Hand Gesture Modelling and Recognition involving Changing Shapes and Trajectories, using a Predictive EigenTracker

Kaustubh Srikrishna Patwardhan [a] Sumantra Dutta Roy [a,*]

[a] *Dept. of Electrical Engineering, IIT Bombay, Powai, Mumbai - 400 076, INDIA*

Abstract

We present a novel eigenspace-based framework to model a dynamic hand gesture that incorporates both hand shape as well as trajectory information. We address the problem of choosing a gesture set that models an upper bound on gesture recognition efficiency. We show encouraging experimental results on a such a representative set.

Key words: Dynamic Hand Gestures, Predictive EigenTracker, Eigenspace

1 Introduction

In gesture-based recognition systems, a common requirement is to develop a gesture set for a particular task. This paper proposes a novel eigenspace-based modelling of a gesture set, which takes into account both trajectory and shape information. The system uses a Predictive EigenTracker to efficiently track the changing appearance of a moving hand. This framework also allows us to choose a gesture vocabulary so as to maximise recognition accuracy.

A dynamic hand gesture comprises a sequence of hand shapes with associated spatial transformation parameters (translation, rotation, scaling/depth variations etc.) that describe the hand trajectory. Pavlovic et al. [1] give an extensive review of the existing hand gesture recognition techniques. While some systems use multiple cameras [2], it is more challenging to use a single uncalibrated camera. Many approaches focus primarily on motion/trajectory

* Author for correspondence
Email addresses: kaustubh@ee.iitb.ac.in (Kaustubh Srikrishna Patwardhan), sumantra@ee.iitb.ac.in (Sumantra Dutta Roy).

information (e.g., [3], [4]), or shape information (e.g., [5]), while some consider both. We review their characteristics, below. Commonly used shape descriptors include normalised Fourier descriptors [6], normalised hand lengths [7], Point Distribution Models (PDMs) [8], and geometric moments of hand pixels [9]. While some of these features are invariant to some distortions, any feature-based method involves a separate time-consuming and noise-prone feature detection step. In contrast, our method does not have any explicit feature detection step. We use appearance-based eigenspaces: using information from all image pixels in the region of interest. Further, our method is independent of common hand shape deformations: rotation, translation, scale and shear. (Our system works on top of our Predictive EigenTracker [10], an enhancement of the original EigenTracker [11]. This tracker gives us both appearance as well as motion/trajectory information, and is robust to background clutter and structured noise. Section 2 describes this in a nutshell.) For the temporal modelling and recognition, most systems use Finite State Machines (FSMs) (e.g., [3]), or the more general Hidden Markov Models (HMMs) (e.g., [12], [13], [4], [6]). HMMs have the disadvantage of a very elaborate training procedure to be effective. Our recognition scheme involves eigenspace projection, and finding computing the probability of a gesture based on the Mahalanobis distance. We explicity consider both shape and motion characteristics - we show results on gestures involving the same shapes but different motion characteristics, and vice versa. We explicitly consider the problem of selecting a gesture set for a particular application, such that the distance between gesture classes in the eigenspace, is maximised. This explicitly models the upper bond on the success rate of a particular set of gestures. To the best of our knowledge, no related work addresses this issue. An earlier preliminary version of this work appears in [14].

2 Predictive EigenTracking: Efficient Tracking of Objects Undergoing Changes in Appearance, and Position

Black and Jepson's EigenTracker [11] can track moving objects, which undergo changes in appearance as well. The authors learn (off-line) the eigenspace of appearances of the object to track, and pose the problem as estimating 2-D affine transformation coefficients $\mathbf{a}$ and the eigenspace reconstruction coefficients $\mathbf{c}$, to minimise a 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 eigenvectors):

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

ALGORITHM PREDICTIVE_EIGENTRACKER

1. INITIALISATION: Delineate object of interest
2. Consider sample set $\{\mathbf{s}_{t-1}^{(i)}, \pi_{t-1}^{(i)}\}, 1 \leq i \leq N$, starting at $t = 1$

