive filter. M is termed the order of the filter. Obviously, $\hat{\mathbf{x}}(k)$ can be seen as an *auto-regressive (AR) model* approximation of the input process $\mathbf{x}(k)$. The filter weights $\mathbf{w}(k)$ are not known a priori, but are instead estimated on-line. The filter is thus able to change (adapt) its model to track variations in the input process. This is in fact a general property of a larger family of filters, known as *adaptive linear filters.*

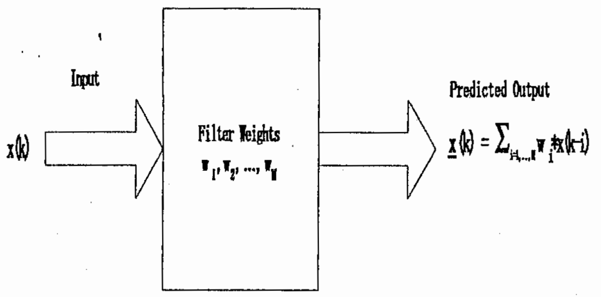


Figure 3 – Block Diagram of a Predictive Filter

At any time k, if an input value $\mathbf{x}(k)$ is available, the filter updates its weight vector $\mathbf{w}(k)$ based on the *a priori prediction error*, $\xi(k) = \mathbf{x}(k) - \hat{\mathbf{x}}(k)$, and such that the *a posteriori prediction error*, $\mathbf{e}(k) = \mathbf{x}(k) - \hat{\mathbf{x}}(k)$, is minimized (see Figure 4). This task is indeed a linear estimation problem. In the sequel, we discuss two linear recursive estimation methods for updating $\mathbf{w}(k)$; the Discrete Kalman Filter (DKF) and Recursive Least Squares (RLS). The main difference between the two is that in DKF $\mathbf{w}(k)$ is assumed to be a stochastic (time varying) process, while in RLS assumes it is assumed to be constant.

B. Kalman Filter

The discrete Kalman filter algorithm (see Figure 5) estimates the state of a stochastic system as a weighted linear sum of the dynamic model-based prediction of its state $x_k(-)$, and a (noisy) measurement of system output, $z(k)$. This weighting is, in a nutshell, based on the relative magnitudes of the measurement and dynamic model noise. Assuming the dynamic and measurement models are driven by Gaussian white noise, the Kalman filter estimate, $x_k(+)$, thus obtained is optimal in the minimum mean square (MMSE) sense.

[2]Vectors and matrices will be herein denoted in bold face.

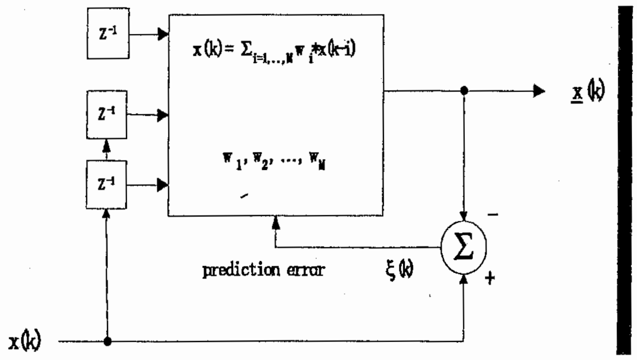


Figure 4 – Adaptive Updating of Weight Vector in Predictive Filter

We apply this into our problem of estimating $\mathbf{w}(k)$ by letting $\mathbf{w}(k)$ be the state vector, $\mathbf{x}(k)$ the measurement, $\mathbf{w}(k) = I_M\ \mathbf{w}(k)$ the dynamic (plant) equation[3], and $\mathbf{x}(k) = \sum_{i=1}^{M} w_i \mathbf{x}(k-i)$ the measurement equation. Consequently, the measurement matrix is given by: $H_k = \begin{bmatrix} \mathbf{x}(k-1) & \mathbf{x}(k-2) & \ldots & \mathbf{x}(k-M) \end{bmatrix}$. Initialization of the filter is done by setting $\mathbf{w}(0) = 0$ and $P(0) = \varepsilon \cdot I_M$, where ε is some appropriate small scalar. The error covariance matrices of the dynamic model noise and measurement model noise (Q_k and R_k respectively) can be assumed constant, i.e. independent of k, and specified similarly to $P(0)$.

