CONDENSATION-based Predictive EigenTracking

Namita Gupta[1] [*] Pooja Mittal[1] [†] Sumantra Dutta Roy[2]
Santanu Chaudhury[3] [‡] Subhashis Banerjee[4]

[1]Dept of Maths, IIT Delhi, New Delhi 110016
[2]Dept of EE, IIT Bombay, Mumbai 400076
[3]Dept of EE, IIT Delhi, New Delhi 110016
[4]Dept of CSE, IIT Delhi, New Delhi 110016

namitag@microsoft.com, pmittal@amazon.com, sumantra@ee.iitb.ac.in,
santanuc@{ee,cse}.iitd.ac.in, suban@cse.iitd.ac.in

Abstract

An appearance-based EigenTracker can track objects which simultaneously undergo image motions as well as changes in view. This paper enhances the framework in two ways. First, we incorporate a novel CONDENSATION-based predictive framework to speed up the EigenTracker. Next, our scheme is on-line: we use efficient eigenspace updates to track unknown objects. We use Importance Sampling for enhancing the capability of an EigenTracker. Our on-line EigenTracker is flexible - it is possible to use it in conjunction with other trackers in a symbiotic manner. We show results of efficient and successful tracking for two important applications – hand gesture analysis, and face and person tracking.

1. Introduction

This paper presents a novel predictive statistical framework for refining the performance of an appearance-based EigenTracker. In addition to the predictive framework, we augment the EigenTracker with a fast and efficient SVD update mechanism [2]. This enables it to learn the appearance space on the fly. We also present a new Importance Sampling mechanism – a uniformity predicate for further refinement in EigenTracking. We present results on general tracking examples, as well as on two specific examples – hand gesture analysis, and face and person tracking.

An EigenTracker [1] has an advantage over traditional feature-based tracking algorithms – the ability to track objects which simultaneously undergo affine image motions and changes in view. There has been work on other aspect of EigenTracking such as extensions for tracking of flexible objects [4], and incorporating the notion of shape in an eigenspace – Active Appearance Models (AAM) [3].

Isard and Blake [8] propose the CONDENSATION algorithm (a predictive tracker more general than a Kalman tracker) for tracking moving objects in clutter, using the conditional density propagation of state density over time. (The Appendix gives salient features of CONDENSATION and EigenTracking). In [9], the authors introduce the idea of Importance Sampling in a CONDENSATION framework. This improves the efficiency of a CONDENSATION tracker when auxiliary knowledge is available in the form of an importance function describing which areas of the state space are most informative. However, to the best of our knowledge, no related work has dealt with developing a predictive framework for EigenTracking.

The rest of the paper is organized as follows. Section 2 discusses our prediction scheme, updates, initialization issues, and the importance sampling mechanism. Here, we also describe an interesting use of our on-line EigenTracker in conjunction with other trackers. In Section 3, we show the application of the methods proposed, for hand gesture analysis, and face and person tracking.

2. An On-line Predictive EigenTracker

We enhance the existing EigenTracking framework [1] using a prediction mechanism, and an appearance space update scheme.

2.1. The Prediction Scheme

One of the main reasons for the inefficiency of the EigenTracking algorithm is the absence of a predictive framework. An EigenTracker simply finds the best reconstruction and affine coefficients *after* each frame, requiring a good seed value for the non-linear optimization. A predictive framework helps in speeding up the tracking process.

[*]current affiliation and address: 10/2449 Microsoft Corporation, One Microsoft Way, WA-98052, USA.

[†]current affiliation and address: 454B Amazon.com, 605 5th Avenue S, Seattle, WA-98104, USA.

[‡]Author for Correspondence

EigenTracking typically assumes motion to be approximated by an affine model (which takes care of effects such as rotation, translation, scaling and shear). The bounding window will thus be a parallelogram. Further, a parallelogram offers a tighter fit to the object being tracked – an important consideration for an Eigenspace-based method. We use a 6-element state vector to characterize affine motion. One can use the coordinates of three image points (any three image points form a 2-D affine basis). Alternatively, the 6 affine coefficients a_i, $i \in \{0, 5\}$ themselves can serve as elements of the state vector. In other words, $\mathbf{X} = [a_0\ a_1\ a_2\ a_3\ a_4\ a_5]^T$. These affine coefficients a_i represent the transformation of the current bounding window to the original one. A commonly used model for state dynamics is a second order AR process (t represents time): $\mathbf{X}_t = \mathbf{D}_2\mathbf{X}_{t-2} + \mathbf{D}_1\mathbf{X}_{t-1} + \mathbf{w}_t$, where $\mathbf{w}_t$ is a zero-mean, white Gaussian random vector. The particular form of the model will depend on the application – constant velocity model, random walk model, etc.

