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

Abstract

Ongoing work towards appearance-based 3D hand pose estimation from a single image is presented. Using a 3D hand model and computer graphics a large database of synthetic views is generated. 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 [15, 7, 3]. 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 [8, 9, 13]. Computer vision systems that estimate 3D hand pose do it only in the context of tracking [11, 4, 17, 14] . 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.

A system aiming to recover hand pose from a single image, using a machine learning approach, is described in [12]. Tens of different functions are learned, that map feature vectors to hand poses. For each input image, the results of all functions are rendered using computer graphics. The result that is chosen is the one whose rendering is the most similar to the input image. That system has been primarily tested with synthetic data. In addition, it uses Hu moments as feature vectors, and as a consequence it cannot differentiate between different classes of shapes that have similar boundary contours. Our system tackles that problem by using edges as features, since edges can give us information about both the boundary and the interior of a hand.

The system presented in [14] uses, like our system, a database of synthetic views and an appearance-based method to find the closest match to the observed input. However, that method is applied in the context of tracking, so that at each frame it only needs to consider database views whose parameters are close to the estimated hand parameters of the previous frame. In addition, that system also relies exclusively on features of the bounding contour, and therefore has the same problem as [12].

In this paper we are take a closer look at the problem of estimating 3D hand pose from a single frame. 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, consider the implications of those results, and put forward some suggestions on how to build future systems that come closer to achieving reliable and efficient performance.

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

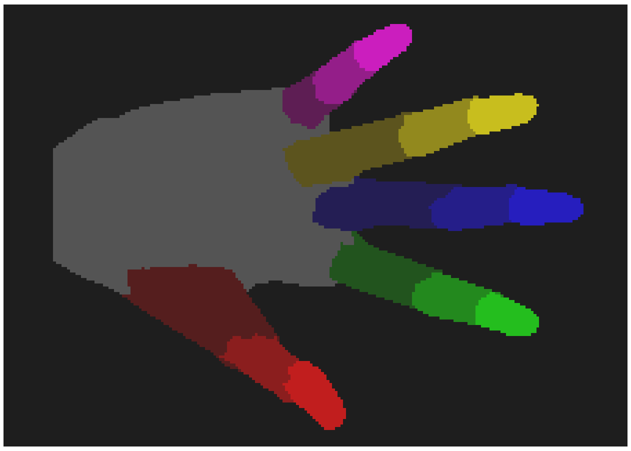

Figure 1: The hand as an articulated object. The palm and each finger are shown in a different color. The three different links of each finger are shown using different intensities of the same color.

is displayed. Although hand segmentation is a very hard problem in real images, good results can be achieved in specific domains by imposing certain constraints, like requiring that the background is known and/or static, that no skin-colored objects are visible except for the hands, or that no objects move faster than the hands [10, 7, 2]. In the real images that we use in our experiments, the system locates and segments the hand using skin color detection [6].

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 the set of all angles between neighboring links. 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 orthographic projection. We also require that the size of the hand is fixed, and that the image plane is perpendicular to the line connecting the center of projection to the center of the hand. 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.

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 among images, a distance in pose parameter space, 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. 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 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 (given some definition of distance in the hand pose space). 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 Complexity Issues

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, [12] rendered each hand shape from 86 viewpoints, and [14] used 128 viewpoints. However, [12] and [14] 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. [12] and [17] 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 80000 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 [17] 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 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. Our current system focuses on space complexity and matching issues, but we discuss issues related to time complexity in 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. This allows researchers to use training sets that are orders of magnitudes 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 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 2, 3). 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

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 [16]. 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. Over-

Figure 2: Three of the 26 basic shapes used to generate training views in our database.

Figure 3: Three different views of the same basic shape.

all, we generated 107328 training views. Figure 2 shows some of the hand shapes and Figure 3 shows a hand shape rendered using different viewing parameters.

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 4 includes examples of edge images.

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 [6], and by normalizing the scale of the segmented hand. Edges are extracted using a Canny detector, implemented in Intel's OpenCV programming library [5].

