Image and Vision Computing 26 (2008) 1607–1620

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Hand gesture recognition and tracking based on distributed locally linear embedding

S.S. Ge *, Y. Yang, T.H. Lee

Social Robotics Lab, Interactive Digital Media Institute & Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576, Singapore

ARTICLE INFO

Article history:
Received 8 December 2005
Received in revised form 28 January 2008
Accepted 4 March 2008

Keywords:
Gesture recognition and tracking
Distributed locally linear embedding (DLLE)
Probabilistic neural network (PNN)
Hand recognition
Gesture tracking

ABSTRACT

In this paper, we present a computer vision system for human gesture recognition and tracking based on a new nonlinear dimensionality reduction method. Due to the variation of posture appearance, the recognition and tracking of human hand gestures from one single camera remain a difficult problem. We present an unsupervised learning algorithm, distributed locally linear embedding (DLLE), to discover the intrinsic structure of the data, such as neighborhood relationships information. After the embedding of input images are represented in a lower dimensional space, probabilistic neural network (PNN) is employed and a database is set up for static gesture classification. For dynamic gesture tracking, the similarity among the images sequence are utilized. Hand gesture motion can be tracked and dynamically reconstructed according to the image's relative position in the corresponding motion database. The method is robust against the input sequence frames and bad image qualities. Experimental results show that our approach is able to successfully separate different hand postures and track the dynamic gesture.
© 2008 Elsevier B.V. All rights reserved.

1. Introduction

Analysis and modeling of human hands posture have attracted much attention of many researchers for developing intelligent human computer interaction systems in recent years. It is an important research issue with many practical applications in the fields of virtual reality, sign language recognition and computer animation. Due to the variation of posture appearance, the recognition and tracking of human hand gestures remains a difficult problem. It is hard to extract information from gesture images for 3D hand reconstruction. In most cases, it is required to model the hand gestures and adjust the parameters of hand models until they match the observations. These parameters should provide the desired information from the captured images.

Recently, there are two main streams of hand gesture analysis. Glove-based methods utilize the mechanical marked gloves to directly capture hand motion and measure hand joints parameters by a set of sensors [1–3]. They can achieve real time speed and can be used reliably for animation. However, these methods are expensive and impractical because of the required equipment. The other solution is to use vision-based approaches which depend on image capture equipments [4–7]. It is the most natural way and yet the most difficult way to implement a satisfactory human computer interface.

Vision-based hand gesture analysis has greatly increased so far and several different methods have been proposed for hand model reconstruction [4,8]. These methods can be subdivided into two main categories: model-based approach and appearance-based approach. Conventional model-based approach for image analysis attempts to infer the pose of the palm and finger joints. This method searches for the kinematic parameters which map the 2D projection images to the 3D hand model. 3D hand tracking and gesture estimation are main issues of this subject. A visual tracking system DigitEyes is introduced which can recover a high articulated 3D hand model from grey scale images [6]. The system tracks the local edges of each finger link and uses the bisector of these lines as the link features for the hand model parameters. To improve the features extracted from images, many researchers paste colored markers on the hand [9,10]. Eight points are used to estimate the hand posture from one single 2D image in [10]. They reduced the hand model from 27 degree-of-freedom (DOF) to 12 DOF by analyzing the hand constraints. Color markers are used to detect the main points of the hand and hand postures are estimated without computing the inverse kinematics.

The appearance-based approach models the hand by a collection of 2D image templates. The gestures are modeled as a sequence of views by associating the images of hand gestures to the appearance of predefined ones [5,11]. The majority of appearance-based methods depend on the parameters extracted from images. This class of parameters is derived from the hand image properties including contours and edges, image moments and image eigenvectors. Eigenspace is currently the most commonly used method for vision-based approaches [12,13]. For vision-based hand gesture recognition and reconstruction, one important issue is to process the images to obtain the raw information that match the

* Corresponding author. Tel.: +65 65166821; fax: +65 67791103.
E-mail address: samge@nus.edu.sg (S.S. Ge).

0262-8856/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2008.03.004

output model. However, there is usually a huge amount of information in the captured images. Dimension reduction is critical for analyzing these images, which can compress image information and discover compact representations of variability.

Typical nonlinear dimension reduction techniques include Isomap and locally linearly embeddings (LLE). In Isomap, the geodesic relationships among the input data and the calculated low dimension embeddings remain constant [14]. In LLE, the local intrinsic structures are maintained in dimensionality reduction [15]. A novel methodology called neighborhood linear embedding (NLE) has been developed to successfully discover the intrinsic property of the input data which does not need the trial and error process inherent in LLE [16]. The compact information provides intermediate results that can be used for various onwards applications such as: machine vision, sensor fusion, documents processing and so on.

In this paper, we modify the LLE algorithm and present a new distributed locally linear embedding (DLLE) to discover the inherent properties of the input data, by noticing that some relevant pieces of information may be distributed. By estimating the probability density function of the input data, an exponential neighbor finding method is proposed. The input data are mapped to low-dimensional space where not only the local neighborhood relationship but also some global distributions are preserved. Because the DLLE can preserve the neighborhood relationships among samples, after embedded in low-dimensional space, a probabilistic neural network (PNN) can be employed for classification. PNN is a feed-forward network and supervised training algorithm [17]. It is based on Bayes' theorem and high recognition accuracy can be achieved with a large number of sample patterns.

The main contributions of this paper are as follows:

- An unsupervised learning method DLLE is introduced to recover the inherent properties of scattered data lying on a manifold embedded in high-dimensional input data.
- A method is presented to use the similarity of the gesture images for static hand gesture recognition. By applying DLLE–PNN, similar static gesture will be grouped together and different gestures can be easily classified.
- A method is introduced to utilize the neighborhood relationship of the gesture images for hand gesture motion tracking. At the run time, the system dynamically obtains the relative position of the captured gesture sequence image in the corresponding motion database and the hand model is then reconstructed in a virtual world.

The remainder of this paper is organized as follows. In Section 2, we illustrate the geometry of relationship between a real world human hand and a virtual hand model through one single web-camera. In Section 3, we briefly describe the neighborhood selection methods of LLE and NLE. In Section 4, the unsupervised learning algorithm, DLLE, is presented to discover the intrinsic structure of the input data by preserving neighborhood relationship. In Section 5, a PNN model is discussed for the classification of different static gestures. In Section 6, we illustrate how to model the virtual hand and how to add constraints to reduce the hand DOF. Section 7 describes the details of hand gesture recognition and tracking implementation. In Section 8, we present the experiment results of our hand gesture recognition and tracking system.

2. Projection relations

Consider the points and coordinate frames as shown in Fig. 1. The 3D point, $P_w = [x_w, y_w, z_w]^T$, in the world coordinate frame, Frame w, can be mapped to a 3D point, $P_i = [x_i, y_i, z_i]^T$, in the image frame, Frame i, by two frame transforms. By considering the pixel

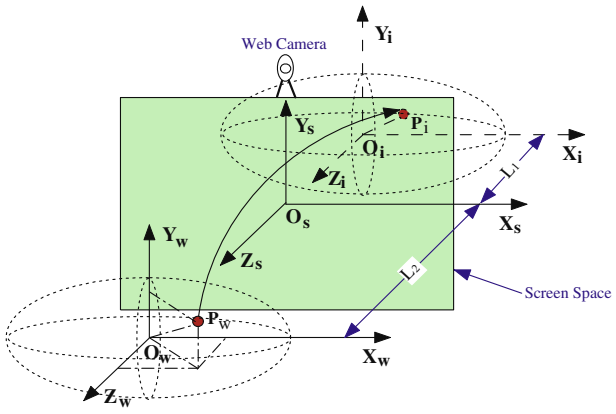

Fig. 1. Projection relations between the real world and the virtual world.

size and the image center parameter and using perspective projection with pinhole camera geometry, the transformation from P_w to point $P_s = [x_s, y_s, 0]^T$ in the screen frame, Frame s, is given by [18].

$$x_s = \frac{f}{s_x}\frac{x_w}{z_w} + o_x$$
$$y_s = \frac{f}{s_y}\frac{y_w}{z_w} + o_y \tag{1}$$

where s_x, s_y are the width and length of a pixel on the screen (o_x, o_y), is the origin of Frame s, and f is the focal length.

The corresponding image point P_i can be expressed by a rigid body transformation:

$$P_i = R_s^i P_s + P_{sorg}^i \tag{2}$$

where $R_s^i \in \mathbb{R}^{3\times3}$ is the rotational matrix, $P_{sorg}^i \in \mathbb{R}^3$ is the origin of Frame s with respect to Frame i.

Fig. 2 illustrates the projection relationship of a real gesture, a gesture image and the corresponding reconstruction model. Any hand movements will result in different images. The similarity between two images can be extracted by comparing the pixel values. These images can be thought of points in a high-dimensional space, with each input dimension corresponding to the brightness of 1 pixel in the image. Although the input dimensionality may be quite high (e.g., 19,200 pixels for a 160×120 image), the perceptually meaningful structure of these images has fewer independent degrees of freedom. In the following sections, we will describe how to discover compact representations of high-dimensional data.

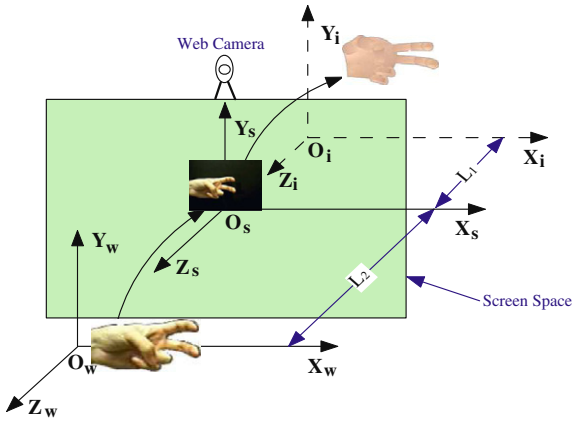

Fig. 2. Projection relationship of a real gesture, a gesture image and the corresponding reconstruction model.

