Boston University Computer Science Tech. Report No. 2003-009, April 1, 2003.
To appear in Proc. IEEE CVPR, June 2003.

Estimating 3D Hand Pose from a Cluttered Image

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

Abstract

A method is proposed that can generate a ranked list of plausible three-dimensional hand configurations that best match an input image. Hand pose estimation is formulated as an image database indexing problem, where the closest matches for an input hand image are retrieved from a large database of synthetic hand images. In contrast to previous approaches, the system can function in the presence of clutter, thanks to two novel clutter-tolerant indexing methods. First, a computationally efficient approximation of the image-to-model chamfer distance is obtained by embedding binary edge images into a high-dimensional Euclidean space. Second, a general-purpose, probabilistic line matching method identifies those line segment correspondences between model and input images that are the least likely to have occurred by chance. The performance of this clutter-tolerant approach is demonstrated in quantitative experiments with hundreds of real hand images.

1. Introduction

Techniques that allow computers to estimate the 3D pose 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, and non-intrusive motion capture systems.

Our system provides estimates of 3D hand pose from a single cluttered image. In our approach, hand pose estimation is formulated as an image database indexing problem. The closest matches for an input hand image are retrieved from a large database of synthetic hand images. The ground truth labels of the retrieved matches are used as hand pose estimates for the input. A previous version of our system was presented in [1]. One limitation of that method was that it required very clean hand segmentation. In this paper we extend our method so that it can work under more difficult segmentation conditions. The system now only requires that a bounding box of roughly the right location and size has been placed around the hand. The bounding box is allowed to include arbitrary amounts of clutter in addition to the hand region. Examples of input images are shown in Figure 1.

Improved performance under clutter is achieved by using two novel similarity measures. In Section 4 we introduce an indexing method, based on the chamfer distance, that can be used to quickly eliminate most database candidate matches. In Section 5 we present a line matching method that also improves retrieval accuracy for cluttered images. Our method combines geometric and *saliency* criteria in a probabilistic way, and identifies line matches that are the *least likely* to have occurred by chance.

2. Related Work

Computer vision systems that estimate 3D hand pose typically do it in the context of tracking [8, 16, 19, 20, 24]. In that context, the pose can be estimated at the current frame as long as the system knows the pose at the previous frame. Since such trackers rely on knowledge about the previous frame, they need to be manually initialized, and cannot recover when they lose the track.

Modules that can estimate hand pose from a single image can be useful in automatically initializing hand trackers. A machine learning system that estimates hand pose from a single image is described in [17]. Hand pose is estimated based on the geometric moments of the input hand image. In [15] 3D locations of fingers are estimated from a stereo image, and they are used to infer 3D joint angles. In [18] hand pose is estimated from a single image using shadow information and assuming a calibrated light source. Due to

[1]This research was supported in part by the National Science Foundation, under grants IIS-0208876, IIS-9912573, and EIA-9809340.

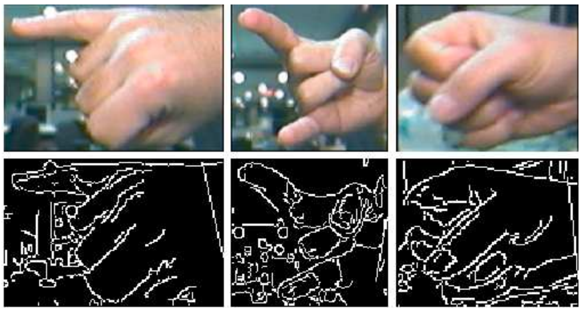

Figure 1. Input hand images, from which our system estimates 3D hand pose, and the corresponding edge images. The edge images are used in computing chamfer distances.

the difficulty of obtaining ground truth estimates, none of these approaches reports quantitative results on real images of hands.

Existing 3D hand pose estimation methods typically assume that the hand is cleanly segmented in the input image. Appearance-based methods for hand pose recognition, like [6, 14, 21, 23], can tolerate clutter, but they are limited to estimating 2D hand pose from a limited number of viewpoints. Our method can handle arbitrary viewpoints.

Our system uses an approximation of the directed chamfer distance to quickly eliminate most candidate database matches. Our approach is a simplified application of the methods described in [5, 10, 11] for constructing low-distortion Euclidean embeddings of arbitrary metric spaces. These concepts are discussed in Section 4.2.