The measurement is the set of 6 affine parameters from the image $\mathbf{Z}_t = \mathbf{a}_{obs}$. Similar to [8], the observation model has Gaussian peaks around each observation, and constant density otherwise. We use a large number of representative sequences to estimate the covariances of the affine parameters obtained in a non-predictive EigenTracker. These serve as the covariances of the above Gaussian.

We use a pyramidal approach for the predictive CONDENSATION-based EigenTracker (The Appendix give a summary of CONDENSATION, and the notation used). We start at the coarsest level. Using $\{\mathbf{S}^i_{t-1}, \pi^i_{t-1}\}$ and the measurement at this level, we get $\{\mathbf{S}^i_t, \pi^i_t\}$. The affine parameter estimate at this level goes as input to the next level of the pyramid. From the estimates at the finest level, we predict the affine parameters for the next frame.

It is often not possible to learn an eigenspace representation corresponding to an object off-line. In hand tracking for example, it is not possible to learn the multitude of poses corresponding to hand gestures. One needs to build and update the eigenspace representation *on-line*. We discuss this in the following section.

2.2. On-line Eigenspace Updates

For a tracking example, the moving object often continuously encounters considerable changes in appearance. Thus, one needs to learn and update the relevant eigenspaces, on the fly. Particularly, one needs efficient incremental SVD update algorithms, since a naive $O(mN^3)$ algorithm for N images having m pixels each is not viable. For our case, we use a scale-space variant of the algorithm of Chandrasekaran *et al.* [2], which takes $O(mNk)$, for k most significant singular values.

There is a disadvantage of a naive use of learning an eigenspace representation on the fly. In some cases, the reconstruction error does not drastically increase due to the view updates. Thus, the tracker may lose the object of interest and eventually end up tracking the background. In Section 2.4, we propose a method which obliterates this problem.

ALGORITHM PREDICTIVE_EIGENTRACKER

```
A. Delineate moving object of interest
B. REPEAT FOR ALL frames:
1.  Obtain image MEASUREMENT optimizing
    affine parameters $\mathbf{a}$ and
    reconstruction coefficients $\mathbf{c}$
2.  IF using importance sampling THEN
    optimize $\tilde{\mathbf{a}}$ & $\tilde{\mathbf{c}}$ parameters in the
    'importance' eigenspace to compute
    importance MEASUREMENT
3.  ESTIMATE new affine parameters
    using output of steps 1 and 2
    (PREDICTION)
4.  FOR EACH eigenspace:
    IF reconstruction error $\in (T_1, T_2]$
    THEN update eigenspace
5.  IF ANY reconstruction error v.large
    THEN construct eigenspace afresh
```

Figure 1: Our On-line Predictive EigenTracker: An Overview

2.3. Tracker Initialization

Initializing a tracker is a difficult problem because of multiple moving objects, and background clutter. Our system performs *fully automatic initialization* under certain conditions. In general, one may use motion cues (dominant motion detection [7]. For a particular application, one may use other cues to advantage – in our hand gesture tracker for example, we augment motion cues with skin colour cues [10], [11] to segment out the moving region of interest.

2.4. An Importance Sampling Mechanism

An Importance function augments a tracker operating with one type of measurement (CONDENSATION, in this case) with information from an auxiliary measurement source [9]. The use of additional knowledge makes the system robust to failures in one measurement process. For Importance Sampling, one samples from an Importance function $g(\mathbf{X})$, rather than the state density $P(\mathbf{X})$ (Appendix).

