Technical Report BU-CS-TR-2001-021

A shorter version of this paper is published in the proceedings of

IEEE Workshop on Cues in Communication, 2001

3D Hand Pose Estimation by Finding Appearance-Based Matches in a Large Database of Training Views

Vassilis Athitsos and Stan Sclaroff

Computer Science Department
Boston University
111 Cummington Street
Boston, MA 02215
e-mail {athitsos,sclaroff}@cs.bu.edu

Abstract

Ongoing work towards appearance-based 3D hand pose estimation from a single image is presented. A large database of synthetic hand views is generated using a 3D hand model and computer graphics. The views display different hand shapes as seen from arbitrary viewpoints. Each synthetic view is automatically labeled with parameters describing its hand shape and viewing parameters. Given an input image, the system retrieves the most similar database views, and uses the shape and viewing parameters of those views as candidate estimates for the parameters of the input image. Preliminary results are presented, in which appearance-based similarity is defined in terms of the chamfer distance between edge images.

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. Appearance-based methods are well-suited to that task [13, 6, 2]. 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 [7, 8, 11]. Computer vision systems that estimate 3D hand pose do it only in the context of tracking [9, 3, 15, 12] . In that context, the pose can be estimated at the current frame as long as the system knows the pose in the previous frame. The limitation of tracking methods is that they do not address the issue of estimating 3D pose when information about the previous frame is not available. Because of that limitation, current 3D hand trackers need to be manually initialized and they cannot recover when they lose the track. Developing methods to estimate 3D hand pose from a single frame can lead to hand trackers that are fully automatic and can recover from mistakes, and that is one of the goals of our approach.

A real-time system that tracks 3D hand pose is presented in [12]. That system uses a database of synthetic views and an appearance-based method to retrieve the closest matches to the observed input. To achieve the retrieval time and accuracy that are required for a real-time system, that method only considers database views that correspond to poses near the estimated hand parameters of the previous frame.

In [10], the system aims to recover hand pose from a single image using a machine learning approach, the specialized mappings architecture (SMA). The relationship between hand images and hand poses is one-to-many, meaning that a given hand image could come from different hand poses. In the SMA framework this issue is addressed by learning tens of different functions, that map feature vectors to hand poses. Intuitively, each learned function is a "specialist" that works well if the underlying pose is in a specific region of the output space (i.e. the 3D pose space). Given an input image, we obtain the estimates of all functions. Using computer graphics a synthetic hand image is generated based on each estimate, and the similarity between that image and the input image can be used to choose the best estimate.

One weakness of that system is that it requires its input space to be relatively low-dimensional (with fewer than 20 dimensions). It is still an open question whether such a low-dimensional representation of hand images exists, that carries enough information to identify 3D hand pose. In order to satisfy the low-dimensionality requirement, the input space of the specialized functions is the seven Hu moments of the input image. It can be

seen in the experimental results for that system that representing the input image using Hu moments discards a lot of discriminative information that could help find the correct pose.

In this paper we discuss preliminary work towards estimating 3D hand pose from a single image by matching the image with a large database of training views. Our long term goal is to evaluate the accuracy and efficiency that can be achieved using such a framework. As we work towards implementing this framework we have to address the following issues:

- What are the storage requirements for an adequate database of training views?
- What are good similarity measures, that can identify the few correct matches and reject the multitudes of possible false matches?
- How can the matching be done efficiently, to satisfy the requirements of a real-time system?

Our work so far provides some insights on the first two questions. We discuss issues related to obtaining a large database of images for which ground truth is known. We examine some pros and cons of using synthetic views for the database. We present some preliminary experimental results, in which our database contains more than 100,000 images, generated from 26 hand shapes, rendered using many different combinations of camera viewpoint and camera orientation. Matching is done using the chamfer distance[1].

In our discussion we assume that we can accurately segment the hand in all images. Segmentation is trivial in our synthetic images, where we control how the hand is displayed. In the real images used in our experiments, the system locates and segments the hand using skin color detection [5]. As we work towards improving our system, we will have to address the issue of segmentation more thoroughly, to ensure that our proposed approach can work in real-life environments, where segmentation may not be as easy as in our controlled experiments. Good segmentation results have been reported in the literature, in domains where certain constraints are satisfied. Constraints often employed include known and/or static background, and the hands being the fastest moving objects in the scene. Such constraints may be appropriate in many human-computer interaction applications.