This paper also introduces a novel line matching method. Existing methods [3, 7, 9, 12] use spatial/geometric criteria to determine line correspondences. It is typically assumed that lines have already been extracted. However, in the presence of noise and clutter, the line extraction process can have a significant effect on the accuracy of line matching. Extracting a lot of lines yields too many candidate matches, and geometric criteria may not be sufficient to discriminate among them. Extracting a small number of lines increases the risk that important object features will be excluded. As described in Section 5, our system addresses this problem by extracting a large number of lines, and using *saliency* criteria to discriminate among them. In addition, our method requires no a priori knowledge of camera parameters, imaging noise, or hard geometric constraints.

3. Framework for Hand Pose Estimation

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 2). There are 15 joints, that have a total of 20 degrees of freedom (DOFs). For the 20-dimensional vector of joint angles we use synonymously the terms "hand shape" and "hand configuration."

The appearance of a hand shape also depends on the camera parameters. For simplicity, we consider only the camera viewing direction (two DOFs), and image plane orientation. We use the terms "camera parameters," "viewing parameters" and "3D orientation" synonymously to denote the three-dimensional vector describing viewing direction and image plane orientation.

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

Using these definitions, our framework for hand pose estimation can be summarized as follows:

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. Given an input image, retrieve the database views that are the most similar. Use the parameters of the most similar views as estimates of the hand pose parameters in the input image.

3.1. Database

Our database contains right-hand images of 26 hand shape prototypes. Each prototype is rendered from 86 different viewpoints (Figure 2), sampled approximately uniformly from the surface of the viewing sphere. The rendering is done using a hand model and computer graphics [22]. To accommodate rotation-variant similarity measures (like the chamfer distance), 48 images are generated from each viewpoint, corresponding to 48 uniformly sampled rotations of the image plane. Overall, the database includes 4128 views of each hand shape prototype and 107,328 images overall. We refer to those images using the terms "database images," "model images," or "synthetic images."

4. Approximate Directed Chamfer Distance

The chamfer distance [2] is a well-known method to measure the distance between two edge images. Edge images are represented as sets of points, corresponding to edge pixel locations. The X-to-Y directed chamfer distance $c(X, Y)$ is defined as

$$c(X,Y) = \frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} \|x - y\| \,, \tag{1}$$

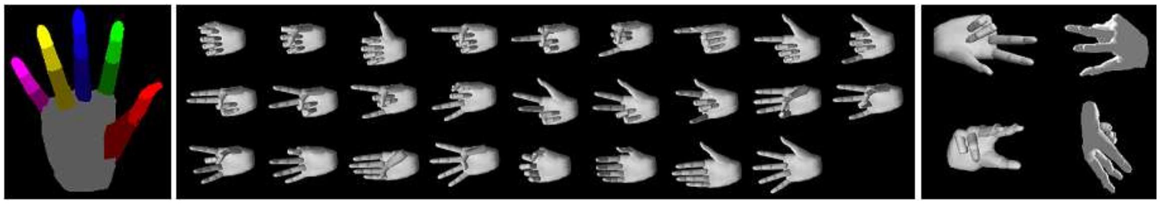

Figure 2. Synthetic images of hands. Left: the articulated hand model. The palm and 15 finger links are shown in different colors. Middle: the 26 basic shapes used to generate model images in our database. Right: four 3D orientations of the same hand shape.

where $\|a - b\|$ denotes the Euclidean distance between two pixel locations a and b. The undirected chamfer distance $C(X, Y)$ is

$$C(X, Y) = c(X, Y) + c(Y, X) . \tag{2}$$

We will use the abbreviations CD to stand for "chamfer distance," and DCD to stand for "directed chamfer distance."

4.1. Efficiency of Chamfer Distance

The model-to-input DCDs between the input image and all model images can be computed very efficiently, using the distance transform method. However, keeping the distance transforms of all database images in memory requires too much RAM, and consequently computing all input-to-model DCDs is much slower: if each edge image has n edge pixels, and the database contains d images, the time complexity is $O(dn \log n)$. In contrast, the time complexity for the model-to-input DCD, using distance transforms, is $O(dn)$.

In the next section we define what we call the *approximate* directed chamfer distance (approximate DCD), which can be evaluated efficiently and which maintains a big part of the discrimination power of the exact DCD.