We use a new importance sampling mechanism – the use of a uniformity predicate applied over a neighbourhood. The aim of this approach is to segment out the region of interest using another source of measurement. There is a justification for using an auxiliary measurement source as an additional source of measurement, rahter than incorporating it in the original eigenspace. Each measurement source is associated with its own characteristics and limitations. Our framework of using an auxiliary independent measurement augments the original, and enhances the reliability of the tracker. As an example, let us consider a non-convex shape enclosed in a bounding parallelogram. A pure Eigentracker will have problems with changing backgrounds in the bounding parallelogram. This is especially important in the context of incrementally constructing the eigenspace, on the fly. Using only texture- or colour-based properties, for example may give an improper segmentation. A combination of the two in an importance sampling framework can result in more reliable tracking.

We use a uniformity predicate to form a new 'Importance' Eigenspace – representing a view of the object of interest in a frame, with the background eliminated. We optimize the $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{c}}$ parameters of the Importance eigenspace to obtain the Importance measurement. This is the set of affine parameters $\tilde{\mathbf{a}}_{obs}$ which transforms the current bounding parallelogram to the original bounding box, such that the reconstruction error is minimum.

2.5. The Overall Tracking Scheme

We present our overall tracking scheme in this section. Figure 1 outlines the main steps. Step A is the tracker initialization (Section 2.3). We now predict affine parameters – a parallelogram bounding box for the next frame (Step 1 in Figure 1, details in Section 2.1). The next step (Step 2 in Figure 1) is obtaining measurements (of the affine parameters) from the image – an optimization of the affine parameters $\mathbf{a}$ and the eigenspace reconstruction coefficients $\mathbf{c}$ (EigenTracker: Appendix). In Section 2.4, we further enhance the algorithm to take advantage of different input measurement sources using importance sampling. This uses the Importance Eigenspace, with its corresponding affine and eigenspace reconstruction parameters, $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{c}}$. The prediction step consists in estimating the new affine parameters, using the output of the previous two steps. Depending on the reconstruction error, it decides on whether or not to perform an eigenspace update. If any of the two reconstruction errors is very large, this indicates a new view of the object. The algorithm recomputes a new bounding box and starts rebuilding the eigenspace (Step 5 in Figure 1). It then repeats the above steps for the next frame.

2.6. Using Other Trackers in Tandem

A simple variant of our EigenTracking framework has an on-line EigenTracker working in conjunction with another tracker. We can thus take advantage of a tracker tracking the same object, using a different measurement process, or tracking principle. The EigenTracker works synergistically

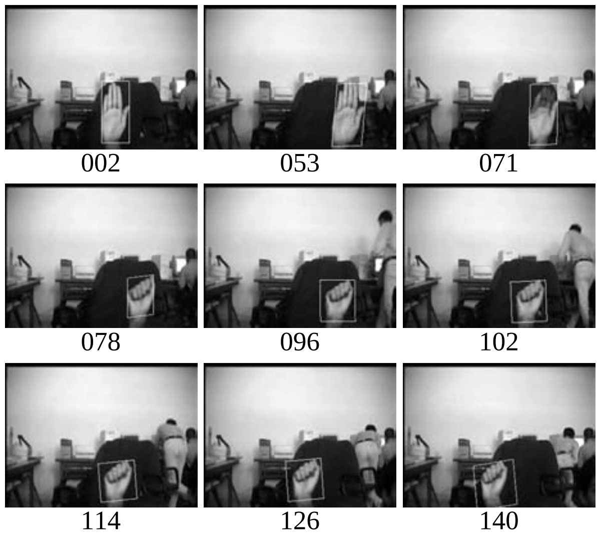


Figure 2: Predictive EigenTracking results: frames (in row major order, frame numbers under each frame) showing successful tracking in spite of background clutter, and other moving objects present in the scene.

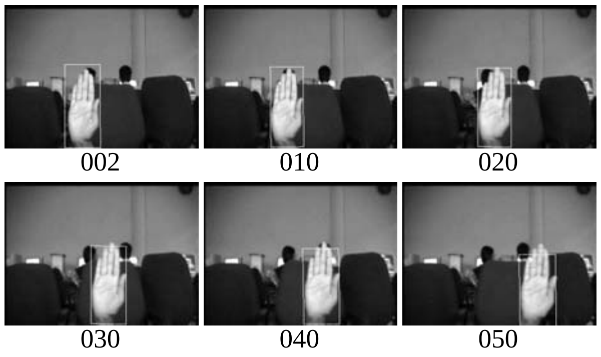
with the other tracker, using it to get its affine parameters. It then optimizes these parameters, and proceeds with the EigenTracking. Such a synergistic combination endows the combined tracker with the benefits of both the EigenTracker as well as the other one – tracking the view changes of an object in a predictive manner. As an example, we have experimented with using a CONDENSATION tracker and an EigenTracker for cases of restricted affine motions – rotation, translation and scaling (details in Section 3.1.2).

