A PREDICTIVE SCHEME FOR APPEARANCE-BASED HAND TRACKING

N. Gupta[1], P. Mittal[1], S. Dutta Roy[2], S. Chaudhury[3], S. 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

Abstract

Hand gestures are difficult to track across views – there is a change in position as well as the view of the moving hand. Further, it is not possible to learn all hand appearances in an off-line phase. This paper presents a novel predictive framework for such an application. It is based on the principle of Eigentracking, with a predictive component to speed up the tracking process. Additionally, our tracker can learn the eigenspace on the fly. We show results of efficient and successful tracking even in cases of background clutter and other moving objects.

1. INTRODUCTION

The use of hand gestures is an important technique for Human-Computer Interaction (HCI) [1]. Hand gesture analysis involves both spatial as well as temporal processing of image frames. Tracking the moving hand across frames is an important part of such a system. This is a difficult task – we only have 2-D images of the human hand, a non-rigid 3-D object. Using elaborate 3-D models involves dealing with high-dimensional parameter spaces. Further, estimating 3-D parameters from 2-D images is also very difficult [1].

Isard and Blake [2] 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. An Eigentracker [3] has an advantage over traditional feature-based tracking algorithms – the ability to track objects which simultaneously undergo affine image motions and changes in view (the Appendix gives salient features of CONDENSATION and Eigentracking).

This paper removes a serious restriction of the Eigentracker framework - the absence of a predictive component. We develop a novel predictive Eigentracker for use in hand gesture tracking. We have an automatic initialization process – our tracker does not have to be bootstrapped. For a hand gesture tracking applications, it is not feasible to learn an eigenspace representation off-line. Our approach uses combines a predictive Eigentracker with efficient eigenspace update methods.

2. A PREDICTIVE EIGENTRACKER FOR HAND GESTURES

In this section, we first introduce a predictive framework in an Eigentracker. Next, we describe on-line Eigenspace updates. The next section focuses on automatic initialization of the hand tracker. Finally, we give an overview of the entire scheme.

2.1. Incorporating a Predictive Framework

One of the main reasons for the inefficiency of the Eigentracking algorithm is the absence of a predictive framework. An Eigentracker simply updates the eigenspace and affine coefficients *after* each frame, requiring a good seed value for the non-linear optimization in each case. A predic-

tive framework helps in speeding up the tracking process. This is one of the main contributions of this paper. In this section, we describe models for the tracker's system state, state dynamics, and the measurement (observation).

We assume that the hand motion can be approximate by an affine model (which takes care of effects such as rotation, translation, scaling and shear). The bounding window will thus be a parallelogram. This is more general than a rectangular bounding window. 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: $\mathbf{X}_t = \mathbf{D}_2\mathbf{X}_{t-2} + \mathbf{D}_1\mathbf{X}_{t-1} + \mathbf{w}_t$, 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 [2], 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 measurements are made at each level of the pyramid. 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 not feasible to learn the multitude of poses corresponding to hand gestures, even for one particular person. One needs to build and update the eigenspace representation *on-line*. We discuss this in the following section.

2.2. On-line Eigenspace Updates

In hand gesture tracking, the moving hand 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.* [4], which takes $O(mNk)$, for k most significant singular values.

3. TRACKER INITIALIZATION

Initializing a tracker is a difficult problem because of multiple moving objects, and background clutter. In other words, one needs to segment out the moving region of interest from the possibly cluttered background in the frames. Our hand gesture tracker performs *fully automatic initialization*. We use a combination of motion cues (dominant motion detection [5] as well as skin colour cues [6], [7] to identify the region of interest in each frame.

4. 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 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 is obtaining measurements (of the affine parameters) from the image – an optimization of the affine parameters $\mathbf{a}$ and the eigenspace reconstruction coefficients $\mathbf{c}$ (Appendix). Depending on the reconstruction error, it decides on whether or not to perform an eigenspace update (Section 2.2). 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

ALGORITHM PREDICTIVE_EIGENTRACKER

A. Delineate moving hand
B. REPEAT FOR ALL frames:
1. PREDICT affine parameters
2. Obtain image MEASUREMENT optimizing affine parameters **a** and reconstruction coefficients **c**
3. ESTIMATE new affine parameters using output of step 2
4. IF reconstruction error $\in (T_1, T_2]$ THEN update eigenspace
5. IF reconstruction error very large THEN construct eigenspace afresh

Figure 1: Our Predictive Eigentracker for Hand Gestures: An Overview

the eigenspace (Step 5 in Figure 1). It then repeats the above steps for the next frame.