4.2. Euclidean Embedding of Edge Images

Embeddings of arbitrary metric spaces into a Euclidean space with an L_p norm have received increased attention in recent years [5, 10, 11]. Typically the goal is to find a *low-distortion* embedding E of an arbitrary metric space G into a k-dimensional Euclidean space $\Re^k$, i.e. an embedding under which pairwise distances between points in G are preserved with low distortion in $\Re^k$. Such embeddings are useful when it is computationally expensive to evaluate distances in G, and it is more efficient to map points of G into $\Re^k$ and compute their L_p distance in $\Re^k$.

A class of embeddings often used in this context are *Lipschitz embeddings* [4, 13]. A short discussion of Lipschitz embeddings in the context of database indexing can be found in [10]. Our approach is mostly related to [11], and is a simple application of Lipschitz embeddings. The basic intuition behind Lipschitz embeddings is that, in most spaces, two nearby points have similar distances to any third point. In a metric space, this property holds because of the triangle inequality. The directed chamfer distance does violate the triangle inequality for some triples of images, but we have found experimentally that such triples of images rarely occur in our system.

To define a Lipschitz embedding E from the space of edge images to $\Re^k$, we randomly choose k database edge images $r_1, r_2, ..., r_k$, which we call *reference images*. In our experiments $k = 200$. The embedding E of an arbitrary edge image g is then defined as

$$E(g) = (c(g, r_1), c(g, r_2), ..., c(g, r_k)) , \tag{3}$$

where c is the DCD as defined in Equation 1. Based on the intuition that nearby edge images have similar distances to other edge images, we expect the Euclidean embeddings of nearby edge images to be close to each other.

The embeddings of all database images are computed off-line. Given an input edge image I, we compute its embedding $E(I)$, which involves computing DCDs between I and the k reference images r_i. We define the *approximate directed chamfer distance* $c'(I, B)$ between I and a database image B to be the L_1 distance between $E(I)$ and $E(B)$. We have also experimented with the L_2 norm, it did not lead to significantly different retrieval results. If every edge image contains n edge pixels, the time complexity of computing the approximate DCDs between the input image and d database images is $O(kn \log n + dk)$, which is significantly lower than the complexity of computing the exact DCDs ($O(dn \log n)$), since $k \ll d$. In our implementation, computing the exact DCDs between the input image and all database images takes about five minutes, whereas computing the approximate DCDs takes less than a second.

Experiments with values of k between 200 and 1000 gave roughly comparable results in terms of retrieval accuracy. On the other hand, retrieval accuracy deteriorated as the value of k decreased, especially for values under 100.

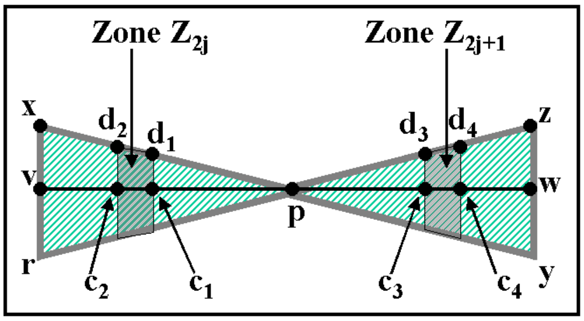

Figure 3. The search region for line segment vw, with center p and length l. It consists of the triangles pxr and pzy. The angle between xp and rp is 7.5 degrees; this value is obtained by dividing 180 degrees by the number N_o of sampled orientations. The search region is divided into l zones Z_i, where $i \in 0, 1, ..., l-1$. Each zone is between two lines perpendicular to vw, that intersect vw at points that are $\lfloor i/2 \rfloor$ and $\lfloor i/2 \rfloor + 1$ pixels away from p. In the picture, c_1 and c_3 are j pixels away from p, and c_2 and c_4 are $j+1$ pixels away from p. Lines $d_1c_1, d_2c_2, d_3c_3, d_4c_4$ are perpendicular to vw. The votes of all point features in the same zone are averaged, so that each zone contributes a vote between -1 and 1.

5. Probabilistic Line Matching

The bounding contours of finger links often appear as nearly-straight line segments in hand images. Some examples can be seen in Figure 4. Those line segments are easy to extract in model images. It is a much harder task to extract them in real hand images, especially when clutter is included.