3. LLE and NLE

For ease of the forthcoming discussion, we first introduce the main ideas of LLE and NLE methods. LLE is an unsupervised learning algorithm that attempts to map high-dimensional data to low-dimensional space while preserving the neighborhood relationship. Compared with principle component analysis (PCA) and multidimensional scaling (MDS), LLE is a nonlinear dimensionality reduction method. It is based on simple geometric intuitions: (i) each high-dimensional data point and its neighbors lie on or close to a locally linear patch of a manifold and (ii) the local geometric characterization in original data space is unchanged in the output data space. The neighbor finding process of each data point of LLE is: for each data point in the given data set, using the group technique such as K nearest neighbors based on the Euclidean distance, the neighborhood for any given point can be found. A weighted graph is set up with K nodes, one for each neighbor point, and a set of edges connecting neighbor points. These neighbors are then used to reconstruct the given point by linear coefficients.

In order to provide a better basis for structure discovery, NLE was proposed [16]. It is an adaptive scheme that selects neighbors according to the inherent properties of the input data substructures. The details on NLE algorithm can be found in [16].

Both LLE and NLE methods can find the inherent embedding in low dimension. According to the LLE algorithm, each point x_i is only reconstructed from its K-nearest neighbors by linear coefficients. However, due to the complexity, nonlinearity and variety of high-dimensional input data, it is difficult to find an appropriate value of K for all the input data to find the intrinsic structure [19]. The proper choice of K affects an acceptable level of redundancy and overlapping. If K is too small or too large, the K-nearest neighborhood method cannot properly approximate the embedding of the manifold. The size of range is closely related to the intrinsic property of the data, such as sample density. An improvement can be done by adaptively selecting neighbors according to the density of the sample points.

Another problem of using K-nearest neighbors in LLE is information redundancy. As illustrated in Fig. 3, e.g., for a certain manifold, we choose $K(K = 8)$ nearest neighbors to reconstruct x_i. However, the selected neighbors in the dashed circle are closely gathered. Obviously, if we use all samples in the circle as the neighbors of x_i, the information captured in that direction will have redundancy. A more straightforward way is to use one or several samples to represent a group of closely related data points.

According to NLE's neighborhood selection criterion, the number of neighbor selected to be used is small. For example, according to our experiment on two peaks data sample, the average number of neighbors for each point in 1000 samples are 3.74. The reconstruction information may not be enough for an embedding.

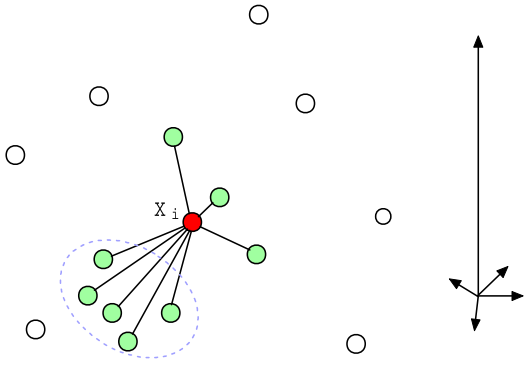

Fig. 3. Select $K(K = 8)$ nearest neighbors using LLE. The samples in the dashed circle along certain direction may cause the information redundancy problem.

By carefully considering the LLE and NLE's neighbor selection criterion, we propose a new algorithm by estimating the probability density function from the input data and using an exponential neighbor finding method to automatically obtain the embedding.

4. Distributed locally linear embedding (DLLE)

4.1. Estimation of probability density function

In most cases, a priori knowledge of the distribution of the samples in high dimension space is not available. However, we can estimate its density function for a given data set. Consider a data set with N elements in m dimensional space, for each sample x_i, the approximated distribution density function $\hat{p}_{x_i}$ around point x_i can be calculated as

$$\hat{p}_{x_i} = \frac{k_i}{\sum_{i=1}^{N} k_i} \tag{3}$$

where k_i is number of the points within a hypersphere kernel of fixed radius around point x_i.

Let $\hat{P} = \{\hat{p}_{x_1}, \hat{p}_{x_2}, \ldots, \hat{p}_{x_N}\}$ denote the set of estimated distribution density function, $\hat{p}_{\max} = \max(\hat{P})$ and $\hat{p}_{\min} = \min(\hat{P})$. The higher the value of distribution density function, the more samples are gathered; the lower the distribution density function, the fewer samples are gathered. These estimated distribution density functions are then used as a criterion for choosing neighbors.

4.2. Compute the neighbors of each data point

Suppose that a data set $\mathscr{X} = \{x_1, x_2, \ldots, x_n\}$, $x_i \in \mathbb{R}^m$ is globally mapped to a data set $\mathscr{Y} = \{y_1, y_2, \ldots, y_n\}$, $y_i \in \mathbb{R}^l$, $m \gg l$. For the given data set, each data point and its neighbors lie on or close to a locally linear patch of the manifold.

Assumption 1. The input data set $\mathscr{X}$ contains sufficient data in $\mathbb{R}^m$ sampled from a smooth parameter space Φ. Each data point x_i and its neighbors, e.g., x_j, lie on or close to a linear patch on the manifold. The range of this linear patch is subject to the estimated sampling density $\hat{p}$.

Based on above geometry conditions, the local geometry in the neighborhood of each data point can be reconstructed from its neighbors by linear coefficients. At the same time, the reconstruction information of two points depends on the mutual distance between them. The larger the distance between points, the less the mutual reconstruction information between them.

Assumption 2. The parameter space Φ is a convex subset of $\mathbb{R}^m$. If x_i and x_j is a pair of points in $\mathbb{R}^m$, ϕ_i and ϕ_j are the corresponding points in Φ, then all the points defined by $\{(1-t)\phi_i + t\phi_j : t \in (0,1)\}$ lies in Φ.

In view of the above observations, the following procedure is conducted by utilizing the neighborhood information to construct the neighborhood set of x_i, $\mathbf{S}_i (i = 1, \ldots, N)$. To better sample the neighbor data points, we propose an algorithm that exponentially extend the range to find the reconstruction sample as shown in Fig. 4.

For a given point x_i, we can compute the distances from all other points around it. According to the distribution density function around x_i estimated before, we introduce α_i to describe the normalized density of the sample point x_i. It is used to control the increment of the segment according to the sample points density for neighbor selection. We first give the definition of α_i by normalizing $\hat{p}_{x_i}$ using the estimated distribution density function computed by Eq. (3):

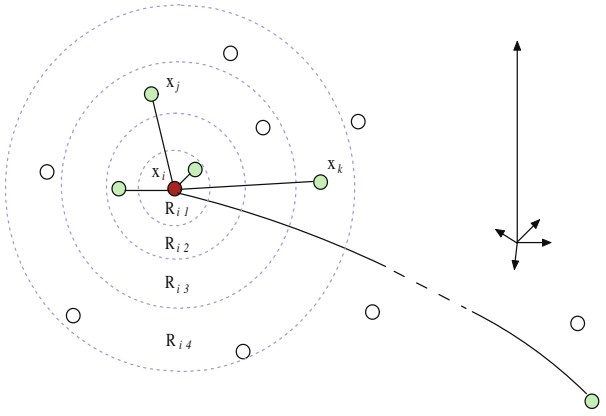

Fig. 4. The neighborhood selection process.

$$\alpha_i = \beta \cdot \frac{\hat{p}_{\max} - \hat{p}_{x_i}}{\hat{p}_{\max} - \hat{p}_{\min}} + \alpha_0 \tag{4}$$

where β is the scaling constant, default value is set to 1.0; and α_0 is the constant to be set. The discussion of this definition is given later.

According to the distances values from all other points to x_i, these points are stored in $\mathscr{R}_i$ by ascending order. Based on the estimated distribution density function, $\mathscr{R}_i$ is separated into several segments, where $\mathscr{R}_i = R_{i1} \cup R_{i2} \cup R_{i3} \cdots \cup R_{ik} \cdots \cup R_{iK}$. The range of each segment is given following an exponential format:

$$\begin{cases} \min(R_{ik}) = \lceil \alpha_i^k \rceil \\ \max(R_{ik}) = \lceil \alpha_i^{k+1} \rceil \end{cases} \tag{5}$$

where k is the index of segment and $\lceil \alpha_i^k \rceil$ denotes the least upper bound integer when α_i^k is not an integer. We can see that the value of α_i must be greater than 1.0. It is obvious that when α_i is less than 1.0, α_i^k will decrease with respect to the increase of k. Therefore, the data samples cannot be segmented. At the same time, if α_i is too large, α_i^k will increase fast and there may not be enough samples captured, and therefore, local information is hard to keep. A suitable range of α_i can be set from 1.0 to 2.0 by setting $\alpha_0 = 1.0$. While α_i is shifting from small to large, the range of the neighbors selected is from small to large. When α_i is close to 1.0, the neighbor selected are just the several nearest neighbors and the performance will be similar to LLE. When α_i is close to 2.0, more global information will be obtained.

Eq. (4) can be illustrated as follows: when some samples are closely gathered, the probability density around these samples is large; therefore, more information should be captured locally and α_i should be small. When the probability density is small, a larger α_i should be adopted.