3. Applications

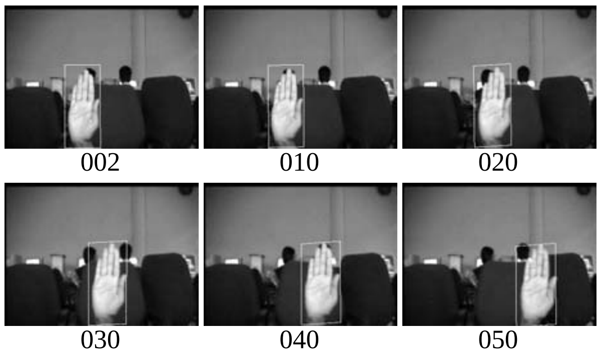
In this section, we study two important applications of our approach namely, gesture analysis, and face and person tracking. Our tracker runs on a 700MHz PIII machine running Linux. In [6], we present some preliminary results of predictive EigenTracking for tracking a moving hand. Our predictive EigenTracker forms the basis of a gesture-based interface [5]. (All videos corresponding to this paper: `http://www.ee.iitb.ac.in/∼sumantra/mpg`)

3.1. Hand Gesture Tracking

First, we show the result of an experiment on a typical hand gesture sequence (Figure 2). This example shows successful tracking of a variety of hand poses. The tracker is not distracted by either background clutter, or other moving objects. Figure 3(b) compares results obtained using the predictive EigenTracker with those corresponding to a non-predictive version (Figure 3(a)). The average number of iterations (for the optimization) improves from 3.5 to 2.9.



(a) Non-predictive EigenTracker



(b) Predictive EigenTracker

Figure 3: Tracking results with (a) a simple EigenTracker, as compared with (b) results of our predictive EigenTracker (bottom row): some representative frames. The hand is not properly tracked using the former.

For the face tracking example in Figure 8 (b), this improves from 12.8 to 12.3. A comparison of a non-predictive EigenTracker with a predictive on for the sequence shown in Figure 5 shows a drastic imporvement in the average number of iterations - from 7.44 to 4.67.

3.1.1 Incorporating our Importance Sampling Mechanism (Section 2.4)

The importance function (Section 2.4) works on the segmented appearance with only the object, eliminating the background. Hence, such an approach eliminates the problem of changing backgrounds for an EigenTracker. the pers such as [10] show that human skin colour occupies a small portion of the entire colour space. For colour $C = [C_b C_r]^T$ in the YC_bC_r colour space, we learn two likelihood functions $P(C|skin)$ and $P(C|not\ skin)$. We can show that n (a number, based on the colour C of a pixel) $\propto \frac{P(skin|C)}{P(not\ skin|C)}$, [11]. We consider those pixels corresponding to the top $p\%$ values of n as skin-coloured pixels.

For a hand tracking application for example, we use human skin colour [10] to segment the hand from the background. For the eigenspace, we consider a vector $\tilde{\mathbf{I}}$ containing the intensity values of only the hand pixels in the bounding window, and background pixels masked out. (As in Figure 4) This is especially evident for cases such as Figure 5 where the background constitutes a large component of an image of an open hand (a non-convex object). The entire hand is better tracked in the latter case.