The problem of matching line segments between two images can be decomposed into four sub-problems: selecting point features, extracting line segments based on the point features, establishing correspondences between segments, and evaluating the quality of the correspondences.

The following subsections describe how our method performs each of these stages. From this point on, the words "lines" and "segments" are both used to mean "finite straight line segments," defined by two endpoints. "Model lines" are lines extracted from database images and "input lines" are lines extracted from the input image.

5.1. Extraction of Edge Points and Line Segments

Our method selects as point features all local maxima in the direction of the image intensity gradient. A lot of noise and clutter is included, but the risk of excluding useful hand features is very small. Examples of point feature selection can be seen in Figure 4.

For line extraction, we pick N_o orientations, uniformly sampled between 0 and 180 degrees, and N_l lengths, uniformly sampled between 5 and 100 pixels. In our experiments, $N_o = 24$ and $N_l = 20$. For every combination of orientation o, length l and point feature position p, the system extracts a line centered at p, with orientation o and length l. Every extracted line segment has two attributes that describe its *saliency*: vote density, and strength (average magnitude of intensity gradient). *Saliency* is a technical term, roughly synonymous with "prominence".

Given a segment vw, centered at p, with length l and orientation o, we determine its vote density and strength from point features in a search region (Figure 3). The search region is split into l parallel zones Z_i, each of which contains points about $\lfloor i/2 \rfloor$ pixels away from p.

Given point feature f_j with edge orientation a, we define the orientation difference $\theta(f_j, vw)$ to be the angle (between 0 and 90 degrees) that a makes with o (the orientation of vw). The vote $V_f(f_j, vw)$ that f_j contributes is

$$V_f(f_j, vw) = \cos(2\theta(f_j, vw)) \ . \quad (4)$$

Values of $\theta(f_j, vw)$ exceeding 45 degrees yield negative votes. In this way, line segments extracted from a cluttered region receive positive support only from similarly oriented point features. Randomly distributed edge orientations will tend to cancel out.

The average vote $V_z(Z_i, vw)$ of each zone Z_i is computed as follows:

$$V_z(Z_i, vw) = \frac{\sum_{f_j \in Z_i} |V_f(f_j, vw)| V_f(f_j, vw)}{\sum_{f_j \in Z_i} |V_f(f_j, vw)|} \ . \quad (5)$$

The vote density $V(vw)$ is the sum of votes of all zones Z_i divided by the length of vw:

$$V(vw) = \frac{\sum_{i \in \{0,1,2,\dots,l-1\}} V_z(Z_i, vw)}{l} \ . \quad (6)$$

The *strength* $S(vw)$ is defined in a similar way. We denote as $S_f(f_j)$ the magnitude of the intensity gradient of the input image at location f_j. Then, we define $S_z(Z_i, vw)$ and $S(vw)$ as

$$S_z(Z_i, vw) = \frac{\sum_{f_j \in Z_i : V_f(f_j, vw) > 0} V_f(f_j, vw) S_f(f_j)}{\sum_{f_j \in Z_i : V_f(f_j, vw) > 0} V_f(f_j, vw)} \ , \quad (7)$$

$$S(vw) = \frac{\sum_{Z_i : V_z(Z_i, vw) > 0} V_z(Z_i, vw) S_z(Z_i, vw)}{\sum_{Z_i : V_z(Z_i, vw) > 0} V_z(Z_i, vw)} \ . \quad (8)$$

5.2. Measuring the Randomness of a Match

In model images we know exactly where each individual finger link is located. Model lines are extracted off-line, by