We use the chamfer distance [1] to estimate the similarity between two edge images.

6 Experimental Results

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.

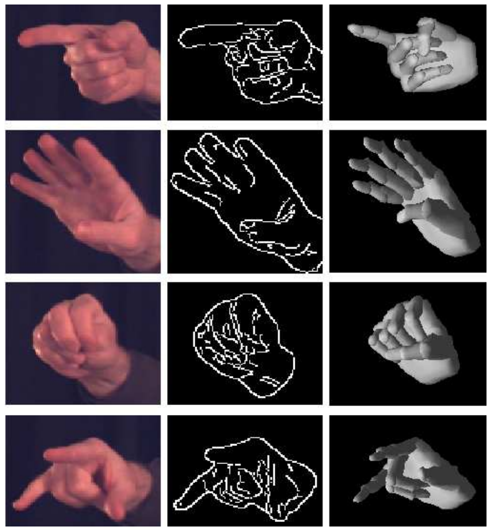

Figure 4: First column: Input images, for which correct matches were found in the top ten matches. Second column: The result of the Canny edge detector on those images (due to downsampling, the quality of these reproductions is not great). Third column: The correct matches that were found for the input images.

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 4, whereas some cases where there were no correct views in the top thousand matches are shown in Figure 5.

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

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.

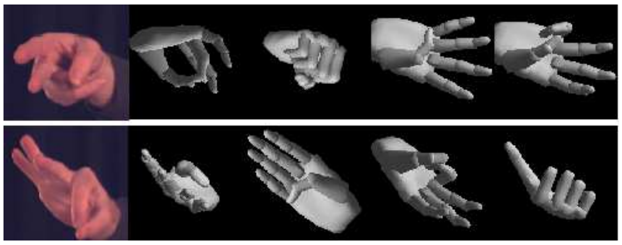

Figure 5: Test images, for which the highest ranking correct match had a rank greater than 1000. First column: Original images: Rest of the columns: False matches for those images, ranking in the top 10.

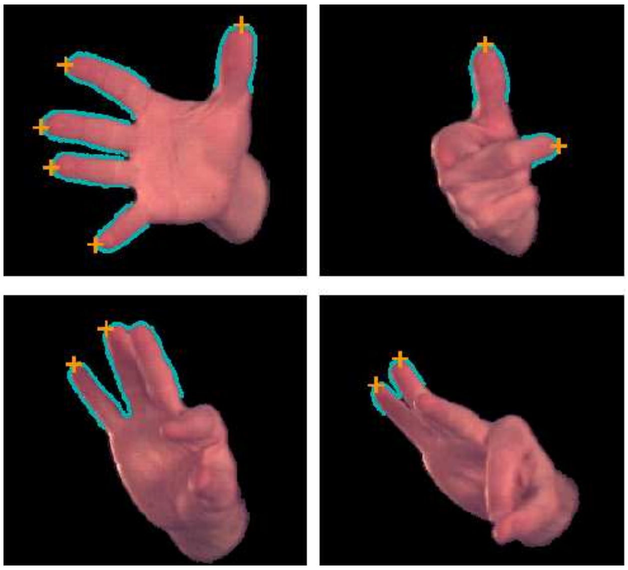

Figure 6: A simple finger detector: Contour protrusions whose length/width ratio exceeds a threshold are considered to be fingers. The blue line shows the detected extent of the finger and the orange cross shows the detected fingertip.

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 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 6, 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 easily see, for example, that several false matches shown in Figure 5 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 [14]. 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.

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.

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. Cutler and M. Turk. View-based interpretation of real-time optical flow for gesture recognition. In *Face and Gesture Recognition*, pages 416–421, 1998.

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

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

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

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

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

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

[9] 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.

[10] J. Martin, V. Devin, , and J.L. Crowley. Active hand tracking. In *Face and Gesture Recognition*, pages 573–578, 1998.

[11] 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.

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

[13] 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.

[14] 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.

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

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

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