An Appearance-Based Framework for 3D Hand Shape Classification and Camera Viewpoint Estimation

Vassilis Athitsos and Stan Sclaroff
Computer Science Department
Boston University
111 Cummington Street
Boston, MA 02215
email: {athitsos, sclaroff}@cs.bu.edu

Abstract

An appearance-based framework for 3D hand shape classification and simultaneous camera viewpoint estimation is presented. Given an input image of a segmented hand, the most similar matches from a large database of synthetic hand images are retrieved. The ground truth labels of those matches, containing hand shape and camera viewpoint information, are returned by the system as estimates for the input image. Database retrieval is done hierarchically, by first quickly rejecting the vast majority of all database views, and then ranking the remaining candidates in order of similarity to the input. Four different similarity measures are employed, based on edge location, edge orientation, finger location and geometric moments.

1 Introduction

Techniques that allow computers to understand the shape of a human hand in images and video sequences can be used in a wide range of applications. Some examples are human-machine interfaces, automatic recognition of signed languages and gestural communication, non-intrusive motion capture systems, video compression of gesture content, and video indexing.

Different levels of accuracy are needed by different applications. In certain domains it suffices to recognize a few different shapes, observed always from the same viewpoint [23, 13, 6]. On the other hand, 3D hand pose estimation can be useful or necessary in various applications related to sign language recognition, virtual reality, biometrics, and motion capture. Currently, systems requiring accurate 3D hand parameters tend to use magnetic tracking devices and other non vision-based methods [14, 15, 19]. Computer vision systems that estimate 3D hand pose typically do it in the context of tracking [17, 7, 26, 20] . In that context, the pose can be estimated at the current frame as long as the system knows the pose in the previous frame. Since such trackers rely on knowledge about the previous frame, they need to be manually initialized, and they cannot recover when they lose the track.

The tracking system presented in [20] uses, like our system, a database of synthetic views and an appearance-based method to find the closest match to the observed input. The hand parameters of the previous frame are assumed to be known, so only poses close to those parameters need to be considered.

A machine learning system that estimates hand pose is described in [18]. The training set consists of synthetic renderings of an artificial hand model. Good performance is reported on a synthetic test set, but no quantitative results are given for real hand images.

In this paper we present a method for estimating 3D hand shape and orientation by retrieving appearance-based matches from a large database of synthetic views. The hand shape in the input image is assumed to be close to one of 26 predefined shapes (Figure 2). The database views are computer-generated renderings of the 26 hand shape prototypes from viewpoints that are sampled uniformly along the surface of the viewing sphere. The advantage of using appearance-based matching for 3D parameter estimation is that the estimation is done indirectly, by looking up the ground truth labels of the retrieved synthetic views. This way we avoid the ill-posed problem of recovering depth information directly from the input image.

Our framework has two main advantages over previous appearance-based methods for hand shape recognition [16, 22, 5, 25]: it can handle images from arbitrary viewpoints, and, in addition to classifying hand shape, it provides estimates for the camera orientation. In [1] we presented an early implementation of our framework, in which the chamfer distance ([3]) between edge images was used to estimate similarity between the input image and the database views. In this paper we present additional similarity measures (Section 3), we introduce a method to combine different measures, and we describe a hierarchical retrieval algorithm that first quickly rejects the vast majority of the database views and then ranks the remaining views in order of similarity to the input (Section 4). Compared to the approach described in [1], experiments with our current system demonstrate higher accuracy and vast improvements in retrieval time.

2 Proposed Framework

We model the hand as an articulated object, consisting of 16 links: the palm and 15 links corresponding to finger

0-7695-1602-5/02 $17.00 © 2002 IEEE

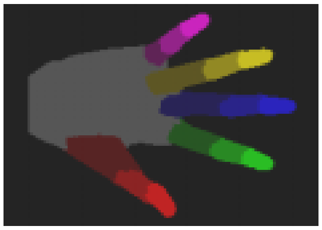

Figure 1: The hand as an articulated object. The palm and each finger are shown in a different color. The three different links of each finger are shown using different intensities of the same color.