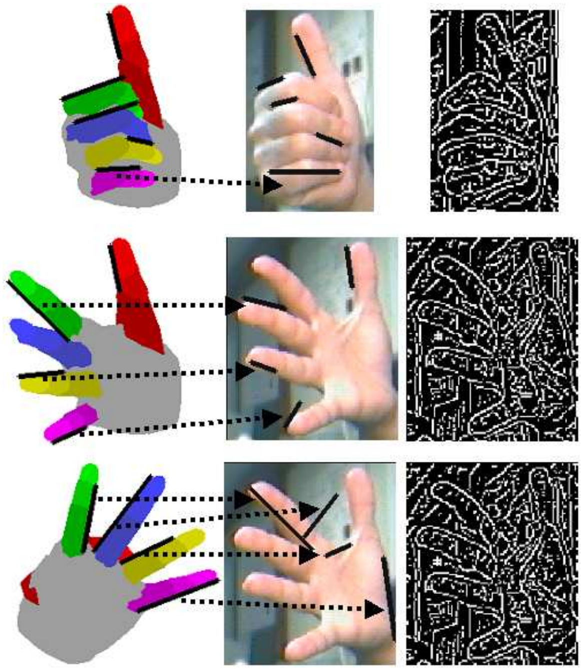

Figure 4. Examples of line matching results. Left column: model images, with some extracted lines shown in black. Middle column: input images, and the best matches for the extracted model lines. Arrows link model lines to input lines, when the correspondence is not clear. Right column: the point features that were extracted from the input images for the purpose of line matching. The matching costs C_L for the three rows were: 0.28, 0.31 and 0.62. In the third row, the hand pose in the model image does not match the hand pose in the input image.

fitting lines to finger contours. No vote density and strength is extracted from those lines, since there is no uncertainty about their saliency.

Given a model line A and an input line B, we define the *quality* Q of the match (A, B) as follows:

$$Q(A,B) = (D(A,B), O(A,B), L(A,B), V(B), S(B)) \;, \tag{9}$$

where $V(B)$ and $S(B)$ are the vote density and strength as defined in Section 5.1. $D(A,B)$ is the Euclidean distance between the center points of A and B. $O(A,B)$ is the angle (between 0 and 90 degrees) between the orientations of A and B. $L(A,B)$ is the absolute difference between the respective lengths of A and B. We use the term "DOLVS vector" for a vector like $Q(A,B)$.

We define a partial order $\leq$ between two DOLVS vectors:

$$(d_1,o_1,l_1,v_1,s_1) \leq (d_2,o_2,l_2,v_2,s_2) \Leftrightarrow$$
$$(d_1 \leq d_2),(o_1 \leq o_2),(l_1 \leq l_2),(v_1 \geq v_2),(s_1 \geq s_2) \tag{10}$$

The sign $\leq$ should be read as "better than or equal to." If, given a model line A and input lines B and C, A is more similar geometrically to B than it is to C, in terms of position, orientation, and length, and B also has higher vote density and strength than C, then B matches A better than C does.

Equation 10 is obviously only a partial order. Given an input image I and the set of line segments X_I extracted from it, we can define a total order in the space of DOLVS vectors, by assigning to every DOLVS vector $b = (d,o,l,v,s)$ a scalar measure of frequency. Let M be the set of lines extracted from all database images. For any line $A \in M$, define the binary function $H(A, X_I, b)$ to be 1 if there exists a line segment $C \in X_I$ such $Q(A,C) \leq b$, and zero otherwise. Then, we can define a scalar measure of frequency $F(b)$:

$$F(b) = E_{A \in M}(H(A, X_I, b)) \;, \tag{11}$$

where E stands for "expected value". $F(b)$ is the percentage of model lines $A \in M$ that have at least one match C in X_I such that $Q(A,C) \leq b$.

Using F, we now define a *total* order $\preceq$ in the space of DOLVS vectors as follows:

$$v_1 \preceq v_2 \Leftrightarrow F(v_1) \leq F(v_2) \;, \tag{12}$$

where v_1, v_2 are DOLVS vectors. Given this total order, we can easily define the best match for model line A in X_I, to be the line $C \in X_I$ that minimizes $F(Q(A,C))$; we consider the quality of the match Q(A,C) to be the least likely to be observed by chance, among the qualities of all matches between A and a segment in X_I.

Suppose that, for some $A \in M$ and some $C \in X_I$, $F(Q(A,C)) = f$. We can ask the following question: what percentage of model lines in M have a match in X_I whose frequency is less than f? The answer to that question would be a quantitative measure of how *unusually good* the match between A and C is. To define this measure, we first define a binary function $H'(A, X_I, f)$ to be 1 if there exists a $C \in X_I$ such that $F(Q(A,C)) \leq f$, and zero otherwise. Then, we define the probability $P_e(f)$ that a random model line has at least one match in X_I with frequency at most f:

$$P_e(f) = E_{A \in M}(H'(A, X_I, f)) \;. \tag{13}$$

$P_e(f)$ is a monotonic function of f, but it has a more intuitive interpretation than f. If a model line A has a match C for which $P_e(F(Q(A,C))) = 0.03$, then we know that only 3% of model lines have such good matches in X_I, i.e. matches with such a low value of P_e.