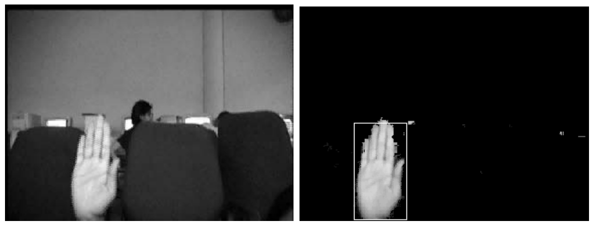
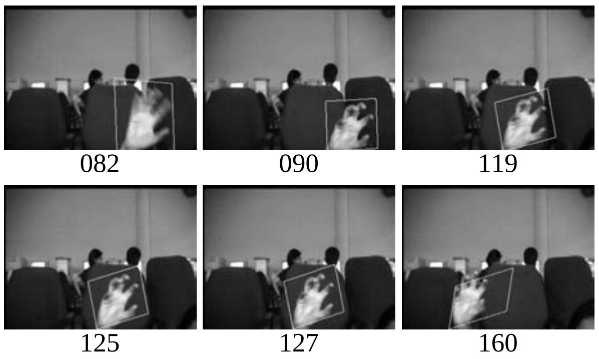
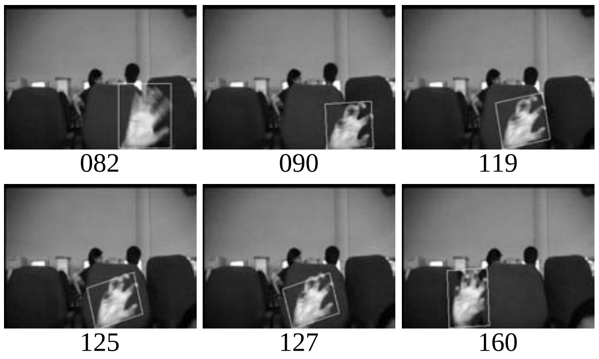


Figure 4: Importance sampling for a hand tracking application: a video frame - the original, and with the skin coloured region segmented out (details in text).



(a) Without Importance Sampling



(b) Using Importance Sampling

Figure 5: Incorporating our Importance Sampling mechanism results in better tracking of the open hand

3.1.2 Synergistic Conjunction with Other Trackers: Restricted Affine Motion (Section 2.6)

We now show experiments of using an SVD update-based multi-resolution EigenTracker with a skin colour-based CONDENSATION tracker [11]. The latter considers the parameters of a rectangular window bounding the moving hand as state vector elements – its centroid, the height and the width. The observation is also a 4-element state vector, consisting of the rectangular window parameters of the largest skin blob. The state dynamics considers a constant velocity model for the centroid position, and a constant position one for the other two parameters.

We use the tracking parameters obtained from the CONDENSATION skin tracker for each frame, to estimate the affine parameters for the the Appearance tracker. The appearance tracker then does the fine adjustments of the affine parameters and computes the reconstruction error. We first consider a restricted case of affine transformations – scaling and translation alone (Figure 6). The processing time per frame is 100–180ms when it can track at the coarsest level itself, and 600–900ms when it goes to the finest level (image size 320×240). This experiment shows that having even a very simple restricted affine model overcomes an inherent problem with the EigenTracker of being able to track motion up to only a few pixels.

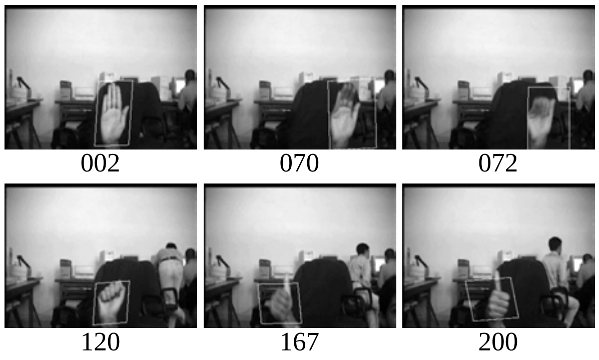


Figure 6: A simple synergistic combination of a CONDENSATION skin tracker with an online EigenTracker: scaling and translation. Details in Section 2.6

We extend the previous scheme to cover rotations as well. We first compute the principal axis of the pixel distribution of the best fitting blob. We align the principal axis with the vertical Y-axis and compute the new width, height and centroid. These parameters give us the restricted affine matrix (scaling, rotation, translation):

$$\mathbf{A}_{restricted} = Inv(\mathbf{SRT})$$

When applied to the current image, these parameters take it to the first bounding window of the CONDENSATION skin tracker. In Figure 7 we show results of this approach. This scheme allows tracking of large rotations (as evident in Figure 7). We get a better fitting window and less background pixels, leading to lower eigenspace reconstruction error. The average processing time per frame is 900ms.