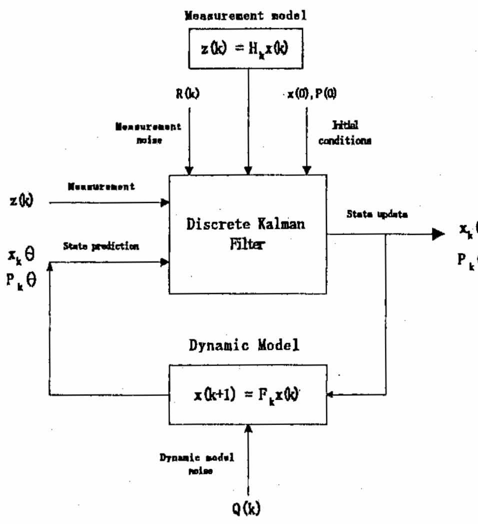


Figure 5 – Discrete Kalman Filter Algorithm

[3]The symbol I_M denotes the identity matrix with order M.

C. *Recursive Least Squares*

Consider the following scalar[4] Least Squares (LS) estimation problem. Given n data samples:

$$\underset{i=1..n}{d(i)} = \mathbf{w}(n)^T\mathbf{u}(i) \equiv \sum_{j=0}^{M-1} w(j)u(i-j)$$

we want to determine the best fit $\widehat{\mathbf{w}}(n)$ that minimized the cost function:

$$\xi(n) = \sum_{i=1}^{n} \lambda^{n-i}(d(i) - \widehat{\mathbf{w}}(n)^T u(i))$$

where $0 < \lambda \leq 1$. λ is the forgetting parameter which determines the (exponential) rate at which old inputs are 'forgotten' in the computation of $\widehat{\mathbf{w}}(n)$. $\lambda = 1$ signifies an infinite memory, and $\lambda < 1$ signifies finite memory. The optimal solution $\widehat{\mathbf{w}}(n)$ satisfies the *normal equations*:

$$\mathbb{P}(n)\widehat{\mathbf{w}}(n) = \mathbf{z}(n)$$

where $\mathbb{P}(n) \equiv \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}(i)\mathbf{u}(i)^T$ is the correlation matrix of the input vector $\mathbf{u}(n)$, and $\mathbf{z}(n) \equiv \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}(i)d(i)^T$ is the cross-correlation vector between $\mathbf{u}(i)$ and $d(i)$.

The RLS method is a recursive (i.e. non-batch) solution to the LS problem. It updates its estimate $\widehat{\mathbf{w}}(n)$ using one data sample at a time, as shown in Figure 6. The recursive solution is especially appropriate for when the data samples are not all available at the same time.

The problem of updating the filter weights vector $\mathbf{w}(n)$ can hence be posed as a LS estimation problem, by setting $d(n)$ to the filter input $x(n)$, and $u(n)$ to the previous filter input $x(n-1)$. Obviously this problem needs to be solved resursively (using RLS), since filter weights are to be updated whenever an input value $x(n)$ is available.

V. Results and Discussion

The proposed method was implemented using the Kalman filter algorithm to update the filter weights, and tested on video sequences in real-time. Shown below are the results of tracking the right hand. An interesting observation is that the weights always sum to 1.0. While results have shown that 2D tracking improves system performance considerably, it is not clear whether predictive filtering is worth our while. There is no significant improvement when using a filter order (or memory) of 1, 2 or 3. This in turn seems to indicate that the AR process is perhaps not a great model (i.e. predictor) of hand motion. Therefore we ought to simply use a filter order of 1, i.e. use the current location of a feature point as its predicted location in the next frame.

[4]We limit our discussion to scalar LS (i.e.$d(n)$ is scalar) for sake of clarity, but it can be easily generalized to vector LS.