We define $P_{\min}(A, X_I)$ to be the value P_e for the best match of A in X_I:

$$P_{\min}(A, X_I) = \min_{C \in X_I}(P_e(F(Q(A,C)))) \;. \tag{14}$$

In order to make it easy to compute and to look up values of F and P_e, we discretize the 5-dimensional space of DOLVS vectors into a DOLVS histogram. In the bin b associated with value (d, o, l, v, s) we store $F(b)$. Computing $F(b)$ is done by sampling from the set of all model lines M, and matching each sampled model line with each line in X_I. In our experiments, the DOLVS histogram has 250,000 bins (25,10,10,10 and 10 values sampled respectively from the range of D,O,L,V and S). About 2,000 model lines are sampled to calculate $F(b)$ for each bin b in the histogram. After values $F(b)$ have been computed for every bin b, we can then compute the value of $P_e(F(b))$, again by sampling from the set of all model lines M.

The values of F and P_e for every bin in the DOLVS histogram are *recomputed* for every new input image. The *rareness* of matches is solely determined based on the current input image and the database images.

Examples of best matches found using our method can be seen in Figure 4.

5.3. Line Matching Cost

Suppose we have a model image J, which contains the set of lines $\{J_1, J_2, ..., J_j\}$, and an input image I with a set of lines X_I. We define the line-based matching cost $C_L(J, I)$ to be

$$C_L(J, I) = \frac{1}{j} \sum_{i=1}^{j} P_{\min}(J_i, X_I) \ . \qquad (15)$$

C_L is the average of the probabilities of the best matches of all lines in the model image. Low values of C_L indicate that the model lines were matched unusually well in the input. One drawback of C_L is that model images with a small number of lines are more likely to achieve a low score by chance, compared to model images with a large number of lines. To account for that, we compute, for each possible number of lines, the standard deviation of C_L for model images with that number of lines. Using the standard deviation we normalize C_L, so that database images with any number of lines tend to have scores in the same range.

As can be seen in Figure 4, line correspondences between correct database matches and the input image are frequently not identified correctly. Model lines are frequently matched to clutter or to nearby fingers. However, the matching cost C_L is on average much lower for correct matches (about 0.35) than it is for random model images (about 0.5). Using that cost to identify correct matches significantly improves retrieval accuracy for cluttered images.

6. Two-Step Retrieval

To combine k different similarity measures, given an input image I, we first obtain rankings of the database images in order of similarity to I using each measure separately. We denote the rank of the i-th database image V_i under the j-th measure as r_{ij}. We define a new, combined similarity measure $M_c(V_i, I)$ as

$$M_c(V_i, I) = \sum_{j=1}^{k} (\log r_{ij}) \ . \qquad (16)$$

Then, we can rank the synthetic views again, using the values of the combined measure. A more detailed discussion of this method can be found in [1].

Since the model-to-image DCD and the approximate image-to-model DCD can be evaluated very fast, we first combine these two measures to obtain a preliminary ranking of all database images. Then we keep a fixed number of top matches (1000 in our experiments), and we rerank those matches using the undirected chamfer distance, orientation histograms [1] and the line matching cost C_L. In this way we obtain the final ranking of database images. Two-step retrieval significantly improves retrieval speed, while preserving to a large degree the accuracy of the computationally expensive similarity measures.

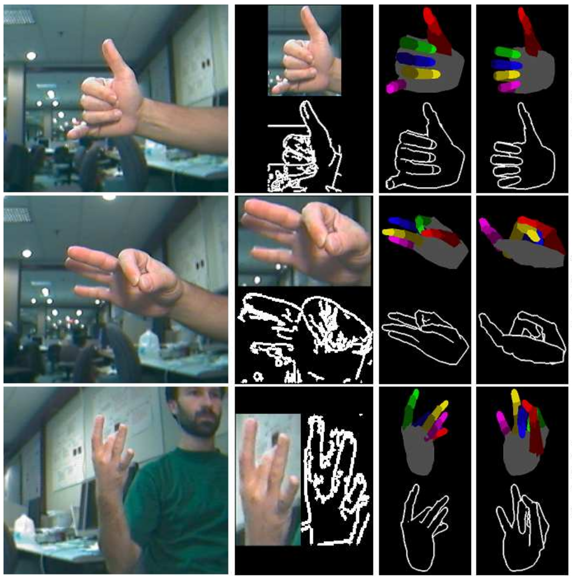

Figure 5. Examples of input images and results. First column: example input images. Second column: hand images segmented using "box" segmentation, and the corresponding edge images. Third column: highest ranking correct match for each input image (synthetic hand images and corresponding edge images). Fourth column: highest ranking incorrect match for each input image. Fifth column: for each input image, rank of highest ranking correct match, rank of highest ranking incorrect match, and frontal angle of the hand pose in the input image.

