Developing a Gesture-based Interface

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

A gesture-based interface involves tracking a moving hand across frames, and extracting the semantic interpretation corresponding to the gesture. This is a difficult task, since there is a change in both the position as well as the appearance of the hand. Further, such a system should be robust to the speed at which the gesture is performed. This paper presents a novel attempt at developing a hand gesture-based interface. We propose an on-line predictive EigenTracker for the moving hand. Our tracker can learn the eigenspace on the fly. We propose a new state-based representation scheme for hand gestures, based on the eigenspace reconstruction error. This makes the system independent of the speed of performing the gesture. We use learning for adapting the gesture recognition system to individual requirements. We show results of successful operation of our system even in cases of background clutter and other moving objects.

1. Introduction

The use of hand gestures provides an attractive alternative to cumbersome interface devices for Human-Computer Interaction (HCI) [10]. Hand gesture analysis involves both spatial as well as temporal processing of image frames. Two important components of the above task are the tracking of the moving hand across frames, and extracting the semantic interpretation corresponding to the gesture. Each one is a difficult task. There is a loss in information due to the projection of the 3-D human hand to the 2-D image plane. Elaborate 3-D models have prohibitive high-dimensional parameter spaces. Further, estimating 3-D parameters from 2-D images is also very difficult [10]. The tracker also has to handle changing shapes, other moving objects, and noise (as in Figure 1). The difficulty of the second task is compounded by different factors – hand shapes and sizes vary from individual to individual. Thus, this is a serious problem for recognizing static hand gestures alone. For a dynamic hand gesture, different people may perform the same gesture in different periods of time.

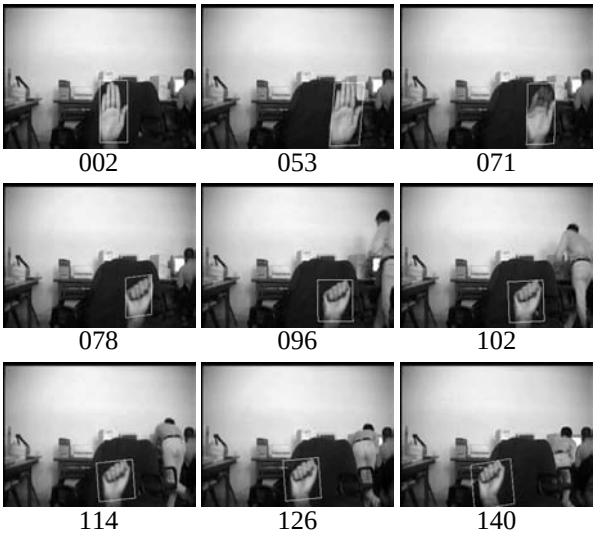


Figure 1: A set of representative frames from a hand gesture analysis (frames in row-major order, with frame numbers). (Details in text).

Pavlovic, Sharma and Huang [10] present an extensive review of hand gesture interpretation techniques. Bobick and Wilson [3] propose a state-based technique for representation and recognition of gestures in which they define a gesture as a sequence of states in a measurement or configuration space. A HMM is a possible tool modeling the spatial and temporal nature of a gesture [10], [1], [11]. Yeasin and Chaudhuri [12] model the temporal signature of a hand ges-

*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.

ture as a finite state machine.

In this paper, we have proposed a hand gesture based interface system which uses hand tracking and changes in hand shapes for the purpose of associating semiotics to the gesture. Isard and Blake [6] propose the CONDENSATION algorithm (a predictive tracker more general than a Kalman tracker) for tracking moving objects, including hand, in clutter, using the conditional density propagation of state density over time. An EigenTracker [2] 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). An important lacuna of EigenTracking is the absence of a predictive framework. This paper removes a serious restriction of the EigenTracker framework – the absence of a predictive framework. We develop a novel predictive EigenTracker with efficient eigenspace update methods – it can learn the eigenspace representation on the fly. We have an automatic initialization process for the tracker – it does not need to be bootstrapped. This learning-and-tracking of changing hand shapes fits in with our gesture recognition framework. We express a gesture as a combination of different *epochs*, corresponding to eigenspace representations of static hand shapes, and their temporal relationships. The system goes through the same set of states, whether the gesture is performed slowly or done fast. We use a shape-based state identification scheme. The identification scheme makes use of hand shapes of individuals (corresponding to different states) learnt *a priori*.