2 Proposed Framework

We model the hand as an articulated object, consisting of 16 links: the palm and 15 links corresponding to finger 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. Therefore, a total of 20 DOFs describes completely all degrees of freedom in the joint angles. For the 20-dimensional vector containing those 20 DOFs we use synonymously the terms “internal hand parameters,” “hand shape” and “hand configuration.”

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.

The appearance of a hand shape also depends on the camera parameters. To keep our model simple, we assume orthographic projection and we require that the size of the hand is fixed. Given those assumptions, hand appearance depends on the viewing direction (two DOFs), and on the camera orientation (up vector) 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. Figure 2 illustrates the definition of camera parameters.

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:

- Preprocessing step: create a database containing a uniform and sufficiently *dense sampling* of all possible views of all possible hand configurations. Label each view with the hand pose parameters that generated it.

- For each novel image, find the database views that are the most similar. Use the parameters of those views as candidate estimates for the image.

When do we consider a set X of views to be a *dense sampling*? Let's fix a similarity measure M among images, a distance in pose parameter space D_p, and numbers d, n, t (based on an application's specific needs). Given an input image A, we find the n views in X that are the most similar to A, using measure M. If at least one of those n views has a ground truth vector that is within distance d of A's vector, we consider that A was successfully matched. We define that a set X of views is a dense sampling if, for any image

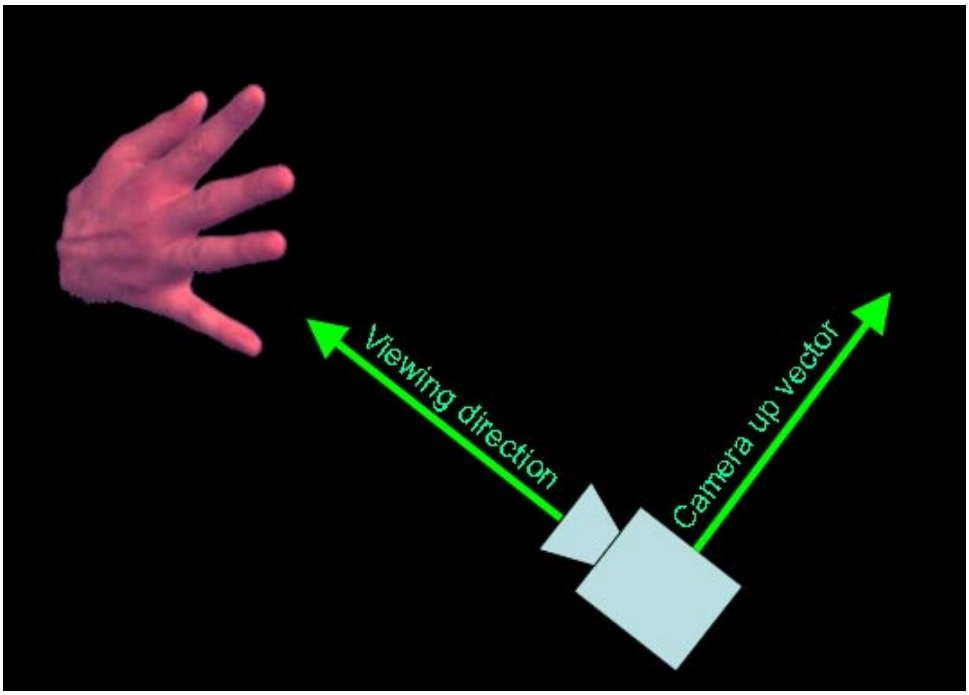

Figure 2: The viewing direction is the line connecting the camera's center of projection to the center of the hand. The camera up vector defines the rotation of the camera, which specifies the upward direction on the image plane. The camera "up" vector is constrained to be on the plane that is perpendicular to the viewing direction.

A given as input to the application, the probability that A will be successfully matched is greater than t.