7. Experiments

We have tested our system with 250 real images of right hands. The hand shape in all test images is one of the 26 shape prototypes used to generate the database. There are no restrictions on the 3D orientation of the hand in the test images, since the database contains samples from the entire space of possible 3D orientations for every shape prototype. We have 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 manual estimates by different people varied by 10-30 degrees. Model views cannot be aligned perfectly because the anthropometric parameters (like finger lengths and widths) of hands in test images do not match those of the model, and because the hand shapes in the real images are not exact replications of the 26 shape prototypes.

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 [1]. On average, there are 30.4 correct matches for each test image in the database. Our measure of retrieval accuracy for a given test image I is the rank of the *highest-ranking correct match* that was retrieved for I. 1 is the highest (best) possible rank.

The experiments were aimed at demonstrating the performance of the system under clutter. We tested two different hand segmentation methods: "fine" segmentation and "box" segmentation. Fine segmentation locates the hand region by applying skin detection. Box segmentation extracts the entire bounding box of the hand region found by fine segmentation. The bounding box includes clutter from the background. Table 1 shows retrieval results for the test images. As expected, accuracy decreases with clutter. Very often, as Figure 5 shows, the hand pose in the top database matches is very different than the pose in the input image. At the same time, Table 1 shows that the two similarity measures introduced in this paper (approximate image-to-model DCD and line matching cost C_L) significantly improve retrieval accuracy under clutter over existing methods. The approximate image-to-model DCD also has a big effect on accuracy under "fine" segmentation.

Given an input image, we use the term "frontal angle" to denote the angle (from 0 to 90 degrees) between the viewing direction and a line perpendicular to the palm. Figure 5 shows the frontal angles of some test images. Table 2 shows the median rank of the highest-ranking correct matches for test images observed from different frontal angle ranges. Retrieval accuracy tends to be higher for frontal views (low frontal angles), where more features are visible.

Results for first retrieval step:					
Method used	1-250	1-500	1-1000	1-2000	median
F, MICD	55.6	64.8	76.8	88.0	193
F, MICD, AIMCD	74.4	82.0	87.6	92.4	49
B, MICD	55.6	68.0	78.0	84.4	169
B, MICD, AIMCD	63.2	74.4	82.8	90.8	127

Results for second retrieval step:						
Method used	1	1-4	1-16	1-64	1-256	median
F, CD,	11.6	25.6	47.2	66.0	82.4	20
F, CD, EO	9.2	23.2	42.0	63.6	83.6	23
F, CD, EO, LM	13.6	26.4	45.2	67.6	84.0	21
B, CD	6.0	10.8	23.2	40.4	61.6	127
B, CD, EO	6.0	11.2	27.6	46.8	68.0	84
B, CD, EO, LM	8.0	18.4	32.8	54.4	74.8	47

Table 1. For every method used, we show the percentage of test images for which the highest ranking correct match was within each range, and the median rank of the highest ranking correct matches. In the first step, all database images are ranked, and the 1000 top matches (using MICD and AIMCD) are reranked in the second step. F and B stand for "fine" and "box segmentation." Other abbreviations: MICD (model-to-image chamfer distance), AIMCD (approximate image-to-model chamfer distance), CD (chamfer distance), EO (edge orientations), LM (line matching).

Frontal angle	0-22.5	22.5-45	45-67.5	67.5-90
# of images	52	71	93	34
Median using F	9	12	50	26
Median using B	24	36	93	44

Table 2. Accuracy over different frontal angles. For each range, we indicate the number of test images with frontal angles in that range, and the median rank of the highest ranking correct matches for those images.