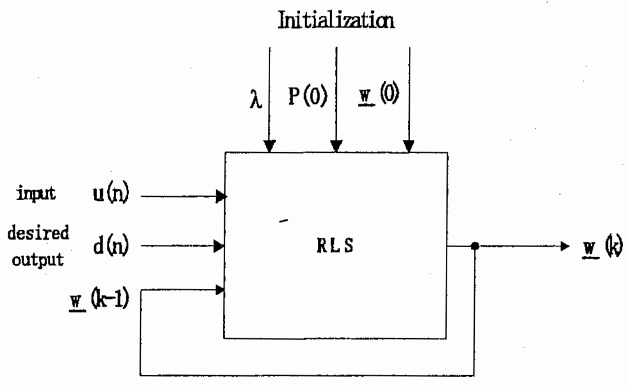


Figure 6 – Recursive Least Squares Algorithm

VI. Conclusions and Future Work

In this paper, we proposed a novel framework for robust 2D feature extraction in real-time. It combines heuristic contour analysis, model-based tracking, and some a priori knowledge about structure of human body. The framework was designed to be modular, so that alternative, possibly more robust, methods for localized feature search can be used. More a priori knowledge about human body can also be incoporated as needed, to improve robustness of the method.

References

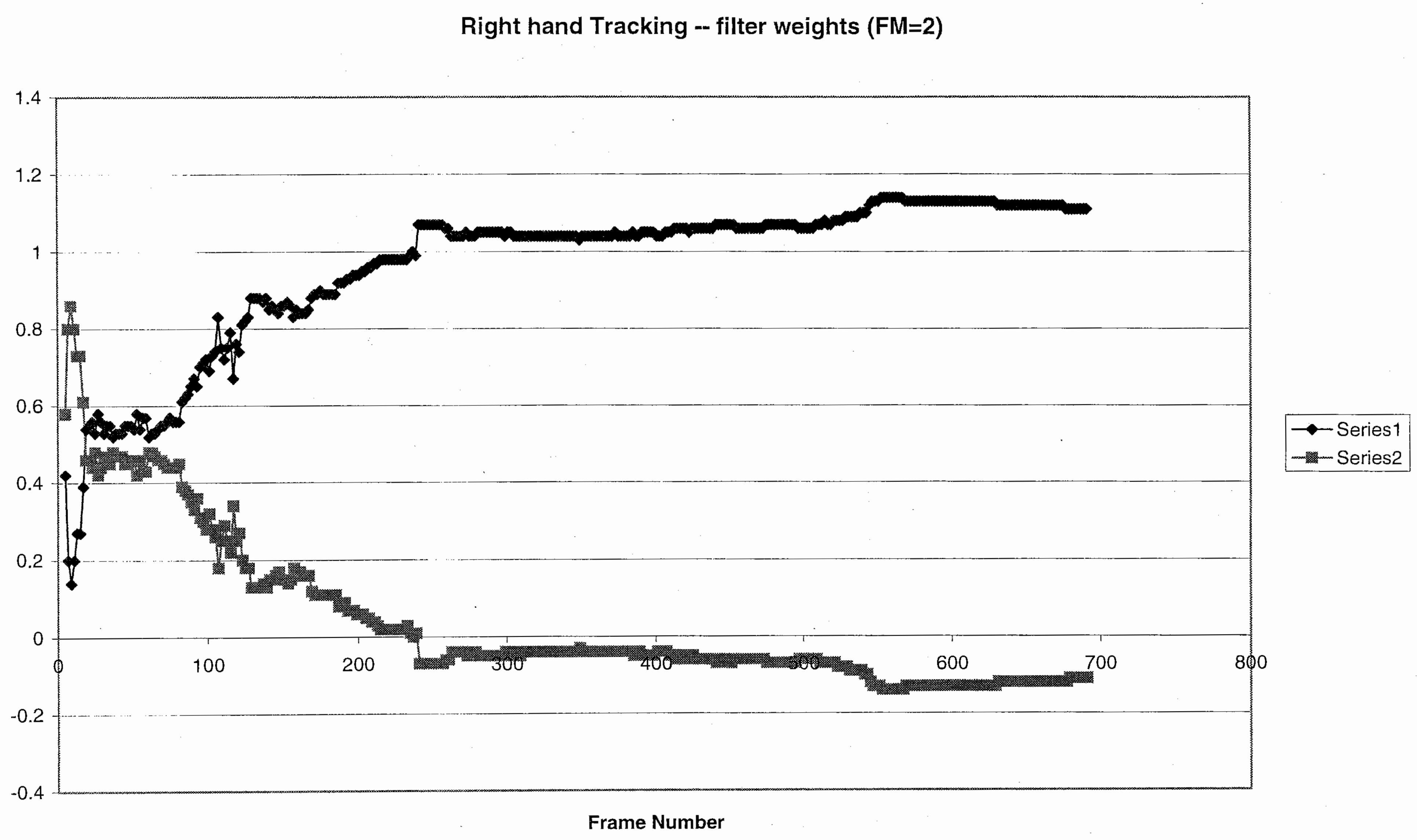
[1] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Englewood Cliffs, New Jersey, 1996.