As a simple example, consider a system that would be used to automatically initialize a 3D hand tracker by estimating the hand parameters of the first input frame. Suppose that the tracker managed to start tracking successfully as long as it was given 10 initial estimates, which included at least one within some distance x of the true hand configuration, using the hand pose distance D_p. If we considered it acceptable for the tracker to initialize correctly 95% of the times, then we would set $d = x$, $n = 10$, and $t = 0.95$.

3 Space Complexity

The space complexity of estimating hand pose in our framework depends on the number of database images that we need for a dense sampling. We need to answer two questions: first, how many different views do we need for each configuration? Second, how many hand configurations should we include? At this point we do not have precise answers to either question, but it is illustrative to look at different possible answers.

As far as the number of views is concerned, [10] rendered each hand shape from 86

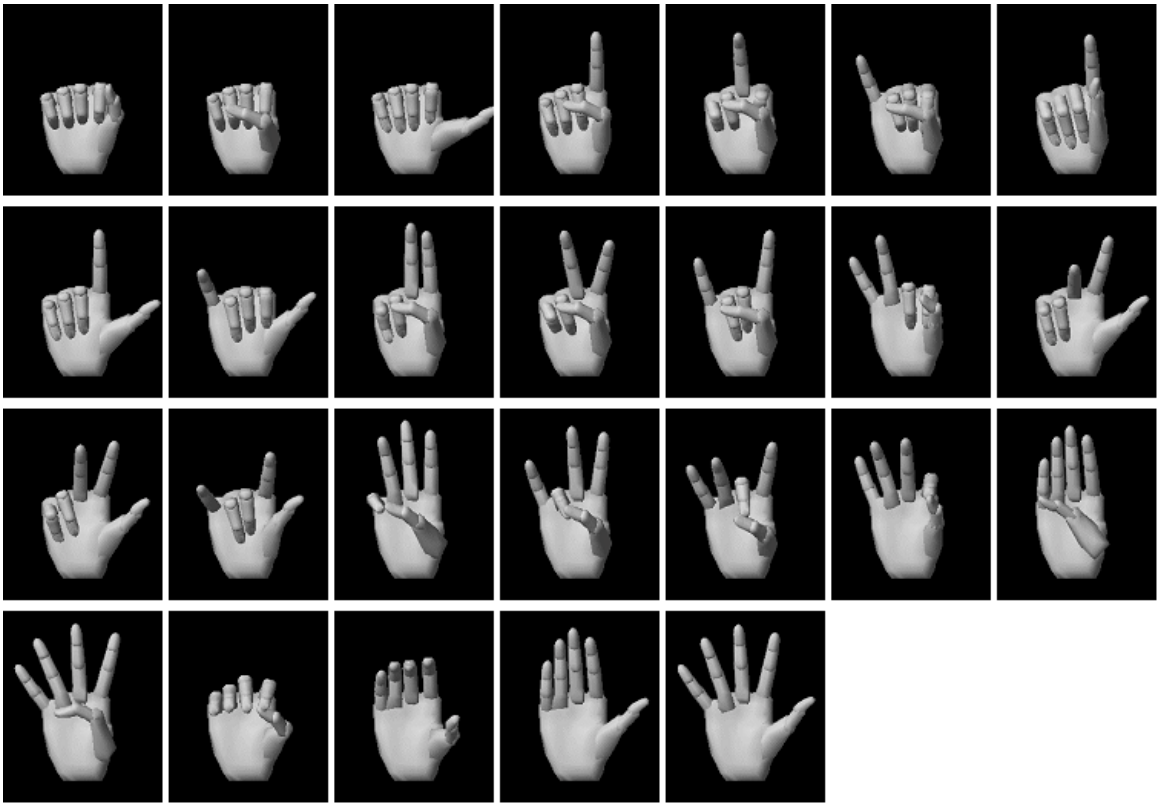

Figure 3: The 26 basic shapes used to generate training views in our database. These images were synthetically generated.

viewpoints, and [12] used 128 viewpoints. However, [10] and [12] used features that they considered to be invariant under image rotation. If we use rotation-variant features, then for each viewpoint we must generate many views corresponding to different orientations of the camera's up vector. In our experiments we have used 86 viewpoints and generated 48 images for each viewpoint, giving us a total of 4128 images for each hand shape. However, it is conceivable that a significantly lower or higher number would offer a more desirable balance between efficiency and accuracy.