The original bounding boxes of the hand in the input images had sizes that were mostly between 80x110 and 120x160 pixels. We have also tested our system with 177 images where the hand is smaller (bounding box sizes between 50x80 and 70x100 pixels). The accuracy is significantly worse (the median rank of the highest ranking correct matches for those images is 222 under "box" segmentation). We have also run experiments on the 276 test images we used in [1], where the hand is very cleanly segmented, and the bounding box sizes are about 150x200. For those images, the median rank of the highest ranking correct matches is 5.

Total processing time, including hand segmentation, extraction of line segments, and the two retrieval steps, was about 15 seconds per input image, on a PC with a 1.2GHz Athlon processor, using a rather unoptimized C++ implementation. The program used about 300MB of memory.

8. Discussion and Future Work

Compared to the system we described in [1], the current system constitutes an improvement, in that it can handle input images for which segmentation is rough and includes some clutter. Retrieval accuracy is still too low for the system to be used as a stand-alone module for 3D hand pose estimation. The system may be useful in providing single frame estimates to a 3D hand tracker, to achieve automatic initialization and error recovery. Ideally the system would provide multiple estimates (a few tens) to the tracker, and the tracker would reinforce correct estimates as it tracked them through time. The current accuracy of the system may be sufficient for such an application. It will be interesting to see how integrating our system with a 3D hand tracker will work in practice.

Our system can only handle a few tens of hand shapes. This number may be sufficient for some applications, like recognition of sign languages (fewer than 100 basic hand shapes are used in ASL) or human computer interfaces. On the other hand, a larger number of hand shapes must be considered for unconstrained 3D hand pose estimation.

Although our method allows for segmentation errors in identifying the boundaries of the hand, it still requires fairly accurate estimates of the center and size of the hand. The system will not work if the estimated diameter of the hand is wrong by, say, a factor of two. Designing similarity measures that can tolerate such errors remains a challenging problem.

9. Conclusions

We have presented a method that estimates hand pose using image database indexing methods. Two novel clutter-tolerant methods, the approximate image-to-model chamfer distance, and a similarity measure based on probabilistic line matching, are employed to improve system performance. Both methods are generally applicable, and can be employed in any domain where the chamfer distance is used or where line matching needs to be performed.

References

[1] V. Athitsos and S. Sclaroff. An appearance-based framework for 3D hand shape classification and camera viewpoint estimation. In *Automatic Face and Gesture Recognition*, 2002.

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

[3] J. Beveridge and E. Riseman. How easy is matching 2D line models using local search? *PAMI*, 19(6):564–579, 1997.

[4] J. Bourgain. On Lipschitz embeddings of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52:46–52, 1985.

[5] C. Faloutsos and K. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *ACM SIGMOD International Conference on Management of Data*, pages 163–174, 1995.

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

[7] W. Grimson, T. Lozano-Perez, and D. Huttenlocher. *Object Recognition by Computer: The Role of Geometric Constraints*. MIT Press, 1990.

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

[9] S. Heuel and W. Förstner. Matching, reconstructing and grouping 3D lines from multiple views using uncertain projective geometry. In *CVPR*, volume 2, pages 517–524, 2001.

[10] G. Hjaltason and H. Samet. Contractive embedding methods for similarity searching in metric spaces. Technical Report TR-4102, Computer Science Department, University of Maryland, 2000.

[11] G. Hristescu and M. Farach-Colton. Cluster-preserving embedding of proteins. Technical Report 99-50, Computer Science Department, Rutgers University, 1999.

[12] S. Lanser and T. Lengauer. On the selection of candidates for point and line correspondences. In *International Symposium on Computer Vision*, pages 157–162, 1995.

[13] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. In *IEEE Symposium on Foundations of Computer Science*, pages 577–591, 1994.

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

[15] C. Nölker and H. Ritter. Parametrized SOMs for hand posture reconstruction. In *IJCNN*, pages 4139–4144, 2000.

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

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

[18] J. Segen and S. Kumar. Shadow gestures: 3D hand pose estimation using a single camera. In *CVPR*, pages 479–485, 1999.

[19] 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.

[20] B. Stenger, P. Mendonça, and R. Cipolla. Model based 3D tracking of an articulated hand. In *CVPR*, volume 2, pages 310–315, 2001.

[21] J. Triesch and C. von der Malsburg. Robotic gesture recognition. In *Gesture Workshop*, pages 233–244, 1997.

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

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

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