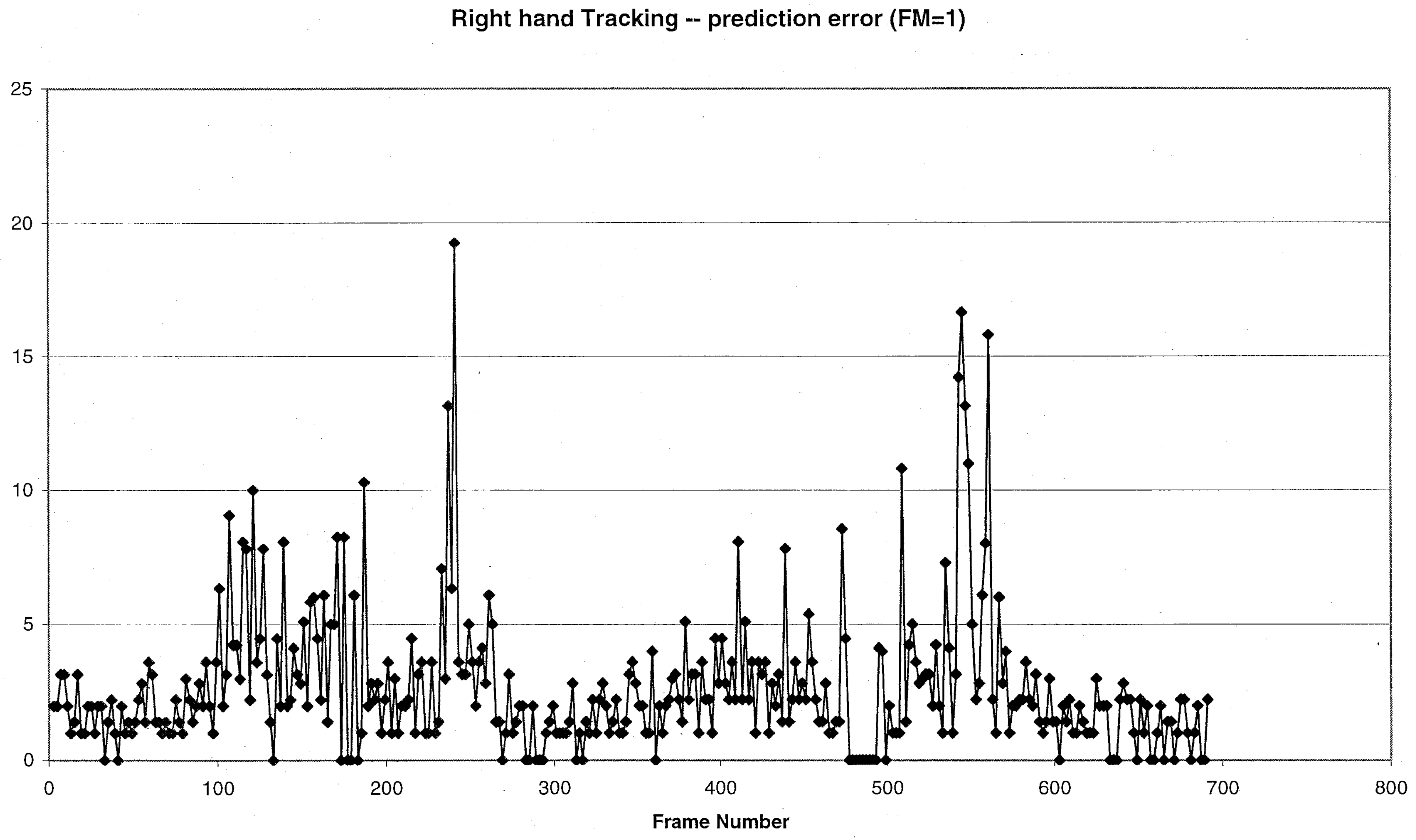
[2] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*, Academic Press, New York, NY, 1988.