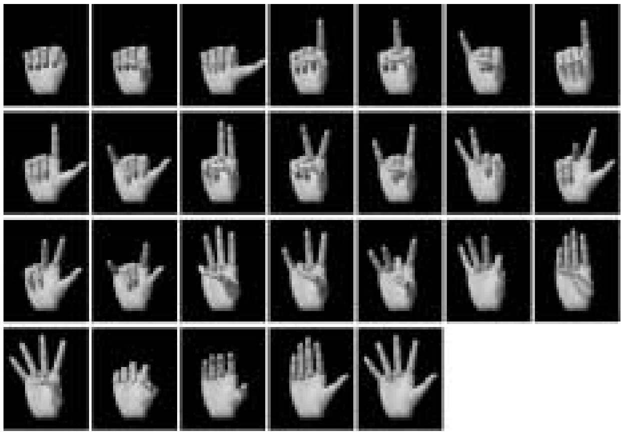

Figure 2: The 26 basic shapes used to generate training views in our database.

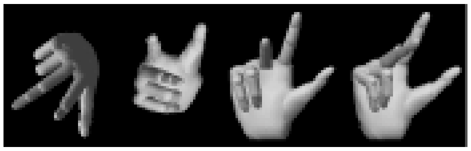

Figure 3: Four different database views of the same basic shape.

parts. Each finger has three links (Figure 1). There are 15 joints, each connecting a pair of links. The five joints connecting fingers to the palm allow rotation with two degrees of freedom (DOFs), whereas the 10 joints between finger links allow rotation with one DOF. For the 20-dimensional vector containing those 20 DOFs we use synonymously the terms "internal hand parameters," "hand shape" and "hand configuration."

The appearance of a hand shape also depends on the camera parameters. To keep our model simple, we assume that hand appearance depends only on the camera viewing direction (two DOFs), and on the camera orientation (up vector, or image plane orientation) that defines the direction from the center of the image to the top of the image (one DOF). We use the terms "camera parameters," "external parameters," and "viewing parameters" synonymously to denote the three-dimensional vector describing viewing direction and camera orientation.

Given a hand configuration vector $C = (c_1, ..., c_{20})$ and a viewing parameter vector $V = (v_1, v_2, v_3)$, we define the hand pose vector P to be the 23-dimensional concatenation of C and V: $P = (c_1, ..., c_{20}, v_1, v_2, v_3)$.

Using these definitions, the generic framework that we propose for hand pose estimation is the following:

1. Preprocessing step: create a database containing a uniform sampling of all possible views of the hand shapes that we want to recognize. Label each view with the hand pose parameters that generated it.

2. For each novel image, retrieve the database views that are the most similar. Use the parameters of the N most similar views as initial estimates for the image.

3. Refine each of the retrieved parameter estimates to optimally match the input.

Our framework allows for systems that return multiple estimates. Multiple estimates can be useful when, either because of deficiencies of the similarity measure, or because of adverse viewing conditions, the retrieval method fails to rank one of the correct matches as the best overall match. If a system returns multiple estimates, we consider the retrieval successful if at least one of those estimates is close to the true parameters of the observed hand. A low value of N may be adequate in domains like 3D hand tracking and sign language recognition, where additional contextual information can be used to discriminate among the returned estimates.

In [1] we speculate on the possibility of using this framework to estimate arbitrary hand shapes, by including a lot of hand shape prototypes in the database, so that for any possible observed shape there is a "close enough" shape in the database. In this paper we tackle an easier version of the problem, by assuming that the observed hand shape is close to one of 26 shape prototypes. We also ignore step 3 of the framework, i.e. the refinement process.

3 Similarity Measures

To retrieve the most similar database views for an input image we combine four different similarity measures: Edge location similarity, edge orientation similarity, finger-based matching and matching based on central and Hu moments. This section describes the individual measures. Section 4 discusses how those measures are combined.