How many hand configurations should we generate? In our model, a hand configuration has 20 degrees of freedom. Grid sampling in such a high dimensional space would be infeasible. Fortunately (at least for our research goals) there is a high degree of correlation in the way those degrees of freedom vary, due to anatomical and behavioral constraints. [10] and [15] applied PCA to hand shapes captured using data gloves, to reduce the dimensionality to eight and seven dimensions respectively. However, even with seven dimensions, discretizing each dimension to, say, five values, would yield about 80,000 configurations, whereas discretizing to 10 values would yield 100 million configurations. We can use the latter figure as an upper bound for our estimate.

In [15] the system uses 28 basic hand configurations and expects each observed shape to belong to the linear manifolds spanned by any two of the basis configurations. If that assumption is valid, we have 378 pairs of basic shapes, and if we sample between one and 20 points in each line segment connecting two basic shapes we get roughly between 400 and 8000 configurations. We can consider the number 400 to be a rough lower bound on the number of hand shapes that we should include in our database.

To sum up, it seems that we need between 400 and 100,000,000 shapes. Obviously more research and experiments are needed to narrow this range. Systems, including our own, have used from about 100 to about 4000 views per shape, but smaller or larger values for that number may turn out to be preferable. So, the number of images required for a dense sampling of the hand configuration space can be in the tens of thousands at the lower end or in the trillions at the higher end. Note that, for each view, we only need to save the feature vector that we will use for matching. If the total number of views is less than a million, we may be able to save the views themselves. However, if it is in the range of trillions, then even saving a low-dimensional feature vector may require a prohibitive amount of storage, at least given the current state of the art in storage devices.

The time complexity of the problem must also be addressed, if we want to implement a real-life system. In general, given an input image, we can find its nearest neighbor under a given similarity measure by comparing the image with every database view. Of course, this kind of exhaustive search may be prohibitively slow for a large database, and that can be seen in our experimental results. However, indexing strategies may allow finding approximate nearest neighbors in time sublinear to the size of the database. Our current system focuses on space complexity and matching accuracy issues, but we briefly discuss time complexity in the future work section (Section 7).

4 Synthetic Versus Real Training Data

In any supervised learning method, one has to provide ground truth for the training data. Manual labeling can be prohibitive when the size of the training set reaches the order of hundreds of thousands. A big advantage of synthetic training sets is that the labeling of the data can be done automatically; when the system generates a hand view, it knows exactly what pose and viewing parameters it used for that view, and it can use the vector of those parameters as a label for the view. This allows researchers to use training sets that are orders of magnitude larger than what would be feasible with manual labeling.

In our particular domain, a real training set, containing views of thousands of hand shapes using hundreds of different viewing parameters for each shape, would also be very hard to collect. For one thing, it would require tedious human effort to generate samples of the appropriate shapes. It would also require sophisticated multicamera setups that, at present, are definitely not commodity items, even for research institutions. Rendering synthetic models circumvents these problems and makes the generation of large training sets relatively effortless. The only limiting factors remaining are the time to generate the

Figure 4: Three different views of the same basic shape. These images were synthetically generated.

training views and the space to store them.

We should note that using synthetic training views can also have disadvantages. The model and the rendering algorithm may ignore certain aspects of the appearance of real-life objects (see Figures 3, 4). The hand model that we use, for example, does not model hand texture; therefore, it cannot be used for a task like learning the appearance of palm wrinkles as the fingers bend. Although we have only used synthetic training so far, we may well decide to use real images for some aspects of the training in the future.

5 Edge-Based View Matching

An important issue in any appearance-based retrieval system is the choice of similarity measure. A yet-to-be-discovered "perfect" measure would find all correct database matches to be more similar to the input image than all incorrect database matches, even under significant amounts of noise and occlusion. At the same time we would like the similarity measure to be reasonably efficient to compute.

In our current implementation we have defined image similarity using the chamfer distance ([1]) between edge images. Given an input image, we extract its edge pixels using an edge detector and store the coordinates of those pixels in a set X. As a preprocessing step, the system has already extracted the edge images of the database views and stored the coordinates of the edge pixels of hand image i into the set Y_i. The chamfer distance can assign a distance between X and each Y_i.