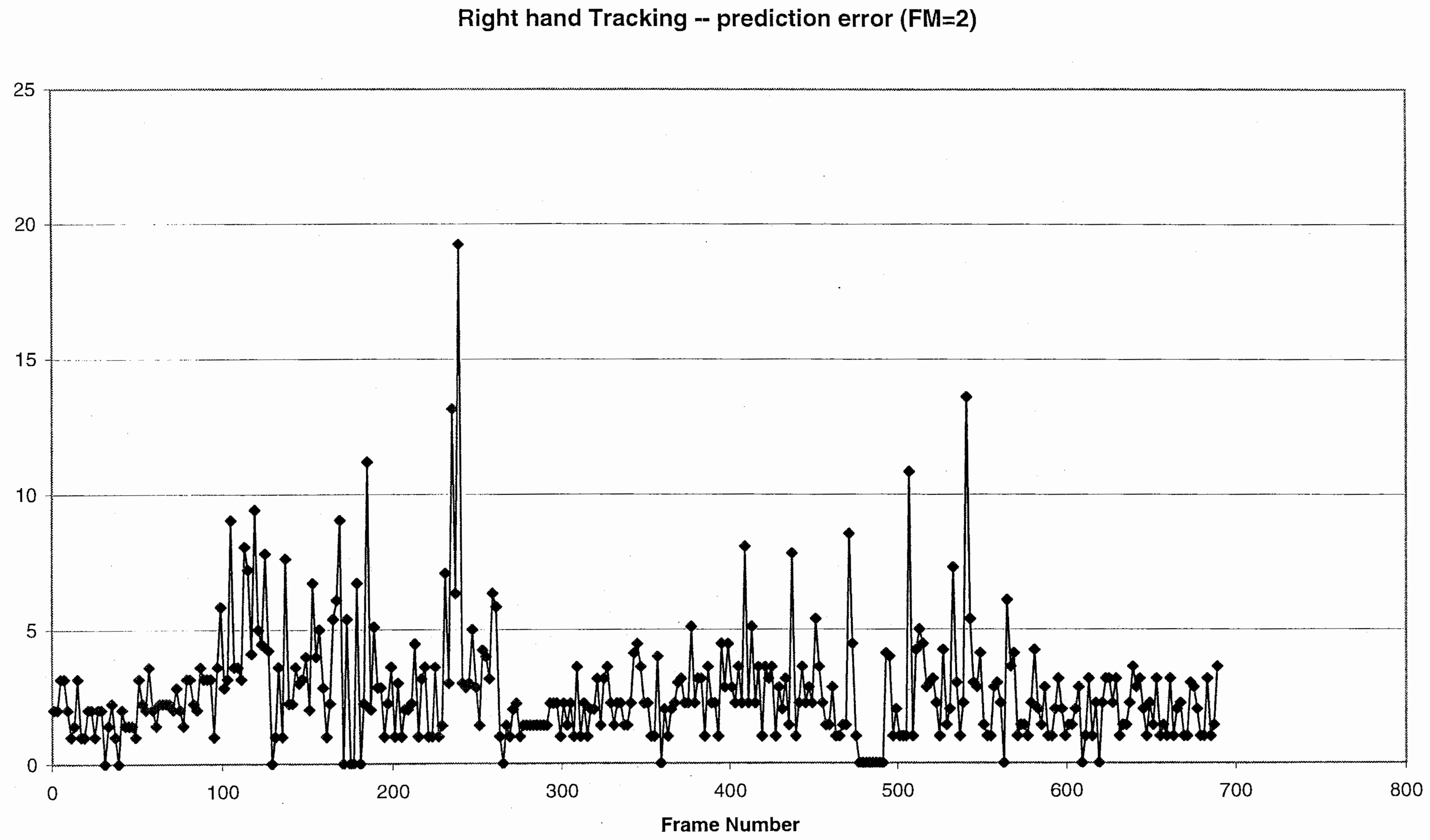
[3] A. Gelb, *Applied Optimal Estimation*, MIT Press, Cambridge MA 1974.