3.1 Chamfer Distance

We define the distance D between a point p and a set of points X to be the Euclidean distance between p and the point in X that is the closest to p:

$$D(p, X) = \min_{x \in X} \|p - x\| \qquad (1)$$

The directed chamfer distance between two sets of points X and Y is defined in [3]. In our system we use the undirected chamfer distance D_c, defined as follows:

$$D_c(X,Y) = \frac{1}{|X|}\sum_{x \in X} D(x,Y) + \frac{1}{|Y|}\sum_{y \in Y} D(y,X) \quad (2)$$

We use D_c to measure the distance between the edge image of the input and the edge image of a database view. Edge images are represented as sets of edge pixels. Before we apply D_c we normalize the scale of both edge images, so that the longest sides of their bounding boxes are equal. The advantage of D_c over the directed chamfer distance is that D_c, in addition to penalizing for points in X that have no close match in Y, it also penalizes for points in Y that have no close match in X. In general, the chamfer distance is relatively robust to small translations, rotations and deformations of the test image with respect to the corresponding model image.

3.2 Edge Orientation Histograms

Given a gray-scale image I, and its corresponding edge image E, we define the orientation R of an edge pixel p to be $R(p) = \arctan \frac{I_y(p)}{I_x(p)}$, where I_x, I_y are the image gradients along the x and y directions. Orientation values are between 0 and 180 degrees. We store those orientation values in an edge orientation histogram with 96 bins, normalized so that the sum of all bin values is 1. We denote the i-th bin of histogram H as $B(H,i)$. If k is the number of bins, and b is the index of one of the bins, we define the cumulative histogram $C(H,b)$ by the formula

$$B(C(H,b),i) = \sum_{j=b}^{b+i} B(H, j \bmod k) \quad (3)$$

As a similarity measure between edge orientation histograms we use the maximum cumulative histogram intersection. The histogram intersection S'_h of histograms H and J is defined in [21] as

$$S'_h(H,J) = \sum_{i=0}^{k-1} \min(B(H,i), B(J,i)) \quad (4)$$

We define the maximum cumulative histogram intersection $S_h(H,J)$ as

$$S_h(H,J) = \max_{0<=b<k} S'_h(C(H,b), C(J,b)) \quad (5)$$

Using cumulative histograms in histogram intersection makes the measure less sensitive to small orientation changes in an image. The reason we don't simply define $S_h(H,J)$ to be $S'_h(C(H,0), C(J,0))$ is that edge orientation histograms are circular; orientation values corresponding to bin 0 are as close to values corresponding to bin 1 as to those corresponding to bin $k-1$. For example, consider the case where $B(H,0) = 1, B(J,k-1) = 1$, and all other bins of H and J are 0. $S'_h(C(H,0), C(J,0))$ would return an inappropriately low similarity value for the two histograms. $S_h(H,J)$ handles this case correctly.

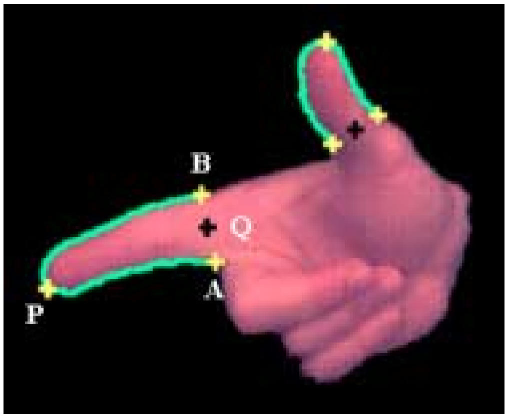

Figure 4: An example output of the finger detector. For the index finger, P is the fingertip, A and B are the boundary endpoints of the finger and Q is the base point. The contour segments AP and PB are shown in green.

3.3 Finger Matching

Given the binary image of a hand, most significant protrusions that we observe are caused by fingers (Figures 2, 3, 5). The fingertips of protruding fingers usually correspond to local maxima in the curvature of the bounding contour of the hand. We represent a protrusion F as the ordered triple (P_F, A_F, B_F) where P_F is the fingertip point, and A_F, B_F are the endpoints of the boundary contour of the protrusion. We define the *base point* Q_F of F as the middle point of the straight line segment between A_F and B_F (Figure 4).