Usually the samples in the manifold, N, is large. Thus, we can modify Eq. (5) to accommodate LLE when α_i is close to 1.0 as follows:

$$\begin{cases} \min(R_{ik}) = \lceil \alpha_i^k \rceil + k - 1 \\ \max(R_{ik}) = \lceil \alpha_i^{k+1} \rceil + k \end{cases} \tag{6}$$

Let us define a sequence of integers to represent the segment boundaries point index $r_{ik} = \max(R_{ik}) = \lceil \alpha_i^{k+1} \rceil + k$. For each segment R_{ik}, the mean distance from all points in this segment to x_i can be calculated by

$$\bar{d}_{ik} = \frac{1}{r_{i(k+1)} - r_{ik}} \sum_{j=r_{ik}}^{r_{i(k+1)}-1} \|x_i - x_{u_i(j)}\|^2 \tag{7}$$

where $u_i(j)$ represents the jth point after sorting in ascending order. To overcome the information redundancy problem, using the mean distance computed by Eq. (7), we find a most suitable point in R_{ik} to represent the contribution of all points in R_{ik} by minimizing the following cost equation:

$$\varepsilon(d) = \min \|\bar{d}_k - x_j\|^2, \quad \forall j \in R_{ik} \tag{8}$$

If point x_j satisfies Eq. (8), then x_j is selected to represent the data points of this segment. Then, we can update the neighborhood set of x_i, $\mathbf{S}_i$

$$\mathbf{S}_i = \mathbf{S}_i \cup \{x_j\} \tag{9}$$

For each segment, we update $\mathbf{S}_i$ once. Finally, when the neighbor set $\mathbf{S}_i$ is computed out, the number of members in $\mathbf{S}_i$ is recorded as n_i.

There are two variations for the choice of neighbor number for different purposes:

- To determine the number of neighbors to be used for further reconstruction and achieve adaptive neighbor selection, we can compute the mean distance from all other samples to x_i

$$\bar{d}_i = \frac{1}{N} \sum_{j=1}^{N} \|x_i - x_j\|^2, \quad i \neq j \tag{10}$$

  Starting with the $\mathbf{S}_i$ computed above at a given point x_i, from the largest element in $\mathbf{S}_i$, delete element one by one until all elements in $\mathbf{S}_i$ is less than the mean distance $\bar{d}_i$ computed by Eq. (10). Then the neighbor set $\mathbf{S}_i$ for point x_i is fixed.
- For manual neighbor selection, by adjusting the index k, we can choose the radius of the neighbor range. A small k means fewer samples are adopted while a larger k will result in more samples included. The number of neighbors k should always be greater than the target dimensionality l.

4.3. Calculate the reconstruction weights

The reconstruction weight W is used to rebuild the given points. To store the neighborhood relationship and reciprocal contributions to each other, the sets $\mathbf{S}_i$ $(i = 1, 2, \ldots, N)$ is converted to a weight matrix $W = \{w_{ij}\}(i, j = 1, 2, \ldots, N)$. W is constructed by setting its columns corresponding to each data vector in the original space and its number of rows corresponding to the number of selected neighbors. The construction weight W that best represents the given point x_i from its neighbor x_j is computed by minimizing the cost function given below:

$$\varepsilon(W) = \sum_{i}^{N} \left\| x_i - \sum_{j=\mathbf{S}_{i(1)}}^{\mathbf{S}_{i(n_i)}} w_{ij} x_j \right\|^2, \quad i \neq j \tag{11}$$

where the weight w_{ij} represents the contribution of the jth data point to the ith point's reconstruction. The reconstruction weight w_{ij} is subjected to two constraints.

(1) First, each data point x_i is reconstructed only from its neighborhood set points, enforcing $w_{ij} = 0$, if x_j is not its neighbor.
(2) Second, the rows of the weight matrix sum to one: $\sum_{j=\mathbf{S}_{i(1)}}^{\mathbf{S}_{i(n_i)}} w_{ij} = 1$.

To compute W row by row, Eq. (11) can be further written as

$$\begin{aligned} \varepsilon(W_i) &= \|x_i - \sum_{j=\mathbf{S}_{i(1)}}^{\mathbf{S}_{i(n_i)}} w_{ij} x_j\|^2, \quad i \neq j \\ &= \left\| \sum_{j=\mathbf{S}_{i(1)}}^{\mathbf{S}_{i(n_i)}} w_{ij} x_i - \sum_{j=\mathbf{S}_{i(1)}}^{\mathbf{S}_{i(n_i)}} w_{ij} x_j \right\|^2 \\ &= \left\| \sum_{j=\mathbf{S}_{i(1)}}^{\mathbf{S}_{i(n_i)}} w_{ij} (x_i - x_j) \right\|^2 \\ &= \sum_{j=\mathbf{S}_{i(1)}}^{\mathbf{S}_{i(n_i)}} w_{ij} \sum_{k=\mathbf{S}_{i(1)}}^{\mathbf{S}_{i(n_i)}} w_{ik} (x_i - x_j)^{\mathrm{T}} (x_i - x_k) \end{aligned} \tag{12}$$

where W_i is the ith row of W. By defining a local covariance C_i

$$C_i(j,k) = (x_i - x_j)^{\mathrm{T}}(x_i - x_k)$$

combined with the Constraint (2) of W, we can apply the Lagrange multiplier and have

$$\varepsilon(W_i) = \sum_{j=\mathbf{S}_{i(1)}}^{\mathbf{S}_{i(n_i)}} w_{ij} \sum_{k=\mathbf{S}_{i(1)}}^{\mathbf{S}_{i(n_i)}} w_{ik} C_i(j,k) + \lambda_i \left(\sum_{j=\mathbf{S}_{i(1)}}^{\mathbf{S}_{i(n_i)}} w_{ij} - 1 \right) \tag{13}$$

where λ_i is the Lagrange coefficient. To obtain the minimum of ε, we can find the partial differentiation with respect to each weight and set it to zero

$$\frac{\partial \varepsilon(W_i)}{\partial w_{ij}} = 2 \sum_{k=\mathbf{S}_{i(1)}}^{\mathbf{S}_{i(n_i)}} w_{ik} C_i(\mathbf{S}_i(j),k) + \lambda_i = 0, \quad \forall j \in \mathbf{S}_i. \tag{14}$$

We can rewrite Eq. (14) as

$$\overline{C} \cdot \overline{W}_i^{\mathrm{T}} = \bar{q} \tag{15}$$

where

$$\overline{C} = \begin{bmatrix} 0 & \mathbf{1} \\ \mathbf{1} & 2C \end{bmatrix}$$

$$C = \begin{bmatrix} C_i(\mathbf{S}_i(1),\mathbf{S}_i(1)) & \cdots & C_i(\mathbf{S}_i(1),\mathbf{S}_i(n_i)) \\ \vdots & \ddots & \vdots \\ C_i(\mathbf{S}_i(1),\mathbf{S}_i(n_i)) & \cdots & C_i(\mathbf{S}_i(n_i),\mathbf{S}_i(n_i)) \end{bmatrix}$$

$$\overline{W}_i^{\mathrm{T}} = [\lambda_i W_i]^{\mathrm{T}}$$

$$W_i^{\mathrm{T}} = [w_{i\mathbf{S}_i(1)} w_{i\mathbf{S}_i(2)} \cdots w_{i\mathbf{S}_i(n_i)}]^{\mathrm{T}}$$

$$\bar{q} = [1, 0, \ldots, 0]^{\mathrm{T}}$$

Therefore, since C is nonsingular [16], W_i can be obtained straightforwardly from the following equation:

$$\overline{W}_i^{\mathrm{T}} = \overline{C}^{-1} \bar{q} \tag{16}$$

The constrained weights of equation obey an important symmetry that they are invariant to rotations, rescalings and translations for any particular data point and its neighbors. Thus, W is a sparse matrix that contains the information about the neighborhood relationship represented spatially by the position of the nonzero elements in the weight matrix and the contribution of one node to another represented numerically by their values. The DLLE neighbor selection process is summarized in Algorithm 1.

Algorithm 1. DLLE neighbor selection

1. Compute $\mathscr{D}$ from $\mathbf{X}$ ▷ $\mathscr{D} = \{d_{ij}\}$ is the distance matrix
2. Sort $\mathscr{D}$ along each column in ascending order to form $\mathscr{R}$
3. **for** $i \leftarrow 1, N$ **do**
4. **for** $k \leftarrow 1, K$ **do**
5. **if** $\alpha_i^k \leqslant N$ **then**
6. $\min(R_{ik}) = \ulcorner \alpha_i^k \urcorner + k - 1$
7. $\max(R_{ik}) = \ulcorner \alpha_i^{k+1} \urcorner + k$
8. **else** $\alpha_i^k > N$
9. break
10. **end if**
11. $\bar{d}_{ik} \leftarrow \alpha_i, k, R_{ik}$ ▷ by solving Eq. (7)
12. $x_j = \arg_{x_j \in R_{ik}} \min \|\bar{d}_{ik} - x_j\|^2$
13. $\mathbf{S}_i = \mathbf{S}_i \cup \{x_j\}$
14. $n_i = n_i + 1$
15. **end for**
16. $\bar{d}_i = \frac{1}{N} R_i$
17. **if** $x_j > \bar{d}_i$ **then**
18. $\mathbf{S}_i = \mathbf{S}_i - \{x_j\}$
19. $n_i = n_i - 1$
20. **end if**
21. **end for**

4.4. Compute embedding coordinates

Finally, we find the embedding of the original data set in the low-dimensional space, e.g., l dimension. Because of the invariance property of reconstruction weights w_{ij}, the weights reconstructing the ith data point in m dimensional space should also reconstruct the ith data point in l dimensional space. Similarly, this is done by trying to preserve the geometric properties of the original space by selecting l dimensional coordinates y_i to minimize the embedding function given below:

$$\begin{aligned} \Phi(Y) &= \sum_i^N \left\| y_i - \sum_{j=\mathbf{S}_{i(1)}}^{\mathbf{S}_{i(n_i)}} w_{ij} y_j \right\|^2 \\ &= \sum_i^N \|Y(I_i - W_i)\|^2 \\ &= \mathrm{tr}(Y(I_i - W_i)(Y(I_i - W_i))^{\mathrm{T}}) \\ &= \mathrm{tr}(YMY^{\mathrm{T}}) \end{aligned} \tag{17}$$

where w_{ij} are the reconstruction weights computed in Section 4.3, y_i and y_j are the coordinates of the point x_i and its neighbor x_j in the embedded space.