To define the chamfer distance, we first need a notion of distance between a point and a set of points. 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:

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

The directed chamfer distance between two sets of points X and Y is defined in [1]. In our system we use the undirected chamfer distance D_c, defined as follows:

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

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.

We use edges as image features, because edges offer some insensitivity to lighting conditions while still carrying enough information to estimate the pose. We make the latter claim after having looked at numerous edge images ourselves, and having verified that we could easily tell the underlying hand pose in most of the cases. Figure 5 includes examples of edge images.

6 Experimental Results

In Section 2 we proposed a framework for estimating hand pose from a single view by finding matches in a large database of views. We have built an experimental system in order to evaluate the feasibility of this framework. Our goal was to see what kinds of performance we could get using a fairly large set of training views and a simple similarity measure.

We generated synthetic data using a commercially available hand model [14]. A hand shape in this model has 20 degrees of freedom, that specify the joint angles between neighboring links. We used 26 different hand shapes. Each shape was rendered from 86 viewing directions, sampled uniformly from the surface of the 3D view sphere. From each viewing direction we obtained 48 images, each corresponding to a different image plane rotation. We normalized each view for scale, enforcing that the maximum distance between any two contour points be 192 pixels. Overall, we generated 107328 training views. Figure 3 shows the 26 hand shapes and Figure 4 shows a hand shape rendered using different viewing parameters.

For our synthetic views, edge extraction can be done in a noise-free way. Each pixel is labeled with the link that it belongs to. A border between different links is considered to be an edge, unless it is identified with a joint connecting neigbboring links. In our input images such borders that correspond to joints do not give rise to particularly strong edges.

Real images used for testing are preprocessed by segmenting the hand using skin detection [5], and by normalizing the scale of the segmented hand. Edges are extracted using a Canny detector, implemented in Intel's OpenCV programming library [4].

We tested our system with 28 real images. We established ground truth for our test images as follows: for each input image A, we manually identified the training view B that was the most similar to it. We considered a training view C to be a correct match for A if it came from the same hand shape as B and its viewpoint parameters were close to the viewpoint parameters of B (within 30 degrees, in the L_∞ distance). Using this definition, for each of our test images there existed about 40 correct matches on average, out of a total of 107328 possible matches.