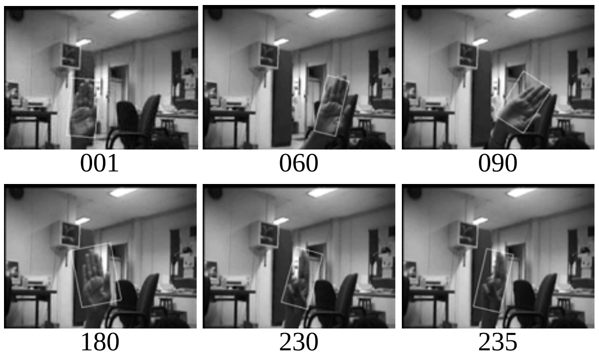


Figure 7: Synergistically using an online EigenTracker in conjunction with a skin colour-based CONDENSATION tracker: rotation, translation, scaling. Details in Section 2.6

3.2. Face Tracking, Person Tracking

In this section, we show examples of our importance sampling method for tracking faces and persons across frames in video sequences. In Figure 8 (a) and (b), we use a skin colour-based importance function again, to track a face across frames. In this figure, the person is not only moving, he is also rotating his head. As a result, the appearance of the face undergoes considerable change throughout the sequence. Due to this reason, the predictive EigenTracker without importance sampling fails after 10 frames, as shown. However, the one with importance sampling is able to track correctly for the entire length of 44 frames – a drastic improvement.

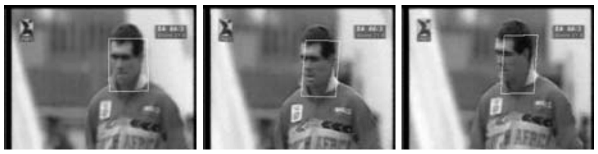
For the next example, (Figure 8) we track a moving person across frames in a movie sequence. The uniformity predicate is based on the colour of the person's shirt. Our system is able to track the moving person in spite his moving against a textured background of a similar colour (as in the last two images in Figure 8). In this case, an EigenTracker based on a colour predicate alone would have failed because the background colour is similar to that of the object being tracked. However, a combination of appearance and a colour-based importance predicate enables the object to be tracked correctly.

In this paper, we show examples using colour-based information. One may use any uniformity predicate to segment out the region of interest – colour or texture for example, for use in an importance function.

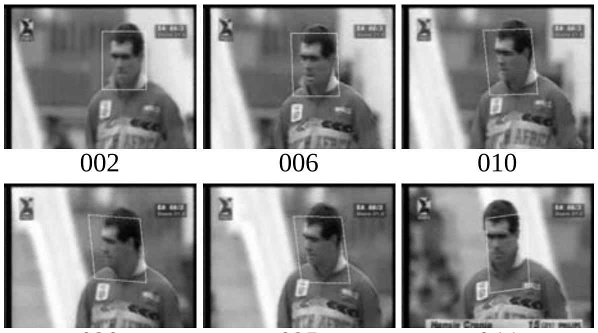
Appendix: CONDENSATION, EigenTracking

The CONDENSATION algorithm [8] represents the state conditional density by a sample set of N states, $\mathbf{S}_t = \{\mathbf{s}_t^i\}$ and a corresponding set of weights $\mathbf{\Pi}_t = \{\pi_t^i\}$, $i \in \{1, n\}$. The algorithm makes use of the principle of factored sampling. This enables the generation of a sample set of

FACE TRACKING IN A SPORTS VIDEO



(a) Without Imp. Sampling: failure at the 10th frame



(b) Using Importance Sampling

PERSON TRACKING

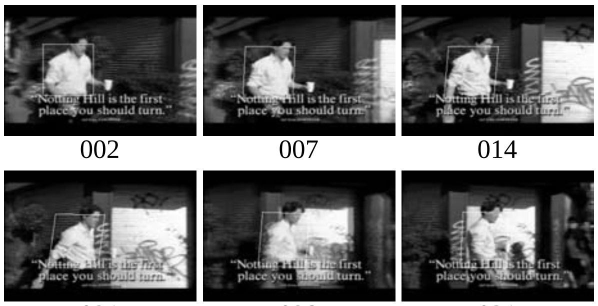


Figure 8: Using our Importance Sampling mechanism (Section 2.4) for two applications: Face tracking in a sports video, and person tracking in a movie sequence.