The rest of the paper is organized as follows. Section 2 presents our predictive EigenTracker, with on-line eigenspace updates, and automatic initialization. We use this predictive Eigentracker to track the motion of the hand. Next, we discuss our gesture recognition framework. This framework uses information from the predictive EigenTracker. In each case, we present results of experimentation with our system.

2. A Predictive EigenTracker for Hand Gestures

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. We use a predictive framework to speed up the EigenTracker. We incorporate a prediction of the position of the object being tracked, using a CONDENSATION-based algorithm. We describe our model for the system state, dynamics and measurement (observation) as follows.

The hand motion between frames has effects such as rotation, translation, scaling and shear – which can be accounted for by an affine model. The shape of the bounding window for the hand will be a parallelogram. This is consistent with the affine motion model. Further, a parallelogram offers a tighter fit to the object being tracked (further reducing the effect of the background) – an important consideration for an Eigenspace-based method. A 6-element state vector characterizes affine motion. One can use the coordinates of three image points (any three image points form a 2-D affine basis). The affine parameters represent the parallelogram bounding the hand shape in each frame. Alternatively, the 6 affine coefficients a_i ($0 \leq i \leq 5$) 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 [6], 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 learn and update the relevant eigenspaces, on the fly. We discuss this in the following section.

2.1. On-line Eigenspace Updates

In a hand gesture, the appearance of the hand often changes considerably. One needs to build and update the eigenspace representation efficiently, *on-line*. A naive $O(mN^3)$ algorithm for N images having m pixels each is computationally inefficient. Particularly, one needs efficient incremental SVD update algorithms, to update the eigenspace at each frame. 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.

ALGORITHM PREDICTIVE_EIGENTRACKER

```
A. Delineate moving hand
B. REPEAT FOR ALL frames:
1.  Obtain image MEASUREMENT optimizing
    affine parameters a and
    reconstruction coefficients c
2.  ESTIMATE new affine parameters
    from step 1 output (PREDICTION)
3.  IF reconstruction error ∈ (T1, T2]
    THEN update eigenspace
4.  IF reconstruction error very large
    THEN construct eigenspace afresh
```

Figure 2: Our Predictive EigenTracker for Hand Gestures: An Overview

2.2. Tracker Initialization

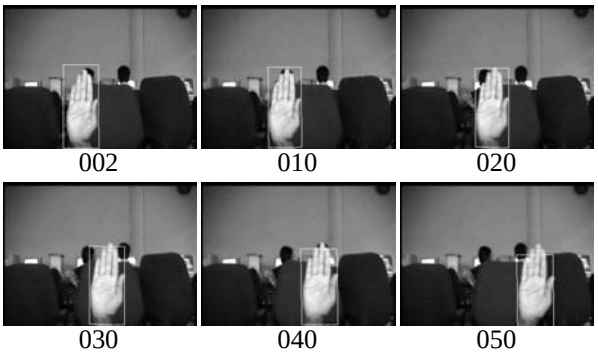
Initializing a tracker is a difficult problem because of multiple moving objects, and background clutter. In other words, one needs to segment out the moving region of interest from the possibly cluttered background in the frames. Our hand gesture tracker performs *fully automatic initialization.* We use a combination of motion cues (dominant motion detection [5] as well as skin colour cues [7], [9], [8] to identify the region of interest in each frame.