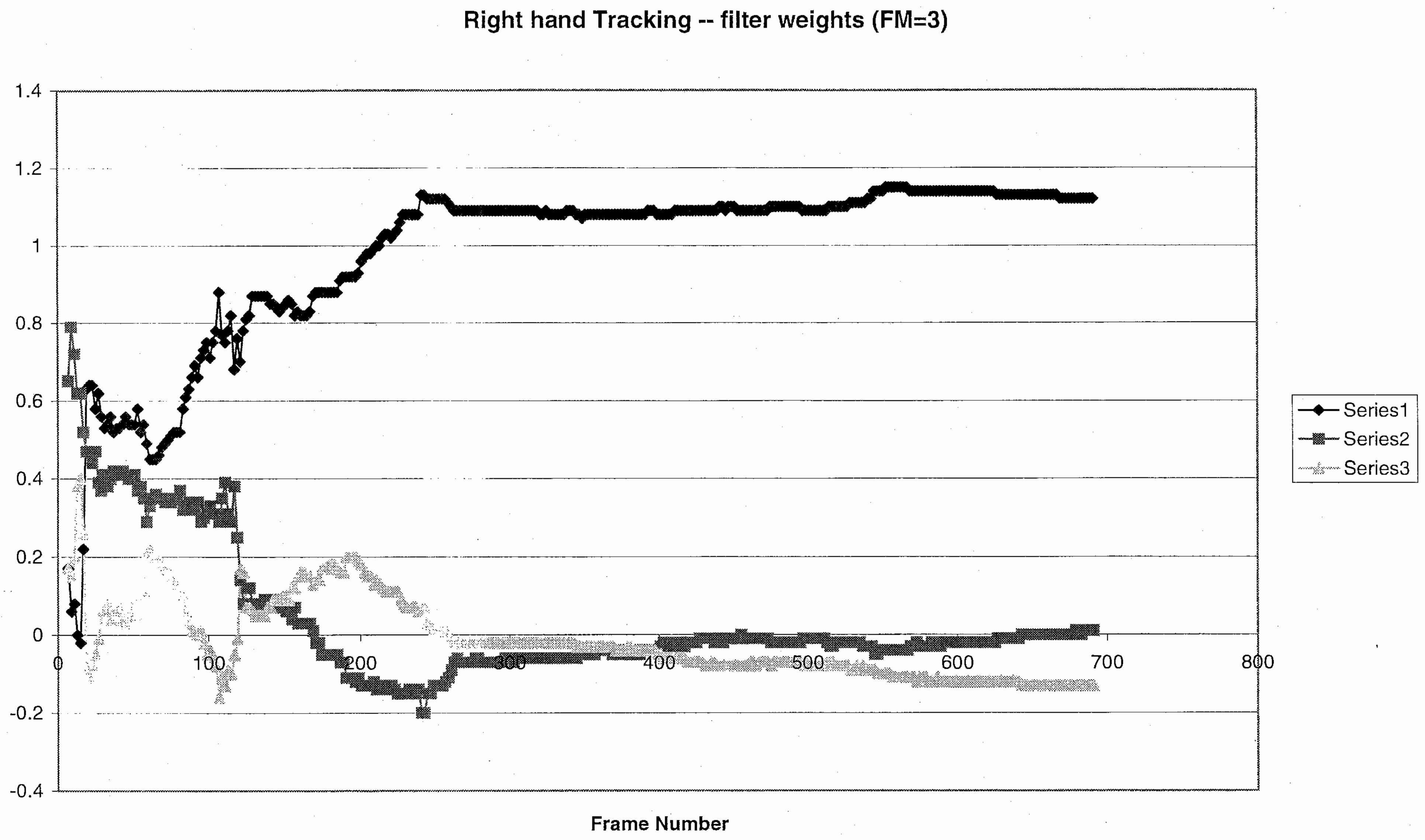
[4] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, John Wiley & Sons, New York, 1983.