REPEAT FOR ALL frames:

3. SELECT sample set for prediction using $\pi_{t-1}^{(i)}$
4. PREDICT new sample set: seed values for non-linear optimisation
5. Get MEASUREMENT $\pi_t^{(i)}$, optimising affine params **a** & recons coeffs **c**
6. Tracker OUTPUT: the sample $\mathbf{s}_t^{(j)} \equiv$ least recons error
7. IF recons error $\in (T_1, T_2]$ THEN update eigenspace

   ELSE IF recons error $> T_2$ THEN construct eigenspace afresh

Fig. 1. Our Predictive EigenTracker: An Overview (All details in Section 2)

Here, $\rho(x, \sigma) = x^2/(x^2 + \sigma^2)$ is the robust error function, and σ is a scale parameter [11]. The 2-D affine transformation is given by

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

A parallelogram offers tighter fit to the object being tracked as compared to a rectangular bounding box. This is an important consideration for an appearance-based method, since we do not want much background to be learnt as part of the eigenspace representation of the object. Our Predictive EigenTracker [10] augment the capability of an EigenTracker in three ways. *One of the main factors for the inefficiency of the EigenTracker is the absence of a predictive framework.* We propose a Particle filtering/CONDENSATION [15]-based predictive framework (this works with any distribution function, unlike the Gaussian assumptions in a Kalman Filter). In addition to speeding up the measurement process (search), it also gives a good seed value for the above non-linear optimisation. For tracking a moving hand, it is not feasible to learn the vast multitude of possible hand appearances off-line. The predictive EigenTracker learns and tracks unknown views of an object *on the fly* with an efficient on-line eigenspace update mechanism. Fig. 1 gives an overview of the Predictive EigenTracker. We use a six-element state vector $\mathbf{X}_t$ (the coordinates of any 3 non-collinear image points serve as a 2-D affine basis), with a second order AR model for state/process dynamics: $\mathbf{X}_t = \mathbf{D}_2\mathbf{X}_{t-2} + \mathbf{D}_1\mathbf{X}_{t-1} + \mathbf{w}_t$, where t represents time, $\mathbf{D}_i$ are 6×6 matrices, and $\mathbf{w}_t$ is a zero-mean, white, Gaussian random vector. For the specific case of hand tracking experiments in this paper, we have found a constant velocity model to suffice: $\mathbf{X}_t = 2\mathbf{X}_{t-1} - \mathbf{X}_{t-2} + \mathbf{w}_t$. (For our original Predictive EigenTracker [10] however, we have experimented with a large array of

state/process dynamics models: random walk, constant velocity, and models with coefficients learnt from representative samples: a commonly used procedure [15].) For the experiments in this paper, we have empirically determined the noise covariance, from a large number of representative gesture sequences.