4.5. LLE, NLE and DLLE comparison

For the comparison of the embedding property, we have conducted several manifold learning algorithms tests on some test examples. Here we mainly illustrate three algorithms LLE, NLE and DLLE graphically using two classical data sets: two peaks and punched sphere. For each data set, each method is used to obtain a 2D embedding of the points. Figs. 5 and 6 summarize these embedding results. The data set is shown at the top left, in a 3D representation. For the two peaks data set, two corners of a rectangular plane are bent up. Its 2D embedding should show a roughly rectangular shape with blue and red in opposite corners. The punched sphere is the bottom 3/4 of a sphere which is sampled nonuniformly. The sampling is densest along the top rim and sparsest on the bottom of the sphere. Its intrinsic structure should be 2D concentric circles. Both the sample data sets are constructed by sampling 2000 points.

In Fig. 5, as expected, all the three algorithms can correctly embed the blue and red samples in opposite corners. However, the outline shape of the embedding using NLE is distorted when projected in 2D. DLLE can give a better preservation of the original rectangle compared to LLE. At the same time, the green samples performing as the inner and outer boundary are also well kept using DLLE.

As can be seen in Fig. 6, both DLLE and LLE are successful in flattening the punched sphere and recover all the original concentric circles. NLE seems to be confused about the heavy point density around the rim. It can preserve the inner circles well but fails on the outer circle because of its neighborhood selection criteria.

5. Probabilistic neural network

According to the DLLE algorithm, the distances between the projected data points in low-dimensional space depend on the similarity of the input images. The images which are similar are projected with a small distance while the images that differ greatly are projected with a large distance. Based on the distances in low-dimensional space, we use the neural network to classify different gesture images. Neural networks are widely employed in pattern

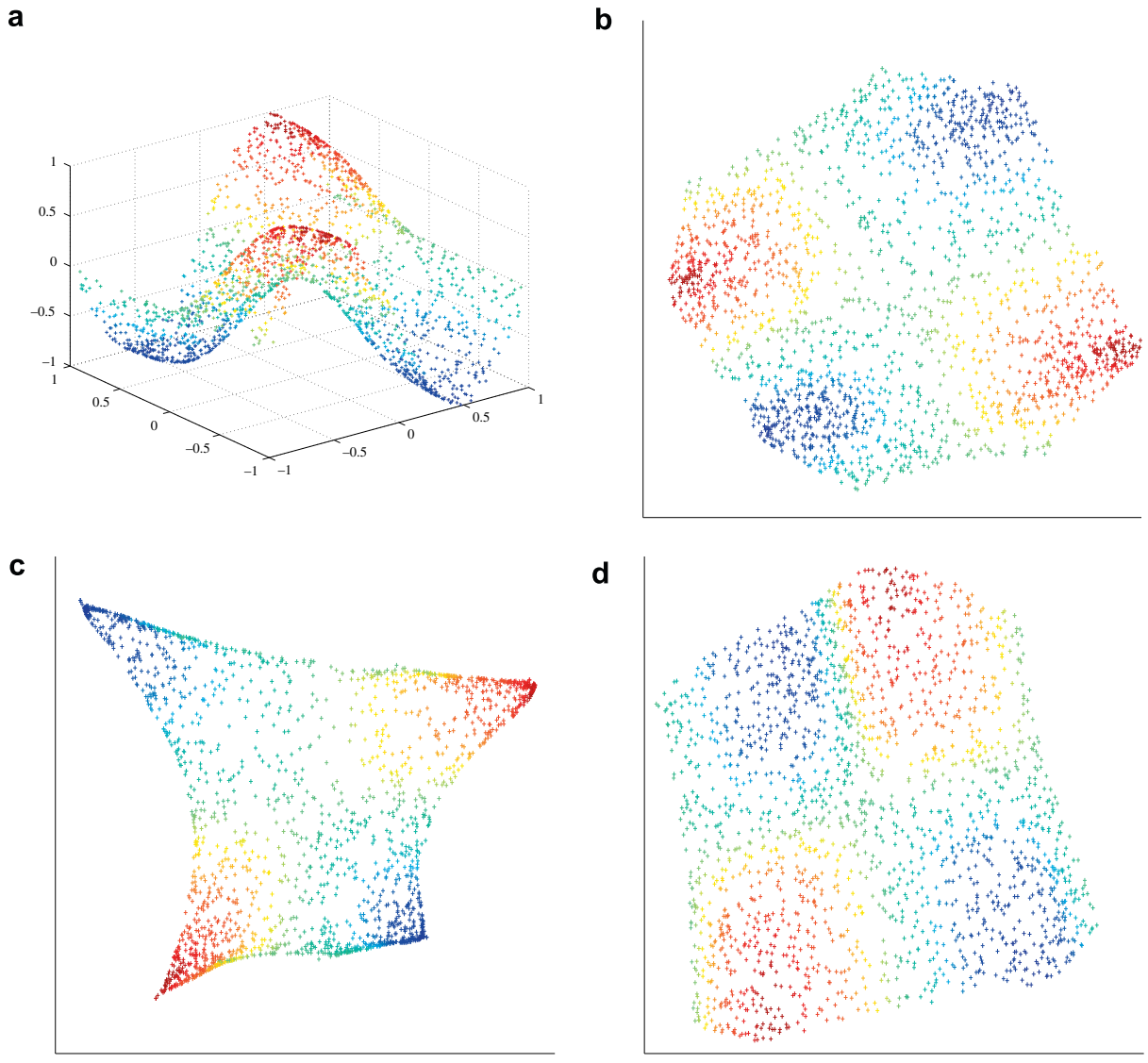

Fig. 5. (a) Two peaks, (b) LLE, (c) NLE and (d) DLLE.

recognition based on learning from samples [20–22]. PNN is selected in this study as the classifier because of its rapid training speed, good accuracy, robustness to weight changes and negligible retraining time.

Consider a general classification problem: the membership of a multivariate random vector $X = [X_1, X_2, \ldots, X_p]$ needs to be classified into one of the M possible categories. The basic idea of PNN is to approximate the unknown discrete distribution $P_i(X)$ of each class by a finite mixture of product components.

It is critical for the accuracy of decision boundaries to estimate probability density function from the training samples. The Parzen window method can be used for estimating probability density function from a random sample [23]. Cacoulos extended Parzen's method to cover the multivariate case [24]. The most common multivariate estimator can be expressed as

$$f_i(X) = \frac{1}{(2\pi)^{p/2}\sigma^p}\frac{1}{n}\sum_{i=1}^{n}\exp\left[-\frac{\|X - X_i\|^2}{2\sigma^2}\right] \tag{18}$$

where p is the dimensionality of the input vector, n is the number of samples in each category, $f_i(X)$ is the sum of small multivariate Gaussian distributions centered at each training sample, and σ is a smoothing factor defining the width of the area of influence, which softens the surface defined by the multiple Gaussian functions. A small value of σ may cause a very spiky approximation which cannot generalize well while a large value produces a greater degree of interpolation between points. However, there is no general method to determine σ and it is usually determined by minutely changing its value and examining the corresponding recognition accuracy. We will give a detailed discussion on σ selection in Section 8.1.

The PNN architecture shown in Fig. 7 consists of four layers: an input layer, a pattern layer, a summation layer and an output layer. The input neurons merely distribute the same inputs values to all of the pattern units. Each pattern neuron performs a dot product of the input pattern vector with a weight vector in the interconnections, followed by a neuron activation function. The pattern layer represents a neural implementation of a Bayes classifier. There is one summation neuron for each class. The summation layer sums the inputs from the pattern layer neurons corresponding to the category from which the training pattern is selected. The output neuron is a threshold discriminator. The maximum neuron of summation layer is selected as the output.