The length of a contour segment is defined to be the number of pixels along that segment. The length $L(F)$ of protrusion F is defined as the minimum of the lengths of the segments $A_F P_F$ and $P_F B_F$. The width $W(F)$ of F is the symmetric Hausdorff distance ([9]) between the contour segments $A_F P_F$ and $P_F B_F$. If $X = A_F P_F$ and $Y = P_F B_F$, then

$$W(F) = \max(\max_{x \in X} D(x,Y), \max_{y \in Y} D(y,X)) \quad (6)$$

using the point-to-set distance D defined in Equation 1. The elongation E of a protrusion F is defined as $E(F) = \frac{L(F)}{W(F)}$.

We have implemented a finger detector that identifies protrusions whose width is less than a threshold T_w and their elongation exceeds a threshold T_e. A detailed description of the finger detector is given in [2].

In defining a distance measure between hand images that uses the results of finger detection, we need to have in mind that a slight change in hand shape or camera viewpoint can cause two nearby fingers to be detected as a single protrusion, or it can cause the elongation of a finger to drop below the detection threshold T_e. Because of that, we generate two sets of fingers for a given hand image I: a set of *definite* fingers S_d^I, detected by setting $T_e = 1.8$, and a set of *potential* fingers S_p^I, detected with $T_e = 1.1$.

We define the distance between fingers F and G to be

$$\|F - G\| = \max(\|P_F - P_G\|, \|Q_F - Q_G\|) \quad (7)$$

The finger-based distance D_f between an input image I and a database view V is defined as

$$D_f(I, V) = \max(\max_{F \in S_d^I}\{D(F, S_p^V)\}, \max_{G \in S_d^V}\{D(G, S_p^I)\}) \quad (8)$$

using the point-to-set distance D defined in Equation 1 and the finger distance defined in Equation 7.

Intuitively, D_f penalizes for any "definite" finger in either image for which there is no nearby "potential" finger in the other image. It does not penalize for any "potential" finger in one image that has no close match in the other image. Before we apply D_f on two hand images, we normalize their scale, as described in the subsection on the chamfer distance.

3.4 Moment-Based Matching

From a hand image I we compute seven central moments and seven Hu moments ([8]), and store them in a 14-dimensional moment vector. We perform Principal Component Analysis ([4]) on the moment vectors of all database views, and we identify the top nine eigenvectors. We define the moment-based distance D_m between an input image I and a database view V to be the Mahalanobis distance ([4]) between their moment vectors, after they have been projected to the eigenspace spanned by the top nine eigenvectors.

4 Hierarchical Retrieval Using Combinations of Measures

In general, we have found that combining different measures we get more accurate results than by using a single measure. The combination of a set of k measures is done as follows: given an input image I, using each of the measures we can rank the database images in order of similarity to I (the most similar view has rank 1). We denote the rank of the i-th synthetic view V_i under measure j as r_{ij}. We define a new combined measure $M(I, V_i)$ as

$$M(I, V_i) = \sum_{j=1}^{k}(w_j \log r_{ij}) \quad (9)$$

where w_j is a preselected weight associated with the j-th measure. Then, we can rank the synthetic views again, using the values of the combined measure. The reason we use in M the *ranks* of a view, as opposed to using the original k measure scores of the view, is that the scores under different measures have different ranges and distributions, and it is not obvious how they should be combined. The rank numbers all belong to the same space and can be easily combined. We sum the logarithms of the ranks, as opposed to the ranks themselves, because this way M behaves more robustly in the frequent cases where a single measure gives a really bad rank to a correct database match.

Weights can be tuned to reflect the accuracy and/or redundancy of the individual measures. Our system picks weights automatically by searching over different combinations and choosing the combination that maximizes accuracy over a small training set of real hand images.