As in any particle filter [15], we start with a set of N samples (our experiments use $N = 100$) $\{\mathbf{s}_{t-1}^{(i)}, \pi_{t-1}^{(i)}\}$, $1 \leq i \leq N$, where samples $\mathbf{s}_{t-1}^{(i)}$ are drawn from $P(\mathbf{X}_{t-1}|\mathbf{Z}_{1:t-1})$, the state distribution given all observations $\mathbf{Z}_1 \ldots \mathbf{Z}_{t-1}$ (6-element observation vectors) thus far. The observation probability set $\pi_{t-1}^{(i)} = P(\mathbf{Z}_{t-1}|\mathbf{X}_{t-1} = \mathbf{s}_{t-1}^{(i)})$ can be initialised to $1/N$ at the start of the algorithm [15]. For a hand tracking application, the Predictive EigenTracker uses a combination of skin colour and motion cues to perform *fully automatic initialisation* (Step 1 in Fig. 1). We initialise the sample set $\mathbf{s}_0^{(i)}$ with N samples from a Gaussian around this tracker initial state. The next step is to select a new set of samples according to the $\pi_{t-1}^{(i)}$ probability distribution, and use the State/Process Dynamics Model to predict the new set of N samples (Steps 3 and 4 in Fig. 1). *These serve as the seed values for the basic EigenTracker's non-linear optimisation (Eqn. 1).* The optimisation finds the affine coefficients $\mathbf{a}$ and eigenspace reconstruction coefficients $\mathbf{c}$ which correspond to the least reconstruction error for each sample. ($\mathbf{a}$ aligns the sample to the eigenspace, and $\mathbf{c}$ corresponds to the reconstruction in the aligned eigenspace, Step 5 in Fig. 1.) We formulate $P(\mathbf{Z}_t|\mathbf{X}_t = \mathbf{s}_t^{(i)})$ as being proportional to the negative exponential of the above reconstruction error. After renormalisation of the $\pi_t^{(i)}$ weights [15], we consider the sample with the least reconstruction error as the tracker output: $\mathbf{c}$ gives the appearance information, and $\mathbf{a}$ gives the deformation information (Step 6 in Fig. 1). We use two (empirically determined) thresholds T_1 and T_2. If the reconstruction error $\in (T_1, T_2]$, we update the eigenspace. If it is too large, this indicates a drastic appearance change: we construct the eigenspace afresh. We show numerous examples of successful tracking in [10] with cluttered backgrounds: colour MPEG video examples are available at `http://www.ee.iitb.ac.in/~sumantra/icip04a/`.

3 Gesture Modelling: Shape-Trajectory Eigenspace

A common requirement in many gesture-based systems is to formulate a set of gestures suitable for a particular task. In this paper, we propose a framework that accounts for both the shape as well as temporal trajectory of the moving hand, and helps in selecting a vocabulary to maximise recognition accuracy.

As mentioned in the previous section (Section 2), the output of our Predictive EigenTracker is a set of eigenspace reconstruction coefficients $\mathbf{c}$ and affine transformation coefficients $\mathbf{a}$. The $\mathbf{c}$ parameters represent the shape of the

moving hand, and the $\mathbf{a}$ parameters, information about the movement. An shape-based eigenspace involves the general overall appearance of the hand, and is thus robust to the existence of any particular feature. The Predictive EigenTracker updates the learnt eigenspace based on the reconstruction error (Step 5 in Fig. 1). *Hence, a drastic change in the appearance of the gesticulating hand, caused by the change in the hand shape, results in a large reconstruction error.* This forces an epoch change, indicating a new shape of the gesticulating hand. We use a subset $\mathbf{s}_{i_n}$ of the most representative eigenspace reconstruction coefficients $\mathbf{c}$, to represent the shape parameters corresponding to shape at epoch number i_n. We train the system to recognise different shapes by constructing an eigenspace of suitably scaled shapes, from a large number of training instances corresponding to the same shape.

After every epoch, the sequence of affine coefficients outputs $\mathbf{a}$ of the Predictive EigenTracker represents the trajectory traced by the hand shape $\mathbf{s}_{i_n}$ in space. We model the trajectory by a curve with parameters $\mathbf{t}_{i_n}$. (If we have a large set of such parameters, we can construct an eigenspace for these as well, and take the most representative ones.) A particular gesture $\mathcal{G}_i^k$ can have k shape-trajectory vector pairs. *We model a particular gesture $\mathcal{G}_i^k$ as an $m-$dimensional vector of a sequence of shape and trajectory coefficients, $\mathbf{g}_i^k = [\mathbf{s}_{i_1}\mathbf{t}_{i_1} \ldots \mathbf{s}_{i_k}\mathbf{t}_{i_k}]^T$. Thus, $\mathcal{G} = \bigcup_{i,k} \mathcal{G}_i^k$ represents the gesture vocabulary, or set.*