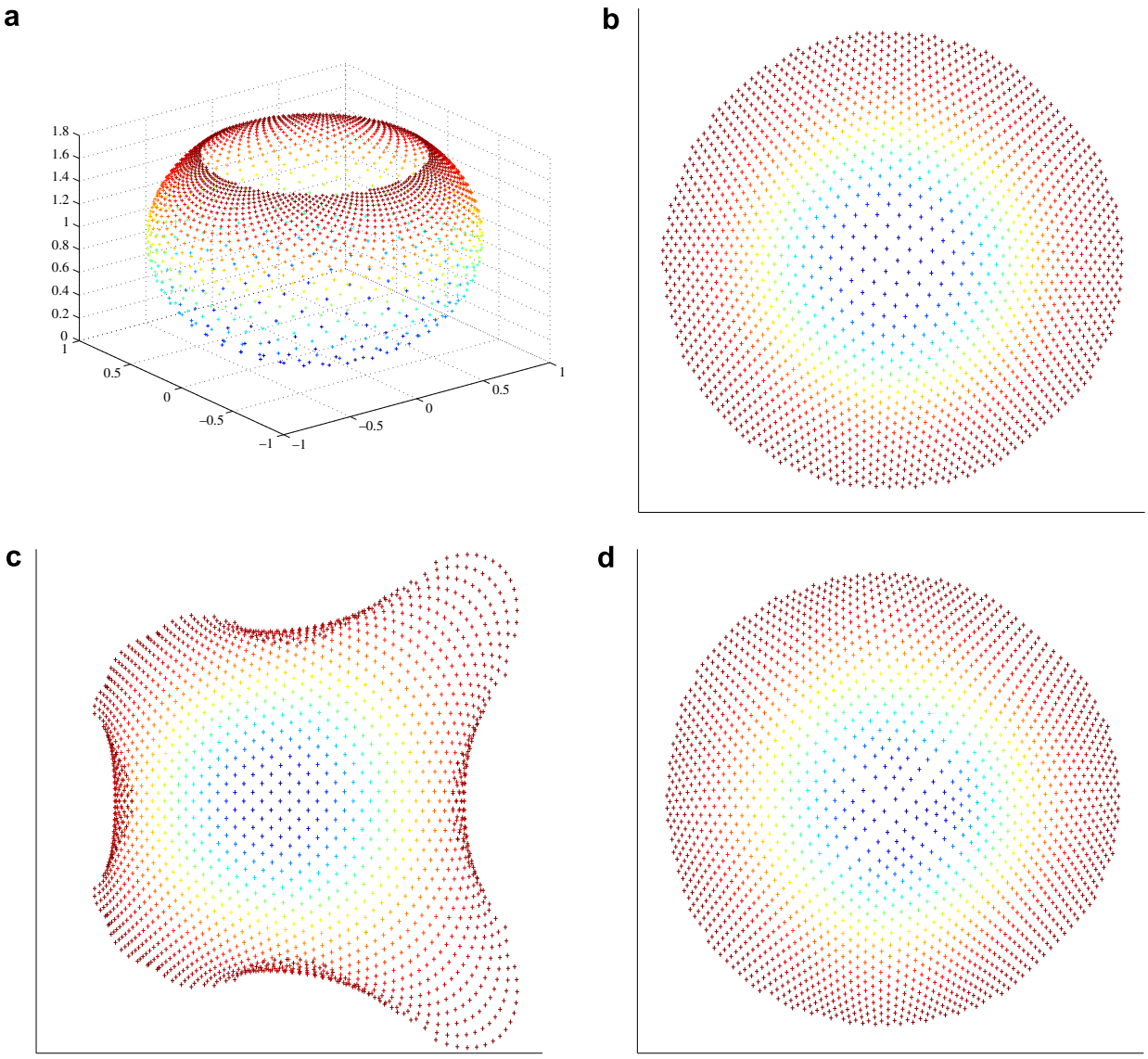

Fig. 6. (a) Punched sphere, (b) LLE, (c) NLE and (d) DLLE.

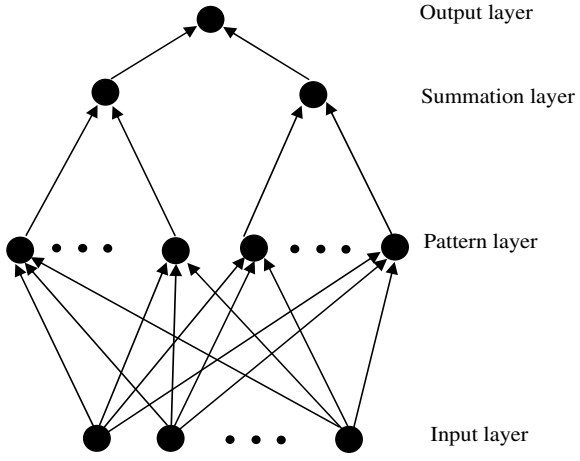

Fig. 7. The architecture of PNN.

After the images are projected to low dimension, they are fed into the PNN for training. The new input data is classified through PNN. The output is the home gesture group which the input image belongs to.

6. Model reconstruction and constraints

We model the skeleton of human hand with a hierarchical structure which was first proposed in [25]. Each finger is considered as a kinematic chain consisting of rigid links and joints with a common base frame at the palm. According to the natural property of the finger, each joint has one or two rotational DOF. Our hand model is shown in Fig. 8 with the name and DOF of each joint. Each joint is described in a 3D coordinates to indicate its rotational DOF. The superscript of each symbol represents the rotational DOF along the associated axis, and the subscript is the name of the joint.

Modeling motion constraints is crucial to the effective and efficient reconstruction of the hand model [9,26,27]. The natural anatomical properties of the human hand implicitly determine some motion constraints. For example, human fingers are impossible to bend backward which limits the range of figure motion.

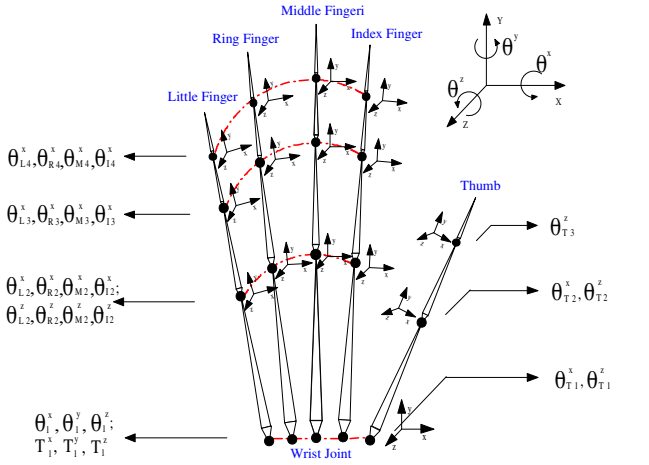

Fig. 8. Kinematic hand model with 27 DOF. The rotational joints are denoted by θ while the translational joints are denoted by T.

Here we analyze some most commonly used constraints and present some constraints based on our observation to reduce the DOF.

Hand motion constraints can be classified into three categories. Type I constraints are the limits of finger motions as a result of hand anatomy. These constraints are usually referred to static constraints. Type II constraints are the limitations imposed on hand joints during finger motions which are usually called dynamic constraints. Type III constraints are usually imposed while performing natural hand motion. A comprehensive study of hand/finger motion constraints is given below.

6.1. Type I constraints

Type I constraints are usually given by inequalities according to the motion range of fingers. These constraints limit hand articulation within a boundary and provide the virtual model with a more natural appearance.

- *Constraint 1.* The interphalangeal joint of hand can only rotate around the joint. Their rotational limits are given below:

$$\begin{aligned} 0^\circ &\leqslant \theta^x_{(I,M,R,L)2} \leqslant 90^\circ \\ 0^\circ &\leqslant \theta^x_{(I,M,R,L)3} \leqslant 120^\circ \\ 0^\circ &\leqslant \theta^x_{(I,M,R,L)4} \leqslant 90^\circ \end{aligned} \tag{19}$$

The thumb's second joint rotational limits can be given by

$$0^\circ \leqslant \theta^z_{T3} \leqslant 90^\circ \tag{20}$$

- *Constraint 2.* The constraints of waist joint can be approximated as follows according to experimental observations:

$$\begin{aligned} 0^\circ &\leqslant \theta^x_1 \leqslant 90^\circ \\ -15^\circ &\leqslant \theta^z_1 \leqslant 15^\circ \end{aligned} \tag{21}$$

- *Constraint 3.* The metacarpophalangeal joints of the thumb and middle finger rarely display abduction/adduction motion and can be approximated to be zero [25]. These constraints are given as follows:

$$\begin{aligned} \theta^x_{T2} &= 0^\circ \\ \theta^z_{(I,M,R,L)2} &= 0^\circ. \end{aligned} \tag{22}$$

6.2. Type II constraints

These types of constrain usually have a closed form representation. These constraints are determined by the inter-finger or intra-finger structure. The inter-finger constraints are used to describe the dependency between joints of different fingers. The intra-finger constraints refer to the relations between different joints of the same finger.

- *Constraint 4.* This constraint is used to describe the dependency between different joints of the same finger caused by the tendon [28]. For the index, middle, ring and little finger, the Distal Interphalangeal cannot bend independently for a natural gesture. We must bend the Proxiamal Interphalangeal joint at the same time. Based on experimental observation, their relationship can be reasonably approximated as below:

$$\theta^x_{(I,M,R,L)4} = \frac{3}{4}\theta^x_{(I,M,R,L)3} \tag{23}$$

- *Constraint 5.* The thumb's movement is more complicated because a large part of the thumb is part of the palm. From experimental observation, the inter-finger constraint of the thumb can be given by

$$\theta^x_{T1} = 2\left(\theta^z_{T2} - \frac{1}{6}\pi\right) \tag{24}$$

- *Constraint 6.* This constraint is used to describe dependency between the Interphalangeal joint and Metacarpophalangeal joint of thumb. In our model, it can be approximated by the form below based on experimental observation

$$\theta^x_{(I,M,R,L)4} = \frac{3}{4}\theta^x_{(I,M,R,L)3} \tag{25}$$

- *Constraint 7.* This constraint is used to describe dependency between the Proxiamal joint and Metacarpophalangeal joints of index, middle, ring and little finger. In our model, it is approximated as shown below to match the experimental observation

$$\theta^x_{(I,M,R,L)3} = k\theta^x_{(I,M,R,L)2} \tag{26}$$

This constraint is called "weak constraint" where the coefficient may vary from time to time. In our experiment, we adjust the coefficient according to the value of the Metacarpophalangeal joints with an initial value set to 0.5.

6.3. Type III constraints

- *Constraint 8.* This constraint defines the four joints of the thumb (Thumb tip, Trapeziometacarpal joint, Metacarpalangeal joint and Interphalangeal joint) to be coplanar [10]. This plane can be defined as the "Thumb Plane".

6.4. Finger control

It is very tedious and difficult to attain a desired posture by extracting all the joints' parameters from images. Higher level control of the fingers is used in our system to ease the burden to compute high DOF of the hand model and to prevent the unnatural gestures from occurring.

In human motion, there are many correlations between joint actions. According to our observation, for most of the nature gestures of our hand, fingers are closely related. For a set of such fingers, their moving or bending directions are quite similar to each other. Usually, these correlations are especially clear for motions like splaying and grasping. For example when the hand is splaying, all of the finger joints angles increase at nearly the same rate. Thus these joints can be approximated by a series of related parameters. Fingers that are closely related with each other are grouped together. The groups can be composed of one finger, two fingers to five fingers. The user's gestures formed by several fingers corre-

sponding to a certain group can be recorded and then reconstructed. However, due to the anatomy of the human hand, not all finger combinations are meaningful, only certain finger groups are natural. We predefine 14 home gestures which are the most commonly used ones. The dynamic gesture motions are derived from these home gestures. In one gesture motion, the user can move the fingers of this group and the system can detect and change the virtual hand postures accordingly.