The similarity measures described in section 3 have different strengths and weaknesses. The chamfer distance is the most accurate, but also the most computationally expensive. Moment and finger-based matching, on the other hand, are less accurate, but they can be done almost in real time, if we precompute and save the corresponding features of the database views.

In order for the system to function at more or less interactive speeds, we need a retrieval method that can reject most of the database views very fast, and that applies expensive matching procedures only to a small fraction of likely candidates. Our database retrieval algorithm achieves that using a hierarchical, two-step matching process. First, it ranks the synthetic views by combining finger and moment-based matching, and it rejects the worst ranking views. This initial screening can be done very fast; it takes under a second in our system. Then, we rank the remaining candidates by combining all four measures. In practice, we have found that retrieval accuracy is only slightly affected if we reject 99% of the views in the screening step. In general, the percentage of views that gets rejected in the first step can be tuned to balance between retrieval speed and accuracy.

5 Experiments

Our database contains renderings of 26 hand shape prototypes (Figure 2). The renderings are done using a commercially available hand rendering programming library ([24]). Each shape is rendered from 86 viewpoints, that constitute an approximately uniform sampling of the surface of the viewing sphere. For each viewpoint we generate 48 database views, using different values for image plane orientation, uniformly spaced between 0 and 360 degrees. Generating multiple rotations of the same image is necessary, since the similarity measures that we use are rotation-variant. Overall, the database contains 4128 views per shape prototype, and 107328 views total.

We have tested our system with 276 real images displaying the right hands of four different persons. In those images, the hand is segmented using skin detection ([12]). Eight examples of segmented test images are shown in Figure 5. We manually established pseudo-ground truth for each test image, by labeling it with the corresponding shape prototype and using the rendering software to find the viewing parameters under which the shape prototype looked the most similar to the test image. This way of estimating viewpoint parameters is not very exact; we found that estimates by different people varied by 10-30 degrees. Model views can't be aligned perfectly with a test image, because each individual hand has somewhat different finger and palm widths and lengths, and also because the hand shapes in the real images are only approximations of the 26 shape prototypes.

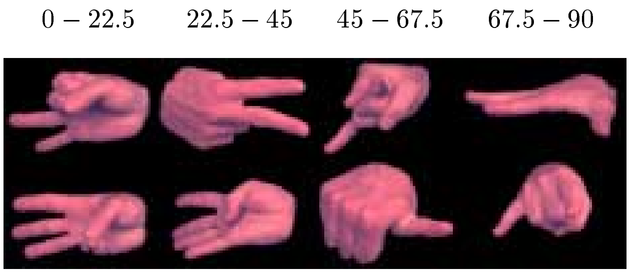

Figure 5: Examples of test images with different frontal angles. The frontal angles of the images in each column belong to the range indicated at the top of that column.

Given the inaccuracy of manual estimates, we consider a database view V to be a *correct match* for a test image I if the shape prototype with which we label I is the one used in generating V, and the manually estimated viewing parameters of I are within 30 degrees of those of V. For any two viewing parameter vectors u and v (see Section 2) there exists a rotation around the center of the viewing sphere that maps u to v. We use the angle of that rotation as the distance between u and v. On average, there are 30.8 correct matches for each test image in the database.

Our measure of the accuracy of the retrieval for a test image is the rank of the *highest-ranking correct match* that was retrieved for that image. 1 is the highest possible rank. Table 1 shows the distribution of the highest ranking correct matches for our test set. We should note that, although the accuracy using the chamfer measure is comparable to the accuracy using the two-step retrieval algorithm, the two-step algorithm is about 100 times faster than simply applying the chamfer distance to each database view.

The viewing parameters for the test images were more or less evenly distributed along the surface of the viewing sphere. We call a hand view ”frontal” if the camera viewing direction is almost perpendicular to the palm of the hand, and we call it a ”side view” if the viewing direction is parallel to the palm. The ”frontal angle” of a view is the angle (between 0 and 90 degrees) between the viewing direction and a line perpendicular to the palm. Figure 5 shows some examples of test images with different frontal angles. Table 2 shows the median rank of the highest-ranking correct matches for test images observed from different frontal angle ranges. As expected, retrieval accuracy is worse for side views, where fewer features are visible. It is fair to mention that, in some of the side views, even humans find it hard to determine what the shape is (see Figure 5).