4.1. 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 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 (Section 5.1).

5. RESULTS AND DISCUSSION

We now give results of using our tracking algorithm on various hand gesture sequences. Our tracker runs on a 700MHz PIII machine running Linux. 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.

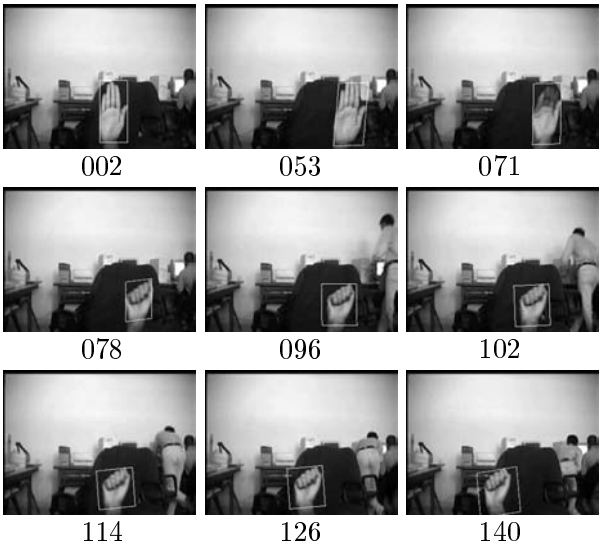
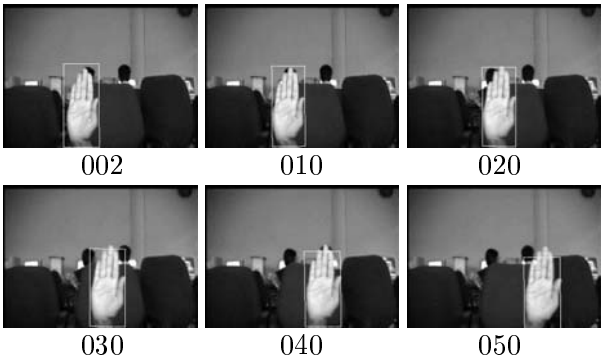


Figure 2: Predictive Eigentracking results: frames (in row major order, frame numbers under each frame) showing successful tracking in spite of background clutter, and other moving objects present in the scene.

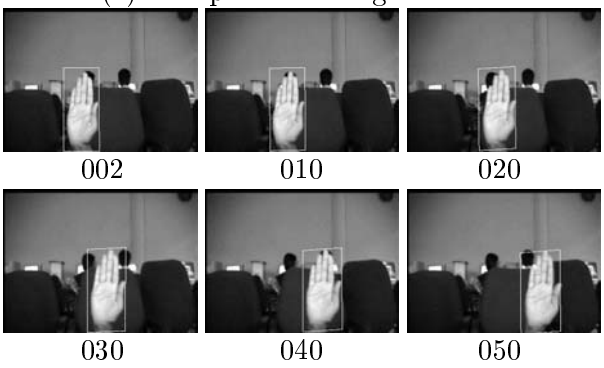
Figure 3(b) compares results obtained using the predictive Eigentracker with those corresponding to a non-predictive version (Figure 3(a)). The average number of iterations (for the optimization) improves from 3.5 to 2.9. A comparison of a non-predictive Eigentracker with a predictive one for the sequence in Figure 4 shows a drastic improvement in the average number of iterations – from 7.44 to 4.67.

5.1. Synergistic Conjunction with Other Trackers: Restricted Affine Motion

We now show experiments of using an SVD update-based multi-resolution Eigentracker with a skin colour-based CONDENSATION tracker [7]. 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