2.3. The Overall Tracking Scheme

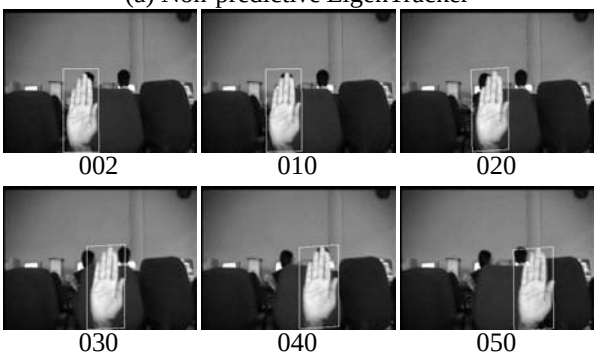
We now present an overview of our overall predictive EigenTracker for hand gestures (Figure 2 outlines the main steps). For the first few frames, we segment out the moving hand (Section 2.2). We now predict affine parameters – a parallelogram bounding box for the next frame (Step 1 in Figure 2, details in Section 2). 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 (Equation 1, Appendix), it decides on whether or not to perform an eigenspace update (Section 2.1). If the reconstruction error 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 2). This cue indicates an *epoch* change (Section 3). It then repeats the above steps for the next frame.

Figure 1 shows the result of an experiment on a typical hand gesture sequence. Our tracker can successfully track the moving hand in a variety of changing poses, in spite of background clutter, as well as 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



(a) Non-predictive EigenTracker



(b) Predictive EigenTracker

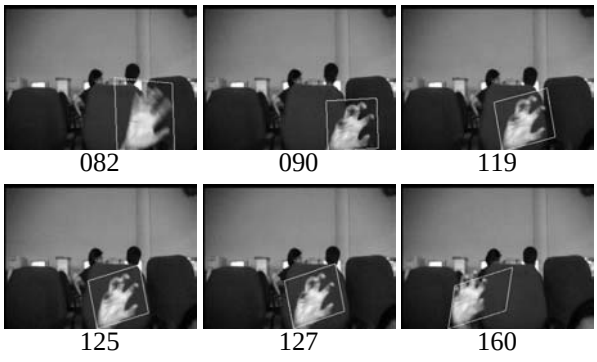
Figure 3: Tracking results with (a) a simple EigenTracker, as compared with (b) results of our predictive EigenTracker (bottom row): some representative frames. The hand is not properly tracked using the former.

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.

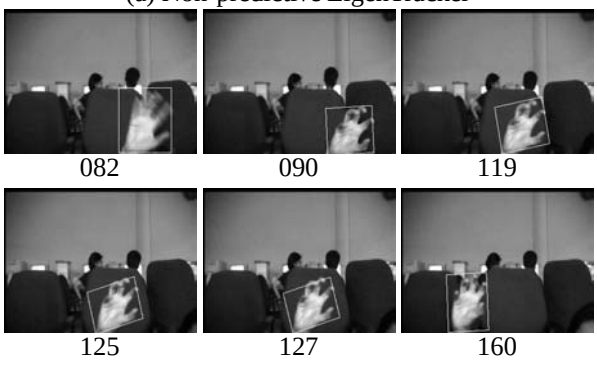
2.4. Synergistic Conjunction with Other Trackers: Restricted Affine Motion

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.

We have experimented with using an SVD update-based multi-resolution EigenTracker with a skin colour-based



(a) Non-predictive EigenTracker



(b) Predictive EigenTracker

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

CONDENSATION tracker [9], [8] for cases of restricted affine motions. 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 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,

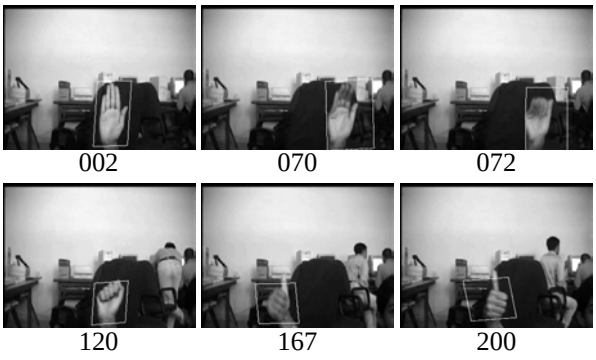


Figure 5: A simple combination of a CONDENSATION skin tracker with an online EigenTracker: scaling and translation. Details in Section 2.4

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.