The weights used to combine the finger and moment-based measures (Equation 9) in the screening step of the retrieval were 0.6 and 0.4. The weights used in the second step of the retrieval were 0.4 for the chamfer and the finger-based measure, and 0.1 for the edge orientation measure and the moment-based measure. These weights were established using a small training set of 28 real images, none of which was included in the test set.

Rank range	Chamfer	Edge hist.	Fingers	Moments	2-step
1	22.8	0.0	7.6	2.5	21.7
1-2	31.9	0.0	11.2	6.5	31.5
1-4	40.9	0.0	18.1	8.6	41.7
1-8	49.6	0.3	26.8	13.4	52.5
1-16	58.3	2.2	34.0	20.3	60.1
1-32	68.8	4.7	43.8	30.1	68.8
1-64	77.5	6.5	50.7	38.0	76.4
1-128	85.9	11.2	56.2	47.5	83.7
1-256	92.0	23.6	68.5	58.0	87.3
256-	8.0	76.4	31.5	42.0	12.7

Table 1: Retrieval accuracy: for each rank range and each measure we indicate the percentage of test images for which the rank of the highest ranking correct match was in the given range. 2-step stands for the two-step retrieval algorithm described in Section 4.

Frontal angle	0-22.5	22.5-45	45-67.5	67.5-90
# of images	54	72	86	64
Median	1	3	9	47

Table 2: Accuracy of the two-step retrieval algorithm over different frontal angles. For each range of frontal angles we indicate the number of test images whose frontal angles are in that range and the median of the highest ranking correct matches for those images.

Retrieval times were between 3 and 4 seconds on a PC with a 1.2GHz Athlon processor. The memory requirements of the system were under 100MB.

6 Future Work

Our long term goal is a system that can provide reliable estimates for arbitrary hand shapes, seen from arbitrary viewpoints, at speeds that allow for interactive applications. In order to do that, we need to include more shape prototypes in the database, and implement the refinement step of the framework presented in Section 2. At the same time we need to work on improving retrieval accuracy. We plan to investigate ways of extracting more information from the input image, using more elaborate bottom-up processing. We are currently looking into methods of detecting fingers and fingertips in the interior of the hand.

As the size of the database grows larger, the issue of retrieval efficiency will become critical. It may take under a second to apply finger and moment-based matching to a database of 100,000 images, but the time may become prohibitive if we use a significantly larger set of hand shape prototypes and the number of views grows into the millions or tens of millions. We need to investigate ways of building index tables, that can automatically focus the search on smaller parts of the database. We are currently developing an indexing scheme that can direct the database search using the locations of detected fingers. Index tables may prove to be feasible even for measures like the chamfer distance or the related Hausdorff distance. [11] describes how to embed the Hausdorff distance into an L_∞ metric and [10] discusses efficient methods for answering approximate nearest neighbor queries in L_∞ spaces.

Another system aspect that we have neglected so far is hand segmentation. In our test images the hand was segmented using skin detection ([12]), but those images were captured using a background that made segmentation relatively easy. It is important to evaluate the performance of our similarity measures under realistic segmentation scenarios and especially in the presence of segmentation errors. As a start, we plan to use our system as the basis for a real-time desktop human computer interface, where the hand is segmented using skin color and motion.

7 Conclusions

We have presented a general framework for 3D hand pose classification from a single image, observed from an arbitrary viewpoint, using appearance-based matching with a database of synthetic views. Using the ground truth labeling of the retrieved images the system can also estimate camera viewing parameters. We use a hierarchical retrieval algorithm, which combines the efficiency of computationally cheap similarity measures with the increased accuracy of more expensive measures, and runs at close to interactive speed.