3.1 Choosing a Gesture Set

For each set of k shape-trajectory coefficient pair vectors $\mathbf{g}_i^k$, we compute the mean gesture vector $\overline{\mathbf{g}_i^k}$, and covariance matrix $\mathbf{\Sigma}_i^k$. Given a query gesture $\mathcal{G}_j^k$, we compute the Mahalanobis distance d_{ij}^k of its vector representation $\mathbf{g}_j^k$ from that of all gestures $\mathcal{G}_i^k$ with k shape-trajectory coefficient pairs:

$$d_{ij}^k = \left[(\mathbf{g}_j^k - \overline{\mathbf{g}_i^k})^T (\mathbf{\Sigma}_\mathbf{i}^\mathbf{k})^{-1} (\mathbf{g}_j^k - \overline{\mathbf{g}_i^k})\right]^{\frac{1}{2}}, \ \forall \mathcal{G}_i^k \in \mathcal{G} \tag{3}$$

This also gives us a probability of the given gesture $\mathcal{G}_j^k$ being one of the gestures in the given set $\mathcal{G}_i^k$: $p_{ij}^k = \frac{exp(-d_{ij}^k)}{\sum_i exp(d_{ij}^k)}$. In this scheme, the gesture-space consists of smaller subspaces, each corresponding to a particular *class* of gestures. *A region* in the gesture-subspace of the corresponding gesture-class represents a gesture, characterised by the mean gesture vector and the covariance matrix. *To formulate a gesture vocabulary for a high recognition accuracy, gestures should be so chosen that the gesture-classes are well-separated in gesture-space, and the intra-class distance is small.* Clearly, the maximum intra-class distance relative to the minimum inter-class distance puts an upper bound on the accuracy of the recognition system. Higher the separation of close-knit gesture-classes in gesture-space, better is the performance of the recognition system.