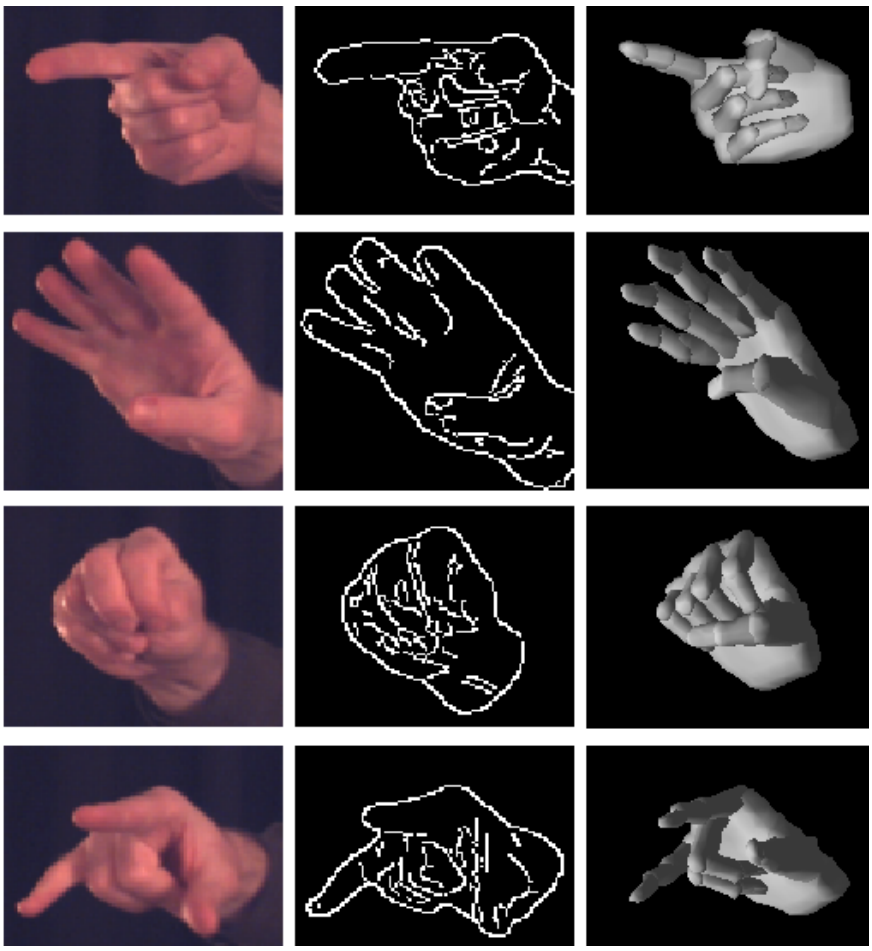

Figure 5: First column: Input images, for which correct matches were found in the top ten matches. Second column: The result of the Canny edge detector on those images (due to downsampling, the quality of these reproductions is not great). Third column: The correct matches that were found for the input images.

Rank	Manual edges	Canny edges
1-10	12	13
11-100	9	6
101-1000	6	6
1001-	1	3

Table 1: Experimental results. For each rank range and each edge extraction mechanism, we indicate the number of test images for which the highest ranking correct match had a rank in the given range.

We also wanted to evaluate the extent to which incorrect matches were caused by inaccuracies in the Canny detector output, and the extent to which they were caused by using a similarity measure based on the chamfer distance. Therefore, we generated a second set of edge images for our test set, in which we manually identified edges.

The results are shown in Table 1. Some cases where correct database views were found in the top ten matches are shown in Figure 5, whereas some cases where there were no correct views in the top thousand matches are shown in Figure 6.

We were actually surprised to find that, for almost half our input images, correct matches were found in the top ten matches. We consider that result very encouraging, given the fact that we used off-the-shelf feature extraction and similarity measure modules. At the same time, the results with the manually-built edge images demonstrate that even perfect edge detection will not yield much better results if we keep our current similarity measure. If we want to continue using edge-based features, a more sophisticated similarity measure is necessary to improve performance. We discuss this issue in Section 7.

Our experiments focused on the issues of space complexity and feature extraction. We did not address issues of time complexity. No particular optimizations were made and each test image was compared with all training images. That process took about 25 minutes per test image, using a not particularly optimized C++ implementation on a PC with a 1GHz Pentium III processor and 1GB RAM. Ongoing work focuses on time complexity as well, as described in the next section.

7 Ongoing and Future Work

The chamfer distance gives equal importance to every edge pixel, and treats it as an isolated entity. There is a range of features that are formed by groups of edge pixels (e.g., straight lines, “finger-like” protrusions, “finger-like” edge groups), some of each are easy to detect. This is illustrated by Figure 7, which shows some results of a very simple finger detector that we developed. That detector considers a contour protrusion to be a finger if its length/width ratio exceeds a given threshold. We are currently working on contour and edge-based ways to detect fingers. Explicit information about fingers can be used to build more discriminating similarity measures and reduce the number of false matches. One can

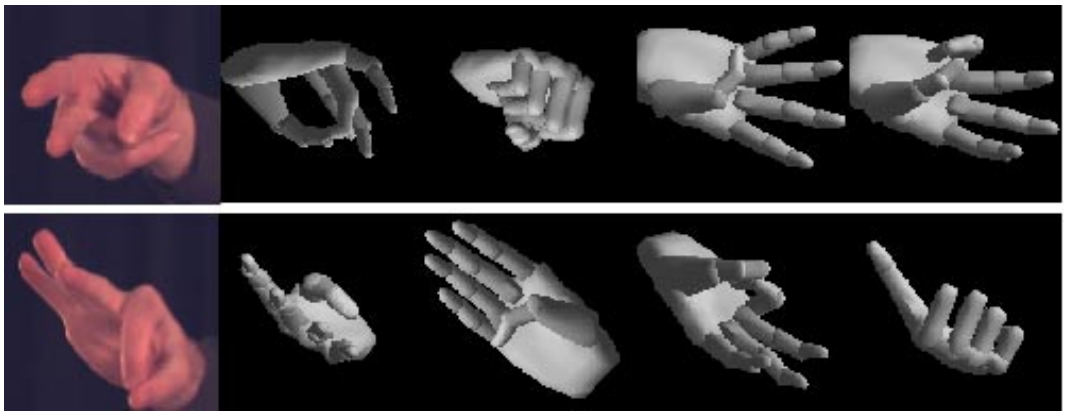

Figure 6: Test images, for which the highest ranking correct match had a rank greater than 1000. First column: Original images: Rest of the columns: False matches for those images, ranking in the top 10.