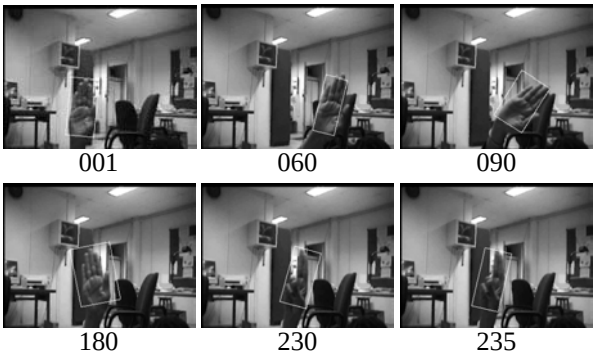


Figure 6: Using an online EigenTracker in conjunction with a skin colour-based CONDENSATION tracker: rotation, translation, scaling. Details in Section 2.4

3. Gesture Recognition

We propose a novel methodology for a gesture recognition system. We use our predictive EigenTracker (Section 2) to track hand motion across frames. Our predictive EigenTracking mechanism fits seamlessly into our gesture representation and recognition framework. We represent each gesture as a finite state machine (Figure 7). The states in the FSM correspond to different static hand shapes. In our system, a fixed stationary hand shape is taken as the start shape

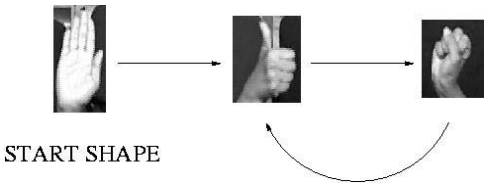


Figure 7: A very simple example of the representation of a gesture. We represent a gesture as composed of a particular temporal sequence of *epochs*, and transitions between them (Details in text).

– a stationary open palm. A stop state (signifying the end of the gesture) is a position of the hand that has not changed its position for at least a particular number of frames. A gesture is composed of a fixed temporal order of transitions between states.

The system stores an eigenspace representation corresponding to each static hand shape. It is important to note that the segmentation of all hand appearances during a gesture is done *automatically*, based on the eigenspace reconstruction coefficients. The tracker works on the basis of the eigenspace reconstruction error (Section 2). If the eigenspace reconstruction error is less than the parameter T_1 (as defined in Figure 2), then we take the hand shape to be the same as that corresponding to the previous frame. It means that the system is in the same state as it was for the previous frame. If the error lies between T_1 and T_2, we update the eigenspace representation corresponding to this hand shape. Only when the error exceeds T_2, does the EigenTracker signal an *epoch change*. This epoch change corresponds to a drastic change in hand shape, and hence, a new state of the FSM. The system searches hypothesized transitions from the current state, based on a predefined set of gestures. Such a state-based representation *imparts robustness to the speed at which a hand gesture is performed* – it will always correspond to the same set of states.

Our current set of gestures explores the idea of having a static hand shape represent a state, or an epoch. Since we have a predictive EigenTracker, we have information about temporal and spatial changes as well. Hence, an extension of our scheme will also include this information – for the case when the shape of the hand does not change significantly, but the position of the hand changes significantly with time.

The static hand-shapes corresponding to the individual states can vary from person to person (*e.g.*, a open hand shape can be different for different people). We propose a personalized gesture recognition system. Hand shapes of individuals with fixed semantics are learnt *a priori*. Our system uses these learnt shapes for identification of states. We exploit hand tracking, epoch changes and state identification for gesture recognition. A change in epochs (or the gesture itself) can switch the system to a different task. The system tries to recognize a particular hand shape when it detects an epoch change. For our system, we use a contour-based shape recognition strategy [11] for verifying particular hand shapes. Figure 8 depicts the system verifying two particular hand shapes: an open hand, and a closed hand.