Acknowledgments

This research was supported in part by the National Science Foundation, under grants IIS-9912573 and EIA-9809340. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] V. Athitsos and S. Sclaroff. 3D hand pose estimation by finding appearance-based matches in a large database of training views. In *IEEE Workshop on Cues in Communication*, 2001.

[2] V. Athitsos and S. Sclaroff. An appearance-based framework for 3D hand shape classification and camera viewpoint estimation. Technical Report 22, Computer Science Department, Boston University, 2001.

[3] H.G. Barrow, J.M. Tenenbaum, R.C. Bolles, and H.C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *IJCAI*, pages 659–663, 1977.

[4] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley-Interscience, 2001.

[5] W.T. Freeman and M. Roth. Computer vision for computer games. In *Automatic Face and Gesture Recognition*, pages 100–105, 1996.

[6] R. Hamdan, F. Heitz, and L. Thoraval. Gesture localization and recognition using probabilistic visual learning. In *CVPR*, volume 2, pages 98–103, 1999.

[7] T. Heap and D. Hogg. Towards 3D hand tracking using a deformable model. In *Face and Gesture Recognition*, pages 140–145, 1996.

[8] MK Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, IT-8:179–187, 1962.

[9] D. Huttenlocher, D. Klanderman, and A. Rucklige. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, September 1993.

[10] P. Indyk. On approximate nearest neighbors in non-euclidean spaces. In *FOCS*, pages 148–155, 1998.

[11] P. Indyk. *High-dimensional Computational Geometry*. PhD thesis, MIT, 2000.

[12] M.J. Jones and J.M. Rehg. Statistical color models with application to skin detection. In *CVPR*, pages I:274–280, 1999.

[13] M. Kohler. Special topics of gesture recognition applied in intelligent home environments. In *Proceedings of the Gesture Workshop*, pages 285–296, 1997.

[14] R. Liang and M. Ouhyoung. A real-time continuous gesture recognition system for sign language. In *Face and Gesture Recognition*, pages 558–567, 1998.

[15] J. Ma, W. Gao, and C. Wang J. Wu. A continuous chinese sign language recognition system. In *Face and Gesture Recognition*, pages 428–433, 2000.

[16] B. Moghaddam and A. Pentland. Probabilistic visual learning for object detection. Technical Report 326, MIT, June 1995.

[17] J.M. Rehg. *Visual Analysis of High DOF Articulated Objects with Application to Hand Tracking*. PhD thesis, Electrical and Computer Eng., Carnegie Mellon University, 1995.

[18] R. Rosales, V. Athitsos, L. Sigal, and S. Sclaroff. 3D hand pose reconstruction using specialized mappings. In *ICCV*, volume 1, pages 378–385, 2001.

[19] H. Sagawa and M. Takeuchi. A method for recognizing a sequence of sign language words represented in a japanese sign language sentence. In *Face and Gesture Recognition*, pages 434–439, 2000.

[20] N. Shimada, K. Kimura, and Y. Shirai. Real-time 3-D hand posture estimation based on 2-D appearance retrieval using monocular camera. In *Recognition, Analysis and Tracking of Faces and Gestures in Realtime Systems*, pages 23–30, 2001.

[21] M.J. Swain and D.H. Ballard. Color indexing. *IJCV*, 7(1):11–32, 1991.

[22] Jochen Triesch and Christoph von der Malsburg. Robotic gesture recognition. In *Gesture Workshop*, pages 233–244, 1997.

[23] A. Utsumi and J. Ohya. Multiple-hand-gesture tracking using multiple cameras. In *CVPR*, volume 1, pages 473–478, 1999.

[24] Virtual Technologies, Inc., Palo Alto, CA. *VirtualHand Software Library Reference Manual*, August 1998.

[25] Y. Wu and T.S. Huang. View-independent recognition of hand postures. In *CVPR*, volume 2, pages 88–94, 2000.

[26] Y. Wu, J.Y. Lin, and T.S. Huang. Capturing natural hand articulation. In *ICCV*, volume 2, pages 426–432, 2001.