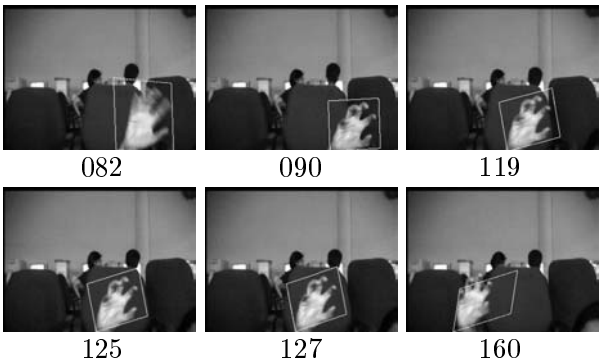


(a) Non-predictive Eigentracker

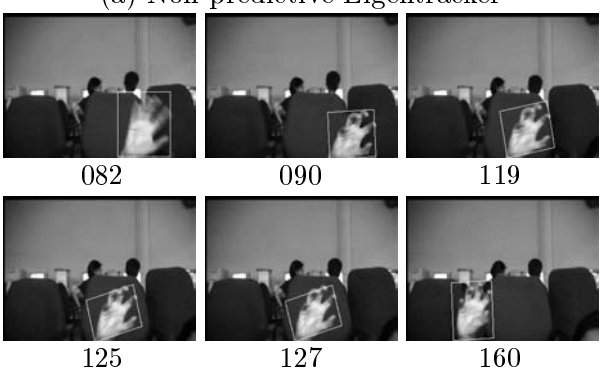


(b) Predictive Eigentracker

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.



(a) Non-predictive Eigentracker



(b) Predictive Eigentracker

Figure 4: Non-predictive Eigentracking versus our Predictive framework: another example

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 5). 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.

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 6 we show results of this approach. This scheme allows tracking of large rotations (as evident in Figure 6). We get a better fitting window and less background pixels, leading to lower eigenspace reconstruction error. The average processing time per frame is 900ms.

Appendix: CONDENSATION and Eigentracking

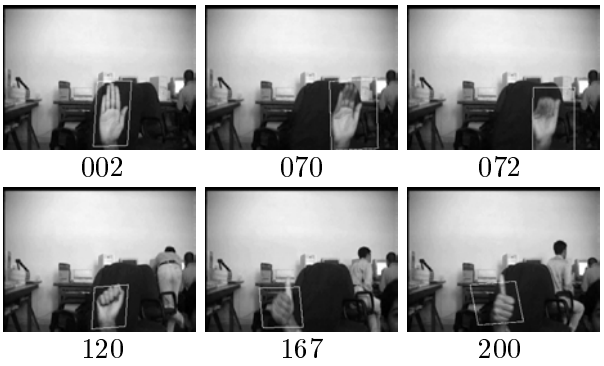


Figure 5: A simple combination of a CONDENSATION skin tracker with an online Eigentracker: scaling and translation. Details in Section 4.1

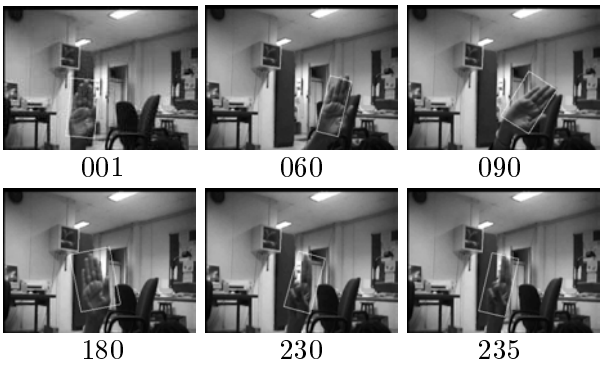


Figure 6: Using an online Eigentracker in conjunction with a skin colour-based CONDENSATION tracker: rotation, translation, scaling. Details in Section 4.1

The CONDENSATION algorithm [2] 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. 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 [3] 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 and the reconstructed one is minimum, for all pixel positions $\mathbf{x} = [x\ y]^T$:

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

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

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

REFERENCES

[1] V. I. Pavlovic, R. Sharma, and T. S. Huang, "Visual Interpretation of Hand Gestures for Human-Computer Interaction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 677 – 695, July 1997.

[2] M. Isard and A. Blake, "CONDENSATION - conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 28, no. 1, pp. 5 – 28, 1998.

[3] M. J. Black and A. D. Jepson, "Condensation - conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 26, no. 1, pp. 63 – 84, 1998.

[4] 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*, vol. 59, no. 5, pp. 321 – 332, September 1997.

[5] M. Irani, B. Rousso, and S. Peleg, "Computing Occluding and Transparent Motions," *International Journal of Computer Vision*, vol. 12, no. 1, pp. 5 – 16, January 1994.

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

[7] J. P. Mammen, "Hand Tracking and Gesture Recognition," M. Tech thesis, Department of Electrical Engineering, IIT Bombay, 2000.