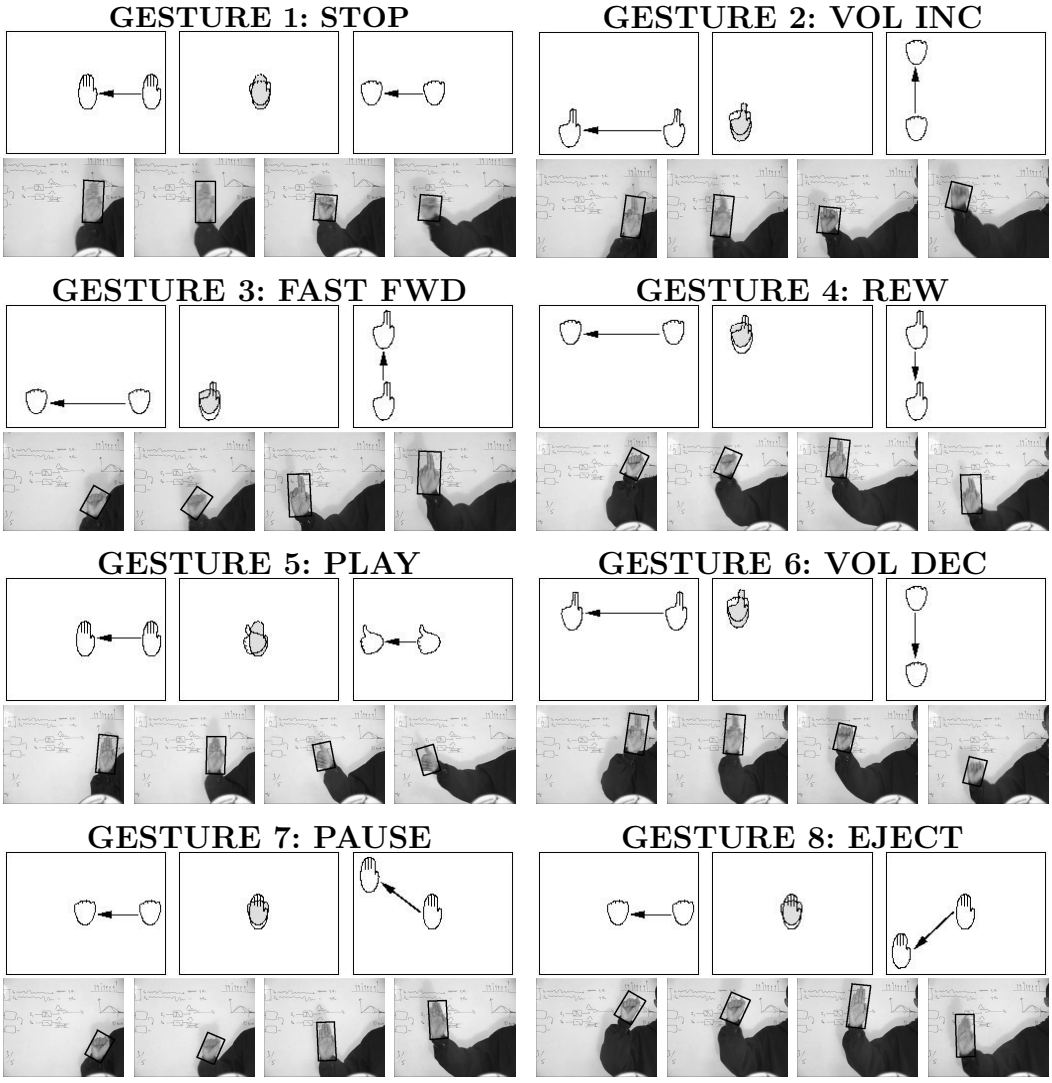

Fig. 2. A sample gesture set for controlling an audio player such as Winamp©: the gestures cannot be distinguished on the basis of shape or trajectory alone. The top row for each gives a schematic, the bottom one shows representative tracker output frames. Colour MPEG videos at `http://www.ee.iitb.ac.in/~sumantra/prl05a/` clearly show the non-uniform background and shadows.

4 Experiments with a Representative Gesture Set

We have chosen a representative gesture set (for controlling an audio player such as Winamp©), in accordance with the guidelines in the previous section (Section 3.1). We have chosen this set so that we cannot recognise a gesture on the basis of shape or trajectory alone. Fig. 2 shows the eight gestures in our set. The upper row in each gesture gives a schematic representation, and the lower row shows the output of our Predictive EigenTracker on a representative sequence. Colour MPEG videos of the same are available at `http://www.ee.iitb.ac.in/~sumantra/prl05a/`. We use four different

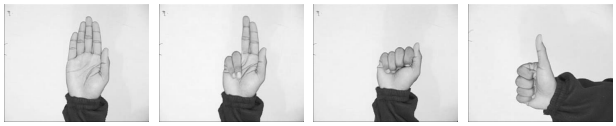

Fig. 3. Different hand shapes in the gesture-set. Not just are the gestures far apart in gesture-space, the shapes themselves significantly differ in appearance (Section 4).

	GES. 1	GES. 2	GES. 3	GES. 4	GES. 5	GES. 6	GES. 7	GES. 8
GES. 1	**0**	1.5×10^8	5.8×10^7	6.7×10^7	8.5×10^7	1.1×10^8	7.8×10^7	1.7×10^8
GES. 2	3.3×10^8	**0**	1.3×10^7	5.4×10^7	1.9×10^9	2.3×10^7	1.6×10^6	3.8×10^8
GES. 3	3.0×10^8	2.0×10^7	**0**	6.7×10^7	1.9×10^9	3.5×10^7	3.2×10^7	6.3×10^8
GES. 4	3.8×10^8	4.8×10^7	6.1×10^7	**0**	2.0×10^9	8.0×10^6	7.6×10^7	5.7×10^8
GES. 5	4.1×10^7	1.8×10^8	7.5×10^7	7.2×10^7	**0**	1.2×10^8	8.6×10^7	5.8×10^8
GES. 6	4.0×10^8	3.9×10^7	6.1×10^7	8.6×10^6	2.0×10^9	**0**	8.7×10^7	3.4×10^8
GES. 7	7.3×10^7	8.1×10^7	2.2×10^7	1.2×10^8	7.6×10^8	1.5×10^8	**0**	1.1×10^8
GES. 8	5.1×10^7	1.1×10^8	1.1×10^8	4.2×10^7	5.6×10^8	6.8×10^7	1.5×10^8	**0**