Figure 8: Contour-based verification of static hand shapes (See text for details)

We present some preliminary results with our eigenspace-based gesture recognition system. In the sequence of Figure 1, the system starts with the eigenspace corresponding to an open hand. The eigenspace reconstruction error starts changing drastically at frame number 75 (corresponding to the upper threshold T_2), and doesn't change much thereafter. This represents a transition from the open hand to the closed hand. Figure 5 shows the result with another gesture, performed by another individual. The gesture starts from the same start shape, goes to the closed hand pose, and ends up at the thumbs-up sign. Here again, each epoch is triggered by a drastic change in the eigenspace reconstruction error. The system re-initializes itself with a fresh eigenspace corresponding to the closest static shape in its database during each such change.

3.1. A Simple Application: A 3-D Mouse

This section describes a simple application of some of the ideas presented in the preceding sections – a 3-D mouse. The motivation behind this is to have a hand (moving in 3-D space) substituting for a mouse (without extracting any 3-D positional information). The first image in Figure 9 shows an example of such a setup: a camera is looking down on a table, where the user moves his or her hand. The other frames of Figure 9 show screen snaps of the program in execution. The left window shows what the camera sees. On the right, we show the hand, segmented out from the image. For each such segmented out hand, we use the following heuristic to compute the position of the virtual mouse pointer. We consider the two eigenvalues corresponding to the hand shape, and find out the principal axis of the hand shape. We use this to compute the position of the extreme tip of the fingers. This is where we place the virtual mouse pointer. If the ratio of the eigenvalues is greater than a threshold, we consider it to be a pointing gesture, and interpret it to be a mouse click. This system is on-line, and implemented on a 700 MHz machine running Linux.

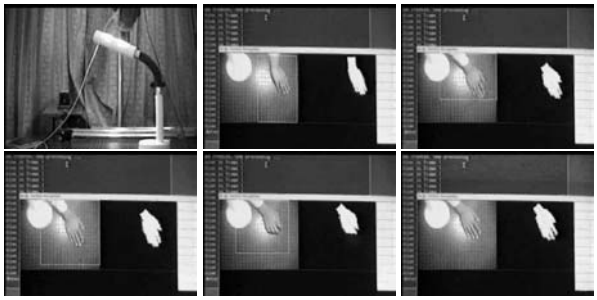


Figure 9: A simple application: a 3-D mouse. The first figure shows the setup with a camera looking at a table lit by a table lamp. The rest are some sample screen shots.

4. Conclusions

In this paper, we propose a novel gesture-based interface formulation. Our on-line predictive EigenTracker tracks the moving hand across frames in cluttered and non-stationary backgrounds. We also propose a representation scheme for gestures which fits in with the predictive EigenTracker. The paper shows the results of experiments with our system, in support of the proposed methodologies.

Appendix: CONDENSATION and EigenTracking

The CONDENSATION algorithm [6] 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 eigenspace approach involves using pixel data from images, rather than extracting features from them. Such an approach involves treating images (or sub-images) as vectors, and constructing the corresponding eigenspace. An advantage of this approach is the encoding of all available data about the appearance of an object (present in the images). An EigenTracker uses an eigenspace for tracking the movement of an object across frames, based on appearance information. An EigenTracking approach [2] 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 $\mathbf{x} = [x\ y]^T$:

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

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

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

References

[1] M. Agrawal, S. Sadhwani, S. Chaudhury, and S. Banerjee. Recognition of Dynamic Hand Gestures. In S. Chaudhury and S. K. Nayar, editors, *Computer Vision, Graphics and Image Processing: Recent Advances*, pages 179 – 184. Viva Books Private Limited, 1999.

[2] 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.

[3] A. F. Bobick and A. D. Wilson. A State-Based Approach to the Representation and Recognition of Gesture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(12):1235 – 1337, December 1997.

[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*, 59(5):321 – 332, September 1997.

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

[6] M. Isard and A. Blake. CONDENSATION - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 28(1):5 – 28, 1998.

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

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

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

[10] 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*, 19(7):677 – 695, July 1997.

[11] A. Ramamoorthy, N. Vaswani, S. Chaudhury, and S. Banerjee. Recognition of Dynamic Hand Gestures. *Pattern Recognition*, (Under review).

[12] M. Yeasin and S. Chaudhuri. Visual Understanding of Dynamic Hand Gestures. *Pattern Recognition*, 33:1805 – 1817, 2000.