7. Hand recognition and tracking

7.1. Image preprocessing

In vision-based gesture recognition, the format of the captured images may vary greatly. In our system, the resolution of the acquired images is 160×120 pixels. It is captured with a roughly uniformly distributed background. The following processes are taken to eliminate noises and make it easier to find the compact representations from these images. First, the color images are converted from RGB to grayscale. Then the background is subtracted from each image to eliminate noises and to enhance hand's outline. Finally, the images are adjusted by mapping the values in intensity image Img_i to new values in image Img_j such that values between low-limit and high-limit of Img_i map to values between low-limit and high-limit of Img_j. This process can be described by the following equation. This adjustment process can increase the contrast of the output image Img_j:

$$\mathrm{value}_J = (\mathrm{value}_I - \mathrm{low}_I)\frac{\mathrm{high}_J - \mathrm{low}_J}{\mathrm{high}_I - \mathrm{low}_I} \tag{27}$$

7.2. Framework of recognition and tracking

The framework of the system is shown in Fig. 9. Two modules are considered in the system:

Static home gesture recognition. The home gesture recognition is carried out when initialization or re-initialization takes place. The home gesture database containing 14 home gestures has been first set up as $\{g_i | i = 1, 2, \ldots, 14\}$, where g_i denotes home gesture i. Then the input image is classified to a certain group i using DLLE–PNN according to the database. This indicates which motion will be followed.

Dynamic motion tracking. In this part, the hand posture's movements are captured by the camera. The captured image is dynamically mapped to the low-dimensional space using DLLE in the corresponding motion database. For each gesture process, a database containing 30 sample images is set up as $\{m_{ij} | i = 1, 2, \ldots, 14; j = 1, 2, \ldots, 30\}$, where m_{ij} denotes the image with its home gesture in group i and its sequence in this motion database at j, to further compute the gesture parameters. These samples are dynamically captured as one gesture motion process in sequence. According to the neighborhood relationships among the gesture sequence and the predefined model joints parameters in the database, the reconstruction joint parameters can be computed.

7.3. Static home gesture recognition

Hand gesture recognition consists of deciding whether the test images (different from the training set) containing the hand gesture is of interest or not. However, due to the complexity of human hand-self occlusion, high DOF and gestures variety, it is difficult to distinguish each hand gesture. To simplify this process, we define a finite number of gestures as the home gestures. These gestures contain the commonly used and natural hand gestures. Fig. 10 shows these home gestures used in the system.

According to the DLLE algorithm, neighborhood relationship can be preserved in the low dimension data set. The distances between the projected data points in low-dimensional space depend on the similarity of the input images. The images that are similar are projected with a small distance while the images that differ greatly with each other are projected with a large distance. Therefore, images of the same gesture are closer than images of different gestures in low-dimensional space. At this time, the training samples of the same gesture are "half clustered" and only a few of them may be apart from their corresponding cluster which can be seen from Figs. 13 and 14. This makes it easier for the neural network to classify different gestures. Training sample images are preprocessed and reconstructed in the low-dimensional space, then they are fed into PNN. After training, images of the same gesture are grouped and the neural network is set up for further classification.

7.4. Dynamic motion tracking

Because of the neighborhood preserving property of DLLE algorithm, the adjacent relationship of the motion sequence images are

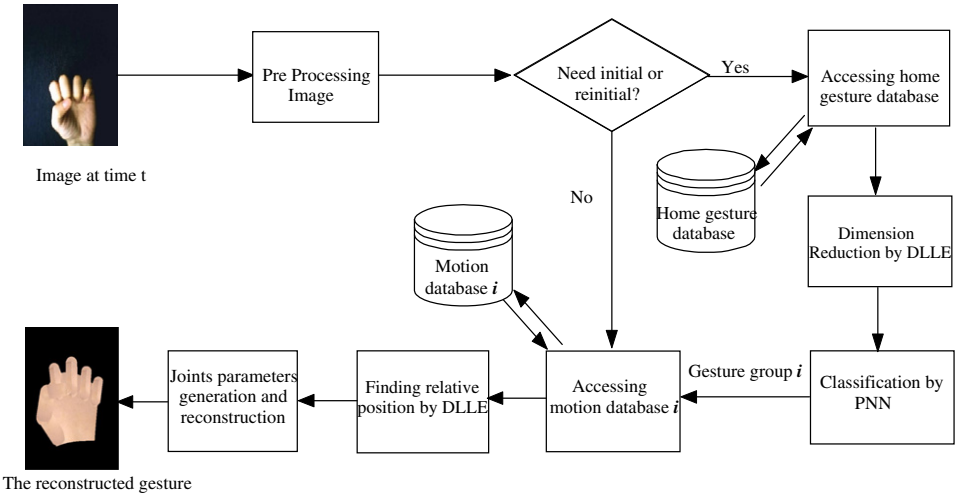

Fig. 9. System overview.

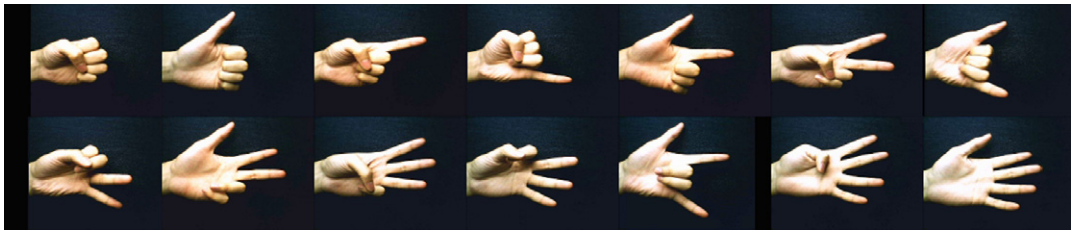

Fig. 10. Base gesture models.

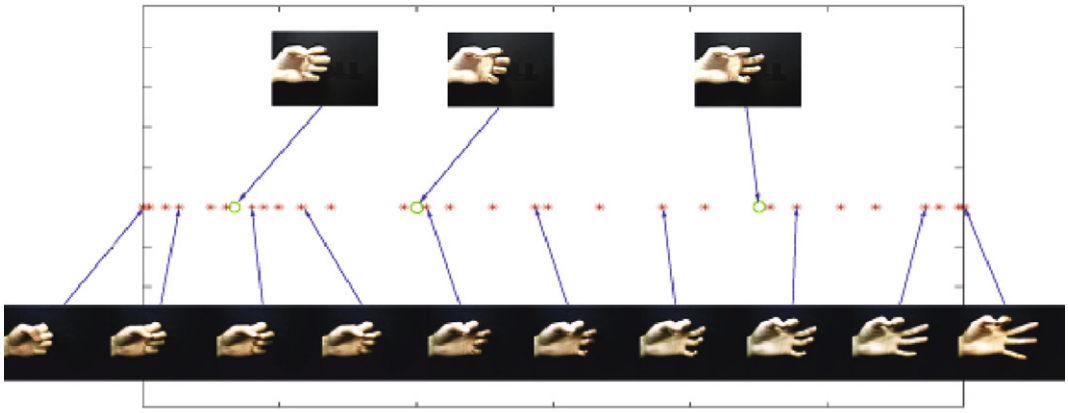

Fig. 11. The sequence images in one motion database and input images with their mapped relative positions. The stars represent the database sequence images and the circles are the test images sampled from the gesture motion. We can see that the motion sequence is well kept after projected by DLLE and the input images' relative positions are appropriately approximated.

Fig. 12. Variation of full extended hand images. The above are three hand scales under three lighting conditions. The below are the variations about different axes.

well kept in low dimension (Fig. 11). The reconstruction model joint parameters can be assigned according to the input gesture's relative position and its adjacent neighbors joints parameter set. When the motion tracking is started, the user's gesture image is dynamically put into the motion database. It is mapped to low dimension and it's relative position in the database can be found as illustrated in Fig. 11. According to its relative position, the new 3D model is constructed by finding the linear interpretation of the joint parameters of its adjunct neighbors. The hand joint constraints described in Section 6 are applied on finger joint

accordingly during reconstructing to prevent the unnatural generation of hand gestures.

According to the Assumption 2, an input gesture image Img_k between the gesture image Img_i and Img_j has its embedding ϕ_k lying between ϕ_i and ϕ_j in low dimension. To approximate the relative position of the input image in the database, we adopt the following equation:

$$\phi_i' = k\sqrt{(\phi_i - \phi_{\min})/\phi_{\max}}, \quad \phi_{\max} \neq 0, \quad i = 1, 2, \ldots, n \tag{28}$$

where the $\phi_{\min}$ and $\phi_{\max}$ are the minimal and maximal value of the given value in low dimension, ϕ_i is the input image embedding in low dimension, ϕ_i' is the computed relative position, and k is scaling constant, default value is set to 1.0. Usually, for a gesture motion sequence, hand at a rest position may be over-sampled while the fast transforming process may be under-sampled. Compared to the first order linear equation to approximate the given points, the approximation above can follow the finger movement more naturally and make the tracking animation more realistic.

8. Experiments

In this section we present the results of experiments using the proposed static home gesture recognition and dynamic motion methods. The system is executed on a PC with Pentium IV 2.8G CPU and 512M RAM. Our experiments are carried out under the following assumptions:

- The hand gesture in the image should be centered with a variation of no more than 15% along both directions of image space.
- The user's hand should be kept stationary during the time when the initialization or re-initialization takes place.
- While tracking, the user should avoid much global movement. Sudden, jerky hand movements should also be avoided.