Table 1
Mahalanobis distance between the template gestures: far apart in gesture-space

hand shapes (Fig. 3) to construct the gesture vocabulary. *Not just are the gestures themselves far apart in gesture-space (Tables 1, 2, 3), we have also chosen the basic hand shapes to be significantly different in appearance - thus minimizing the possibility of incorrect shape identification.* Each gesture consists of two different hand shapes, requiring two epoch changes in the tracking phase. For the trajectories in our sample gesture set, we use a least-squares linear approximation. *It is important to note that gesture pairs {2, 6}, {3, 4}, and {7, 8} involve identical hand shapes (in order) and differ only in the hand trajectories. Conversely, in gesture pairs {1, 5}, {2, 3}, and {4, 6}, different hand shapes trace identical trajectories* (Fig. 2).

To calculate the mean gesture vector and corresponding covariance matrix, we used eight test sequences for every gesture in the vocabulary: 64 training sequences in all. Since every gesture involves two hand shapes, the training data has 128 hand shape images. For our set, 5 significant eigenvalues contributed more than 90% of the total energy. This can be explained by the redundancy in the hand shape images – 4 different hand shapes and 128 sample images. We therefore describe hand shapes by taking their projections on these 5 basis eigenvectors. Each gesture is thus represented by a 14 element vector, with 5 (shape) + 2 (trajectory) parameters corresponding to each epoch.

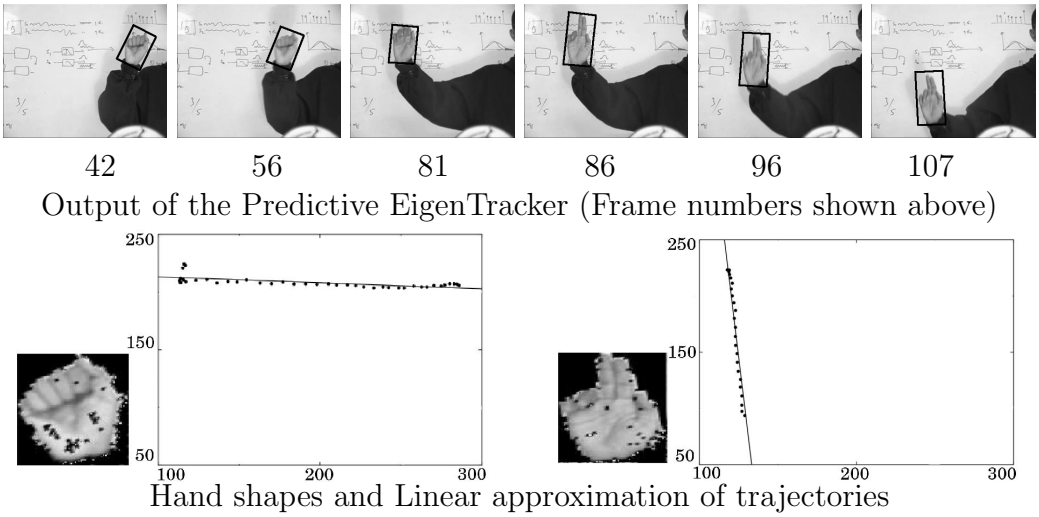

Fig. 4. Example: Gesture 4, Set 7 (Section 4.1). (Black dots correspond to faulty skin colour detection.) Video: `http://www.ee.iitb.ac.in/~sumantra/prl05a/`