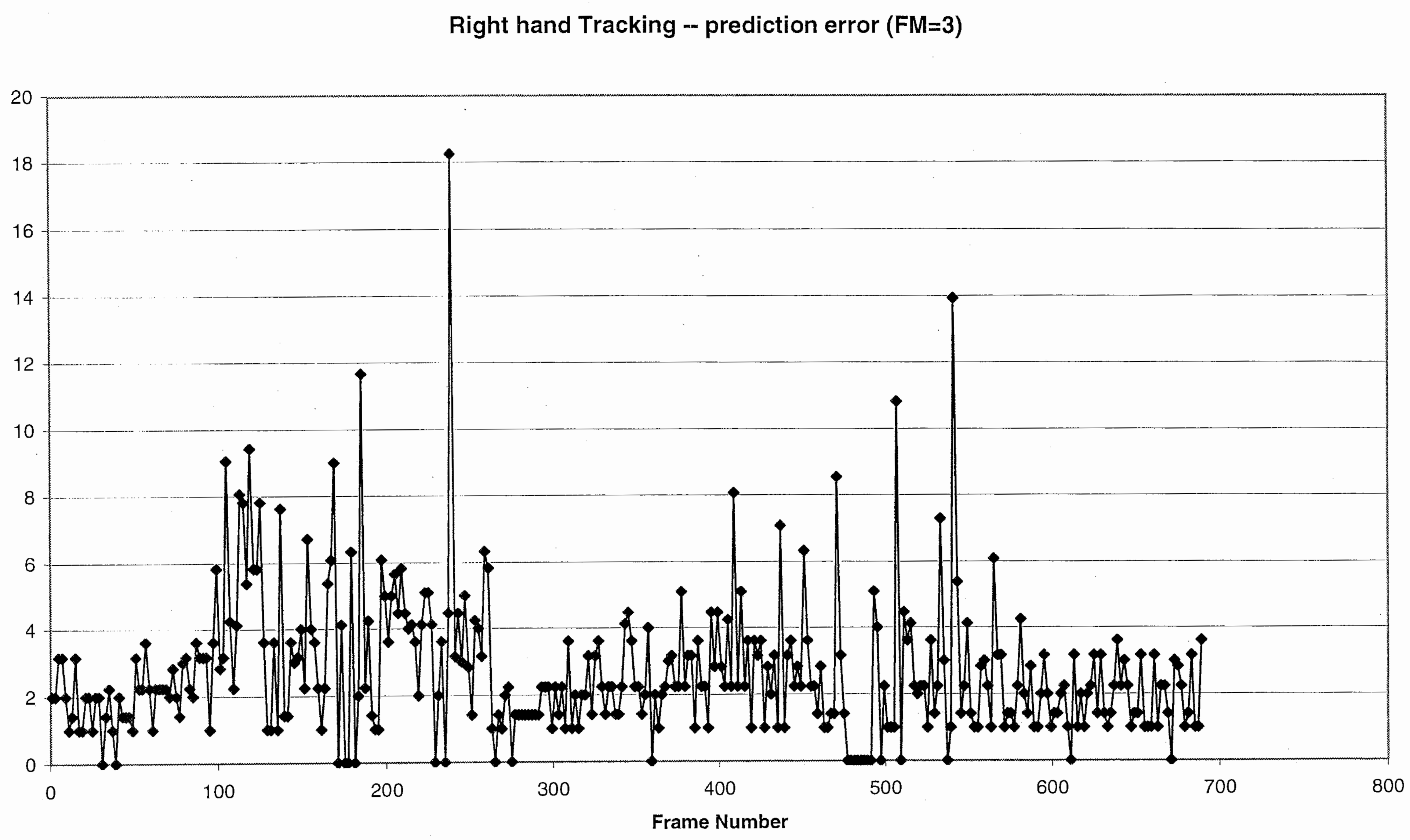
[5] Kazuhiko Takashi, Tatsumi Sakagushi, and Jun Ohya, "Real-time 3D Estimation of Human Body Postures from Trinocular Images".


Right hand Tracking -- filter weights (FM=2)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
100
200
300
400
500
600
700
800
Frame Number
Series1
Series2


Right hand Tracking -- prediction error (FM=1)
25
20
15
10
5
0
0
100
200
300
400
500
600
700
800
Frame Number


Right hand Tracking -- prediction error (FM=2)
25
20
15
10
5
0
0
100
200
300
400
500
600
700
800
Frame Number


Right hand Tracking -- filter weights (FM=3)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
100
200
300
400
500
600
700
800
Frame Number
Series1
Series2
Series3


Right hand Tracking -- prediction error (FM=3)
0
2
4
6
8
10
12
14
16
18
20
0
100
200
300
400
500
600
700
800
Frame Number