easily see, for example, that several false matches shown in Figure 6 display easily detected finger protrusions. A similarity measure that would take into account the length, position and orientation of protrusions should be able to eliminate such false matches.

Furthermore, features like finger parameters may be useful in building index tables for our training views, that would guide the search towards the most likely candidates. Fast rejection of many candidates may also be feasible using features like Hu moments, central moments, edge orientation histograms, or the shape feature described in [12]. Matching based on such features is orders of magnitude faster than calculating chamfer distances. What we plan to determine experimentally in the short term is the extent to which such features can be used to eliminate false matches while preserving the correct ones.

It is also important to take a closer look at the issue of the necessary number of training views. As discussed in Section 3, the right answer for a general purpose system is somewhere between some tens of thousands and some trillions of views, using between 400 and 100 million hand shapes. We need to find ways to estimate those ranges in a more educated way and find more specific answers. In the short term we plan to extend our training set to include hundreds of different hand shapes. We do believe that, even if it turns out that a few hundred shapes are too few for a general purpose system, systems trained on that many shapes can still find applications in several hand pose estimation and gesture recognition domains.

If we manage to develop our approach so that it can handle hundreds of different shapes and it can retrieve the closest database matches efficiently, then it would be interesting to see how well this approach would integrate with a 3D hand pose tracker. Our system could be used as a front end, estimating the hand pose in the first frame, and re-estimating the hand pose when the track was lost.

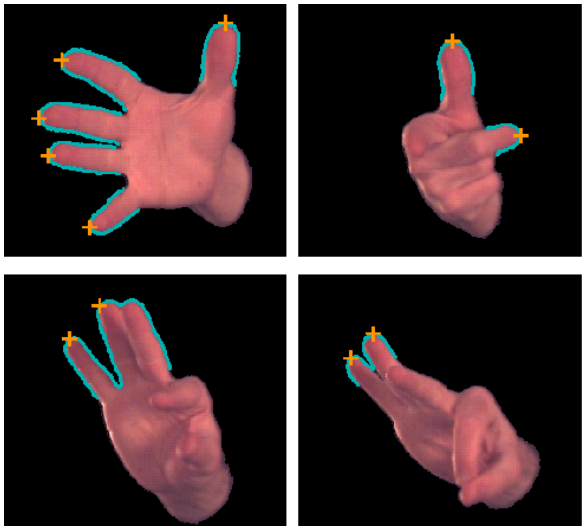

Figure 7: A simple finger detector: Contour protrusions whose length/width ratio exceeds a threshold are considered to be fingers. The blue line shows the detected extent of the finger and the orange cross shows the detected fingertip.

8 Conclusions

We have suggested a general framework for 3D hand pose estimation from a single image, using appearance-based matching with a database of synthetic views. The use of synthetic images lets us obtain very large training sets, with ground truth information. Our initial feasibility study shows that this is a promising approach. In our experimental results, for almost half of the test images the system retrieved correct views in the top ten matches, despite the large size of our database and the use of fairly simple feature extraction and matching mechanisms. We describe our ongoing and future work to improve performance, by using even more training data, performing more elaborate bottom-up processing of hand images and building index tables to speed up the search.

Acknowledgements

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] 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.

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

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

[4] Intel Corporation. *Open Source Computer Vision Library Reference Manual*, December 2000.

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

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

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

[8] 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.

[9] 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.

[10] Romer Rosales, Vassilis Athitsos, Leonid Sigal, and Stan Sclaroff. 3D hand pose reconstruction using specialized mappings. In *ICCV*, volume 1, pages 378–385, 2001.

[11] 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.

[12] Nobutaka Shimada, Kousuke Kimura, and Yoshiaki 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.

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

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

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