4.1 Gesture Recognition

Fig. 4 shows the intermediate steps in processing of Gesture 4 from Set 7 using our scheme. The tracker follows the hand with initial bounding box parameters and eigenspace till frame 85. Here, a large reconstruction error forces an epoch change. Tracking commences with this new hand shape detected in frame 86. In the second row of images in Fig. 4, the left part shows the hand shape (scaled) detected by the tracker in frame 42, and the corresponding linear trajectory approximation. On similar lines, the right part shows the corresponding items for the second epoch of the gesture. We have tested the gesture recognition performance of this framework using 64 gestures present in the training set, and 16 additional gestures which were not used during the training phase. Table 2 lists the Mahalanobis distances of gestures of Set 7 from the template gestures. These gestures were used, among others, during the training phase to calculate the gesture templates. The system correctly recognised all these gestures. Similar results were also observed for other gestures that were used during training. Table 3 lists the Mahalanobis distances of Set 9 gestures (not used for training) from the template gestures. To conclude, the system recognised all 80 gestures with 100% accuracy.

5 Conclusions and Scope for Future Work

We present a novel approach to represent gestures that cannot be recognised by shape or trajectory information alone. We also address the problem of choosing a gesture set that models the upper bound on gesture recognition.

	GES. 1	GES. 2	GES. 3	GES. 4	GES. 5	GES. 6	GES. 7	GES. 8
1	**0.79**	1.5×10^8	1.2×10^8	7.9×10^7	3.8×10^7	1.2×10^8	8.6×10^7	1.5×10^8
2	2.5×10^8	**5.3**	2.8×10^7	2.4×10^7	1.8×10^9	1.8×10^7	5.8×10^6	2.0×10^8
3	2.6×10^8	2.0×10^7	**1.3**	3.8×10^7	2.0×10^9	3.7×10^7	2.9×10^6	6.2×10^7
4	3.8×10^8	8.3×10^7	6.2×10^7	**1.4**	2.0×10^9	6.2×10^6	6.3×10^7	3.1×10^8
5	6.6×10^7	1.6×10^8	3.8×10^7	5.7×10^7	**0.97**	1.1×10^8	8.2×10^7	1.2×10^8
6	4.4×10^8	7.3×10^7	4.7×10^7	1.2×10^7	2.1×10^9	**0.86**	4.6×10^7	3.1×10^8
7	6.6×10^7	1.3×10^8	3.7×10^7	8.8×10^7	6.4×10^8	1.1×10^8	**0.51**	2.7×10^8
8	4.0×10^7	1.1×10^8	1.1×10^8	3.4×10^7	6.7×10^8	5.6×10^7	1.6×10^8	**1.9**

Table 2
Mahalanobis distance of gestures of Set 7 from templates: they are widely apart.

	GES. 1	GES. 2	GES. 3	GES. 4	GES. 5	GES. 6	GES. 7	GES. 8
1	**0.98**	1.6×10^8	5.9×10^7	2.7×10^7	9.4×10^7	1.0×10^8	8.6×10^7	1.6×10^8
2	2.1×10^8	**0.26**	2.1×10^7	4.4×10^7	1.6×10^9	2.9×10^7	1.5×10^7	2.1×10^8
3	3.0×10^8	5.1×10^7	**0.88**	4.6×10^7	1.8×10^9	3.8×10^7	5.4×10^7	1.7×10^8
4	3.2×10^8	6.5×10^7	6.0×10^7	**0.77**	1.8×10^9	5.4×10^6	6.5×10^7	1.6×10^8
5	5.9×10^7	1.8×10^8	7.4×10^7	6.9×10^7	**0.95**	1.1×10^8	9.5×10^7	1.3×10^8
6	4.2×10^8	4.4×10^7	4.7×10^7	4.7×10^6	2.2×10^9	**0.99**	7.1×10^7	2.9×10^8
7	3.6×10^7	8.9×10^7	4.5×10^7	1.0×10^8	6.8×10^8	1.3×10^8	**0.88**	2.6×10^8
8	7.7×10^7	9.2×10^7	1.3×10^8	4.2×10^7	6.2×10^8	6.5×10^7	1.2×10^8	**0.97**

Table 3
Mahalanobis distance of Set 9 gestures (not used for training) from template gestures. The inter-gesture distances are orders of magnitude apart, as before.