We first perform a series of off-line captures to create a large home gesture database. The database contains 1120 images of 14 different type of home gestures. These samples are used for training the PNN. Apart from the training samples, another 20 samples of each gesture are employed to be tested. The hand gesture can have a variation of 15% along the horizontal (X) axis and the verti-

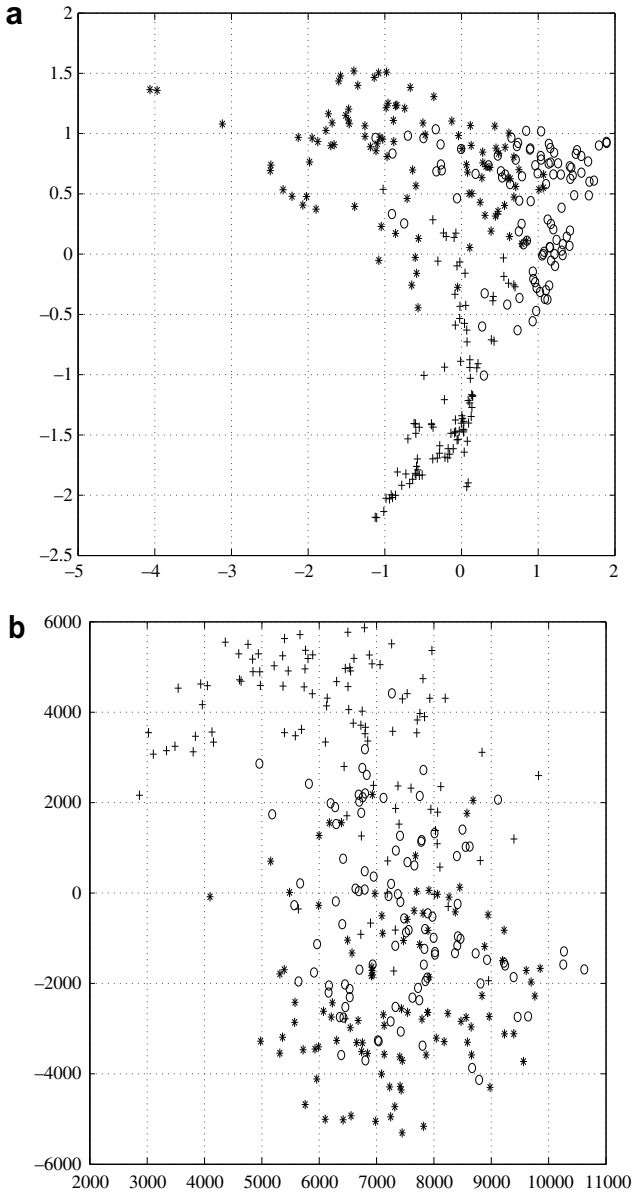

Fig. 13. Two-dimensional embeddings of three hand gestures. (a) Results from DLLE and (b) results from PCA.

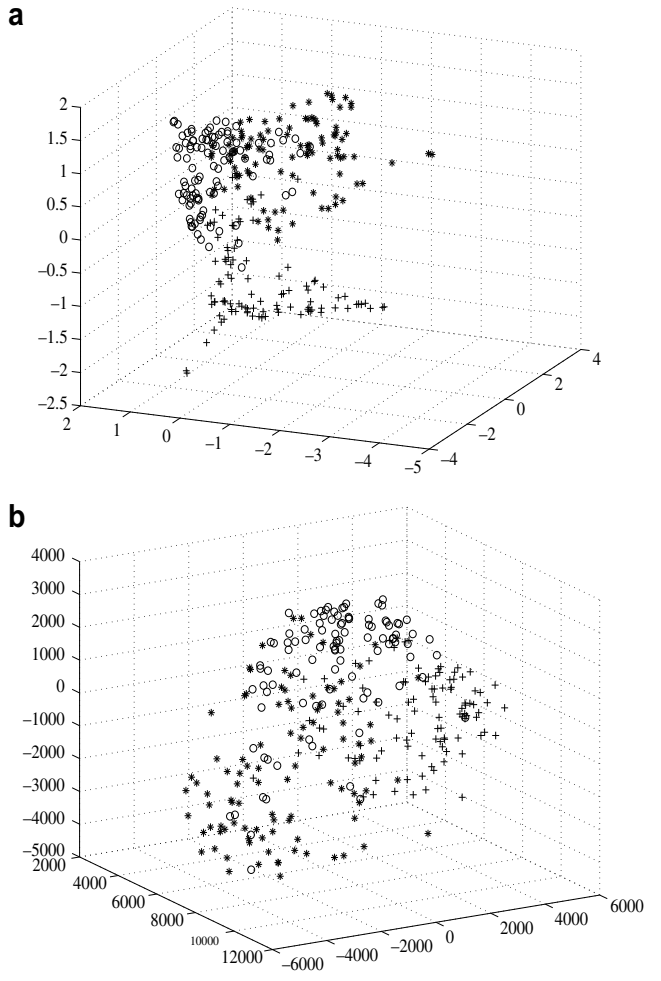

Fig. 14. Three-dimensional embeddings of three hand gestures. (a) Results from DLLE and (b) results from PCA.

cal (Y) axis of the world coordinate about the center of the image. While the orientation of the gesture can have a variety about -10°—10° around the X, Y, Z axis of the world coordinate. The images are captured under three lighting conditions, three hand scales and various orientations within the above range which are shown in Fig. 12.

8.1. Static recognition

We first compare the properties of the DLLE and PCA after the sample images are mapped to low dimension. After the projection, the projected low dimension data should keep the neighborhood relations of the original images. Images of the same gesture should cluster together while different gesture images should be apart. Fig. 13 compares the two-dimensional embeddings obtained by DLLE and PCA for $N = 100$ samples from three hand gestures, respectively. For clarity, we only adopt three home gestures for comparison. The circle, cross and star points represent three different kinds of gesture samples. We can see from Fig. 13(a) that for $l = 2$, the embedding of DLLE separates the three gestures quite well. Samples of the same gesture clustered together while only a few different gesture samples are overlapped. Fig. 13(b) shows that the three gesture samples overlapped seriously. PCA cannot separate the three gestures in any meaningful way in 2D.

Fig. 14 compares the three-dimensional embeddings obtained by DLLE and PCA for $N = 100$ samples of each gestures. From Fig. 14(a) we can see that for $l = 3$, the embedding of DLLE can keep the similarity of each gesture samples and separate the three gestures quite well in three-dimensional space. As seen in Fig. 14(b), the projected sample points by PCA are still in a chaos. The circle samples and the cross samples are mixed together. PCA fails again to cluster the three gestures in 3D.

We test how the projected dimensionality l affects the recognition rate when using PNN. Generally speaking, the higher the projected dimension, the more feature information is kept in the projected data set. Fig. 15 demonstrates the relationship of the projected dimension l of DLLE and recognition rate. For the flat portion of each diagram, the average recognition rate for $l = 5$ is above 70%, for $l = 10$ is above 80%, for $l = 30$ is above 90% and for $l = 50$ is around 90%. We can see that with the increase of the l, the recognition rate can be improved dramatically until l is over 30, the improvement becomes unconspicuous. It means that the projected dimension is sufficiently large. The recognition accuracy has little improvement when l is over 30. Thus, we take $l = 30$ as the trade-off between efficiency in computing and satisfactory performance (Table 1).

Fig. 15 also illustrates the recognition rate versus the value of the smoothing factor σ of PNN. For each specific l, the parameter σ is adjusted from small to large to obtain the "best" results. It is clear that with the increase of σ, the recognition rate will rise until the peak is reached and then it descends slowly. We can see from Fig. 15(b) to 15(d), the diagnostic accuracy can remain sufficiently high with a board range of σ. So by comparing the highest recognition accuracy, it is not difficult to obtain the optimal σ. For $l = 30$,

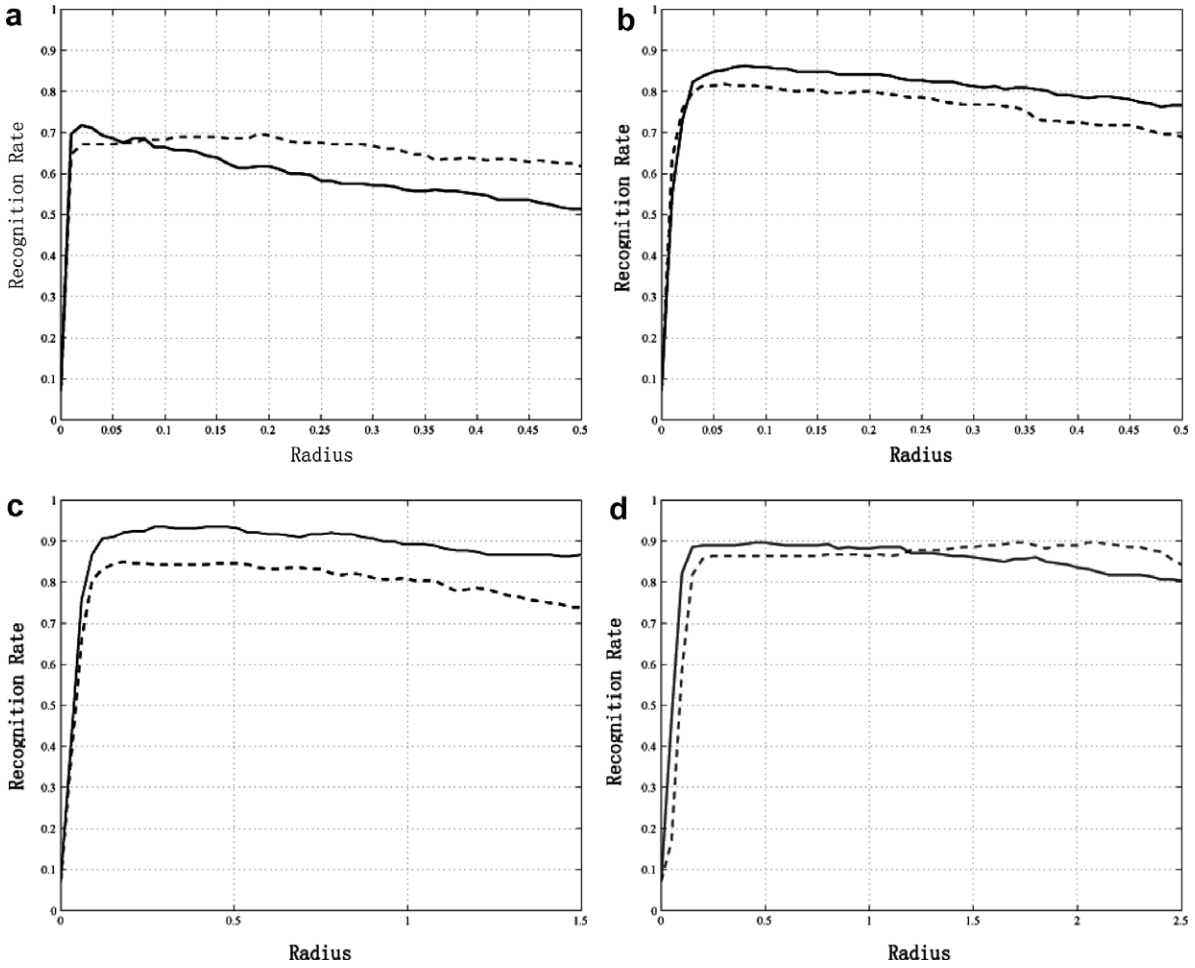

Fig. 15. The recognition rate with respect to σ of the PNN. The solid line represents recognition rate using DLLE while the dashed line represents LLE. (a) Recognition rate for $l = 5$, (b) recognition rate for $l = 10$, (c) recognition rate for $l = 30$ and (d) recognition rate for $l = 50$.