states, given the weights representing the conditional observation probability density. The CONDENSATION algorithm needs:
1. a model for the system state $\mathbf{X}$,
2. a state dynamics model $P(\mathbf{X}_t|\mathbf{X}_{t-1})$, and
3. a model for an observation $\mathbf{Z}$: $P(\mathbf{Z}_t|\mathbf{X}_t)$

An EigenTracking approach [1] involves estimating the view of the object (using the eigenspace), as well as the transformation that takes this view into the given image (modeled as a 2-D affine transformation). Black and Jepson pose the problem as finding affine transformation coefficients $\mathbf{a}$ ($= [a_0\ a_1\ a_2\ a_3\ a_4\ a_5]^T$) and the eigenspace reconstruction coefficients $\mathbf{c}$, such that the robust error function between the parameterized image $\mathbf{I}$ (indexed by its pixel location $\mathbf{x}$) and the reconstructed one $\mathbf{Uc}$ (where $\mathbf{U}$ is the matrix of the most significant engenvectors) is minimum, for all pixel positions $\mathbf{x} = [x\ y]^T$:

$$arg\ min_{\forall \mathbf{x}}, \rho(\mathbf{I}(\mathbf{x} + \mathbf{f}(\mathbf{x}, \mathbf{a})) - [\mathbf{Uc}](\mathbf{x}), \sigma) \tag{1}$$

Here, $\rho(x, \sigma) = x^2/(x^2 + \sigma^2)$ is a robust error function, and σ is a scale parameter. The 2-D affine transformation is given by

$$\mathbf{f}(\mathbf{x}, \mathbf{a}) = \begin{bmatrix} a_0 \\ a_3 \end{bmatrix} + \begin{bmatrix} a_1 & a_2 \\ a_4 & a_5 \end{bmatrix} \mathbf{x} \tag{2}$$

References

[1] M. J. Black and A. D. Jepson. EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation. *International Journal of Computer Vision*, 26(1):63 – 84, 1998.

[2] S. Chandrasekaran, B. S. Manjunath, Y. F. Wang, J. Winkeler, and H. Zhang. An Eigenspace Update Algorithm for Image Analysis. *Graphical Models and Image Processing*, 59(5):321 – 332, September 1997.

[3] T. Cootes, G. J. Edwards, and C. Taylor. Active Appearance Models. In *Proc. European Conference on Computer Vision (ECCV)*, 1998.

[4] F. De la Torre, J. Vitria, P. Radeva, and J. Melenchon. Eigenfiltering for Flexible Eigentracking (EFE). In *Proc. International Conference on Pattern Recognition (ICPR)*, pages III:1118 – 1121, 2000.

[5] N. Gupta, P. Mittal, S. Dutta Roy, S. Chaudhury, and S. Banerjee. Developing a gesture-based interface. *IETE Journal of Research: Special Issue on Visual Media Processing*, 2002. (Accepted for Publication).

[6] N. Gupta, P. Mittal, S. Dutta Roy, S. Chaudhury, and S. Banerjee. A Predictive Scheme for Appearance-based Hand Tracking. In *Proc. National Conference on Communications (NCC)*, pages 513 – 522, 2002.

[7] M. Irani, B. Rousso, and S. Peleg. Computing Occluding and Transparent Motions. *International Journal of Computer Vision*, 12(1):5 – 16, January 1994.

[8] M. Isard and A. Blake. CONDENSATION - Conditional Density Propagation For Visual Tracking. *International Journal of Computer Vision*, 28(1):5 – 28, 1998.

[9] M. Isard and A. Blake. ICONDENSATION: Unifying Low-level and High-level Tracking in a Stochastic Framework. In *Proc. European Conference on Computer Vision (ECCV)*, pages 893 – 908, 1998.

[10] R. Kjeldsen and J. Kender. Finding Skin in Color Images. In *Proc. Intl. Conf. on Automatic Face and Gesture Recognition*, pages 312 – 317, 1996.

[11] J. Mammen, S. Chaudhuri, and T. Agrawal. Tracking of both hands by estimation of erroneous observations. In *Proc. British Machine Vision Conference (BMVC)*, 2001.