Further extensions of this work include applying this framework to two-handed gestures, possibly using our robust two-hand tracker [16]. (This models all possible cases of hand-hand interactions.) Another interesting problem would be to best adapt the framework to a given set of gestures.

References

[1] V. I. Pavlovic, R. Sharma, T. S. Huang, Visual Interpretation of Hand Gestures for Human-Computer Interaction, IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (7) (1997) 677 – 695.

[2] H. Hongo, M. Ohya, M. Yasumoto, K. Yamamoto, Visual Recognition

of Static/Dynamic Gesture: Face and Hand Gesture Recognition for Human-Computer Interaction, in: Proc. International Conference on Pattern Recognition (ICPR), 2000, pp. 2921 – 2924.

[3] M. Yeasin, S. Chaudhuri, Visual Understanding of Dynamic Hand Gestures, Pattern Recognition 33 (2000) 1805 – 1817.

[4] B. Min, H. Yoon, J. Soh, Y. Yang, T. Ejima, Visual Recognition of Static/Dynamic Gesture: Gesture Driven Editing System, Journal of Visual Languages and Computing 10 (1999) 291 – 309.

[5] J. Triesch, C. Malsburg, Classification of Hand Postures against Complex Backgrounds using Elastic Graph Matching, Image and Vision Computing 20 (2002) 937 – 943.

[6] C. W. Ng, S. Ranganath, Real-Time Gesture Recognition System and Application, Image and Vision Computing 20 (2002) 993 – 1007.

[7] O. Al-Jarrah, A. Halawani, Recognition of Gestures in Arabic Sign Language using Neuro-Fuzzy Systems, Artificial Intelligence 133 (2001) 117 – 138.

[8] T. Ahmad, C. Taylor, A. Lanitis, T. Cootes, Tracking and Recognition of Hand Gestures using Statistical Shape Models, Image and Vision Computing 15 (1997) 345 – 352.

[9] Y. Zhua, G. Xu, D. Kriegman, A Real-Time Approach to the Spotting, Representation, and Recognition of Hand Gestures for Human-Computer Interaction, Computer Vision and Image Understanding 85 (2002) 189 – 208.

[10] N. Gupta, P. Mittal, K. S. Patwardhan, S. Dutta Roy, S. Chaudhury, S. Banerjee, Online Predictive Appearance-based Tracking, in: Proc. IEEE International Conference on Image Processing (ICIP), 2004, pp. 1041 – 1044.

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

[12] Y. Nam, K. Wohn, Recognition of Hand Gestures, with 3D, Non-linear Arm movement, Pattern Recognition Letters 18 (1997) 105 – 113.

[13] T. Kapuscinski, M. Wysocki, Hand Gesture Recognition for Man-Machine Interface, in: Proc. Second International Workshop on Robot Motion and Control, 2001, pp. 91 – 96.

[14] K. S. Patwardhan, S. Dutta Roy, Dynamic Hand Gesture Recognition using Predictive EigenTracker, in: Proc. Indian Conference on Computer Vision, Graphics and Image Processing, 2004, pp. 675 – 680.

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

[16] K. A. Barhate, K. S. Patwardhan, S. Dutta Roy, S. Chaudhuri, S. Chaudhury, Robust Shape Based Two Hand Tracker, in: Proc. IEEE International Conference on Image Processing (ICIP), 2004, pp. 1017 – 1020.