Table 1
Hand gestures recognition results using DLLE

No. of training samples	No. of test samples	Recognition rate (%)	Gesture classes	Dimension
800	200	85.5	10	19
1120	280	71.8	14	5
1120	280	82.1	14	10
1120	280	88.6	14	18
1120	280	93.2	14	30

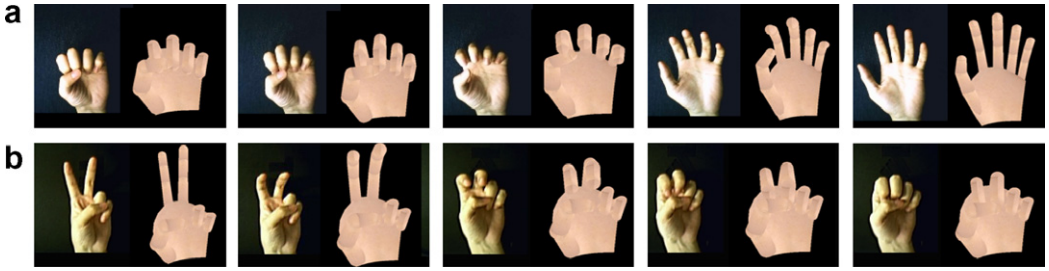

Fig. 16. Dynamic tracking of two hand motions from left to right: (a) from a fist to a palm and (b) from two fingers to a fist.

when σ equals to 0.27, the recognition rate can reach up to 93.2%. Compared to other methods in the literature, in [29], the minimal recognition rate of the distinct hand gestures is about 60–85% using contour discriminant-based static shape recognition method. The recognition rate achieves 91.9% with 14 predefined gestures in [30] using hidden Markov models (HMM) and recurrent neural networks (RNN).

In addition, we examine the recognition rate using our DLLE and LLE algorithm by comparing the highest part of both curves. For $l = 5$, DLLE has a better peak recognition rate of 71.8% while LLE's is nearly 70%. For $l = 10$ and $l = 30$, DLLE has a better recognition rate than LLE both on average and peak value. For $l = 50$, DLLE and LLE's performance are more or less the same. Our proposed algorithm DLLE does not need the trial-and-error process and does not need to specify how many neighbors must be found for reconstruction. Therefore, the accuracy of the recognition rate is increased.

8.2. Dynamic tracking

The tracking system can reach a rate up to two frames per second. Our system can track the hand motion and follow the movement naturally. The experimental results are given in Fig. 16. The image captured by the camera is on the left, while the corresponding model reconstructed using our method is shown on the right. From the reconstruction model, we can see that our model can follow the gesture motion quite well. The reconstructed model may not follow the gesture image exactly and there may be some difference between the captured image and reconstruction model. However, it is similar enough for modeling and tracking.

9. Conclusions

This paper is concerned with the problem of static hand gesture recognition and dynamic gesture tracking. An unsupervised learning algorithm, DLLE, has been developed to discover the intrinsic structure of the data. These discovered properties were used to compute their corresponding low-dimensional embedding. Associated with PNN, a high recognition accuracy algorithm has been developed for static hand gesture recognition. The neighborhood preserving property of DLLE preserved the adjacent relationship of the motion sequence images in low-dimensional space. For dynamic gesture tracking, we made use of the similarity measures among the images, so that hand gesture motion from a sequence of images can be tracked and dynamically reconstructed according to the image's relative position in the corresponding motion database. Experimental results showed that our approach was able to successfully separate different hand postures and track the dynamic gesture.

References

[1] D.J. Sturman, D. Zeltzer, A survey of glove-based input, IEEE Comput. Graph. Appl. 14 (1) (1994) 30–39.
[2] G.D. Kessler, L.F. Hodges, N. Walker, Evaluation of the CyberGlove as a whole-hand input device, ACM Trans. Comput. Hum. Interact. 2 (4) (1995) 263–283.
[3] S.S. Fels, G.E. Hinton, Glove-talkii: an adaptive gesture-to-formant interface, in: Proceedings of the Computer Human Interaction 1995, SIGCHI95, Denver, CO, May 1995, pp. 456–463.
[4] Y. Wu, T.S. Huang, Vision-based gesture recognition: a review, in: Gesture Workshop, 1999, pp. 103–115.
[5] T. Darrell, I.A. Essa, A. Pentland, Task specific gesture analysis in real time using interpolated views, IEEE Trans. Pattern Anal. Mach. Intell. 18 (12) (1996) 1236–1242.
[6] J.M. Rehg, T. Kanade, Visual tracking of high DOF articulated structures: an application to human hand tracking, in: Proceedings of the European Conference on Computer Vision, vol. 2, 1994, pp. 35–46.
[7] V. Athitsos, J. Alon, S. Sclaroff, G. Kollios, Boostmap: a method for efficient approximate similarity rankings, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Washington, DC, 2004.
[8] V.I. Pavlovic, R. Sharma, T.S. Huang, Visual interpretation of hand gestures for human–computer interaction: a review, IEEE Trans. Pattern Anal. Mach. Intell. 19 (7) (1997) 677–695.
[9] J. Lee, T.L. Kunii, Model-based analysis of hand posture, IEEE Comput. Graph. Appl. (1995) 77–86.
[10] C.S. Chua, H. Guan, Y.K. Ho, Model-based 3D hand posture estimation from a single 2D image, Image Vision Comput. 20 (2002) 191–202.
[11] T.F. Cootes, C. Taylor, D. Cooper, J. Graham, Active shape models their training and application, Comput. Vision Image Understand. 61 (1995) 38–59.
[12] M.J. Black, A.D. Jepson, Eigentracking: robust matching and tracking of articulated objects using a view-based representation, in: Proceedings of the European Conference on Computer Vision, vol. 1, 1996, pp. 329–342.
[13] N. Gupta, P. Mittal, S.D. Roy, S. Chaudhury, S. Banerjee, Developing a gesture-based interface, IEEE J. Res. (2002) 237–244.
[14] J.B. Tenenbaum, V. de Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (2000) 2319–2323.
[15] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (2000) 2323–2326.
[16] S.S. Ge, F. Guan, A.P. Loh, C.H. Fua, Feature representation based on intrinsic discovery in high dimensional space, in: Proceedings of the 2006 IEEE International Conference on Robotics and Automation, May 2006, pp. 3399–3404.
[17] D.F. Specht, Probabilistic neural network, Neural Networks 3 (1990) 109–118.
[18] D.A. Forsyth, J. Ponce, Computer Vision: A Modern Approach, Prentice Hall, Upper Saddle River, New Jersey, 2002.
[19] L.K. Saul, S. Roweis, Think globally, fit locally: unsupervised learning of low dimensional manifolds, J. Mach. Learn. Res. 4 (2003) 119–155.

[20] P.P. Raghu, B. Yegnanarayana, Supervised texture classification using a probabilistic neural network and constraint satisfaction model, IEEE Trans. Neural Networks 9 (1998) 516–522.
[21] Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin, A neural probabilistic language model, J. Mach. Learn. Res. 3 (2003) 1137–1155.
[22] B.L. Zhang, H. Zhang, S.S. Ge, Face recognition by applying wavelet subband representation and kernel associative memory, IEEE Trans. Neural Networks 15 (2004) 166–177.
[23] E. Parzen, On the estimation of a probability density function and mode, Ann. Math. Stat. 33 (1962) 1065–1076.
[24] T. Cacoullos, Estimation of a multivariate density, Ann. Inst. Stat. Math. (Tokyo) 18 (1966).
[25] J. Lee, T.L. Kunii, Constraint-based hand animation, Models Tech. Comput. Anim. (1994) 110–127.
[26] J. Kuch, T. Huang, Vision based hand modeling and tracking for virtual teleconferencing and telecollaboration, in: Proceedings of the International Conference on Computer Vision, vol. 8(6), 1995, pp. 666–671.
[27] J. Lin, Y. Wu, T.S. Huang, Modeling the constraints of human hand motion, in: Proceedings of the Workshop on Human Motion, IEEE Computer Society, Washington, DC, USA, 2000, pp. 121–126.
[28] H. Rijpkema, M. Girard, Computer animation of knowledge-based human grasping, IEEE Comput. Graph. Appl. 25 (4) (1991) 339–348.
[29] A. Ramamoorthy, N. Vaswani, S. Chaudhury, S. Banerjee, Recognition of dynamic hand gestures, Pattern Recogn. 36 (9) (2003) 2069–2081.
[30] C.W. Ng, S. Ranganath, Real-time gesture recognition system and application, Image Vision Comput. 20 (2002) 993–1007.