Boston University Computer Science Tech. Report No. 2003-010, April 1, 2003.
Accepted to the Gesture Workshop, April 2003.

Database Indexing Methods for 3D Hand Pose Estimation

Vassilis Athitsos and Stan Sclaroff*

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

Abstract. Estimation of 3D hand pose is useful in many gesture recognition applications, ranging from human-computer interaction to automated recognition of sign languages. In this paper, 3D hand pose estimation is treated as a database indexing problem. Given an input image of a hand, the most similar images in a large database of hand images are retrieved. The hand pose parameters of the retrieved images are used as estimates for the hand pose in the input image. Lipschitz embeddings of edge images into a Euclidean space are used to improve the efficiency of database retrieval. In order to achieve interactive retrieval times, similarity queries are initially performed in this Euclidean space. The paper describes ongoing work that focuses on how to best choose reference images, in order to improve retrieval accuracy.

1 Introduction

Automatic estimation of the 3D pose of a human hand can be useful 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. This paper describes a system that provides estimates of 3D hand pose from a single 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.

In [2] we describe the use of Lipschitz embeddings for approximating the chamfer distance, in order to improve retrieval efficiency. With Lipschitz embeddings, the distance between two images can be approximated by looking at the distance between each of those images and a set of reference images. The basic intuition is that, if two images are similar to each other, then their distances to other images will also be similar. In this paper we discuss and evaluate different methods for choosing reference images, in order to improve overall system accuracy.

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

2 Related Work

Computer vision systems that estimate 3D hand pose typically do it in the context of tracking hand pose in a video sequence [7, 13, 16, 17, 21]. In the tracking context, the basic assumption is that the system already knows the hand pose in the previous video frame. The goal of the system is to update the hand pose to optimally match the observations in the current frame. Such a tracking framework is insufficient for fully automatic 3D hand pose estimation, because it does not address two problems: estimating hand pose in the first video frame, when no information about previous frames is available, and recovering from errors, when the hand pose for the previous frame was estimated incorrectly.

Solutions to those problems may be provided by methods that estimate hand pose from a single image. A machine learning method for single-image hand pose estimation is described in [14]. In [12] 3D locations of fingers are estimated from a stereo image, and they are used to infer 3D joint angles. In [15] hand pose is estimated from a single image using shadow information and assuming a calibrated light source. Due to 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, 11, 18, 20], can tolerate clutter, but they are limited to estimating 2D hand pose from specific viewpoints.

Our system uses the chamfer distance as a measure of similarity between hand images. The accuracy of the chamfer distance degrades gracefully in the presence of clutter and errors in hand segmentation. The chamfer distance is approximated using Lipschitz embeddings, using an approach similar to the methods described in [5, 8, 9]. The main novelty in this paper with respect to previous approaches is the formulation and quantitative comparison of methods to choose reference images, in Section 5.

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 (Figure 1a). Overall, the values of all joint angles can be specified using a 20-dimensional vector, for which we use synonymously the terms "hand shape" and "hand configuration."

The appearance of a hand shape also depends on the 3D orientation of the hand. Given a hand configuration vector $C_h = (c_1, ..., c_{20})$ and a 3D orientation 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.

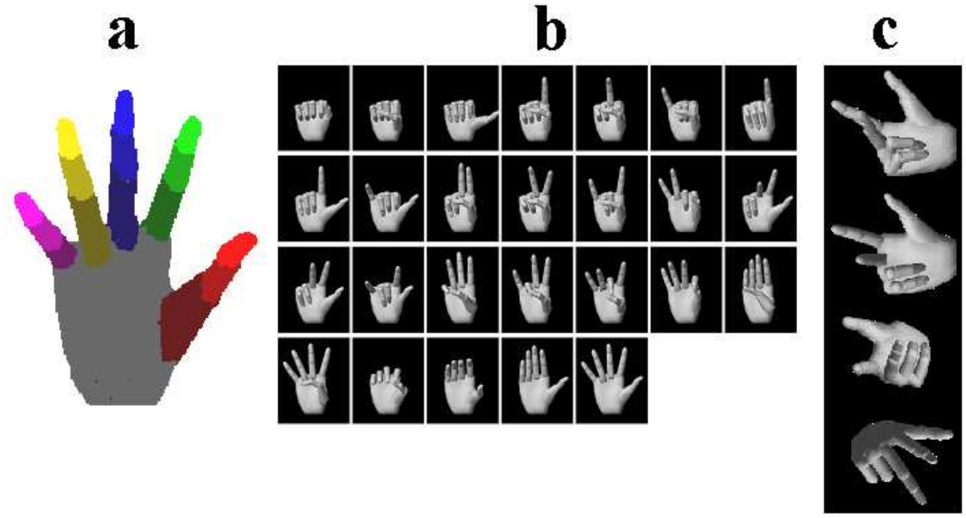

Fig. 1. Synthetic images of hands: a). The articulated hand model. The palm and 15 finger links are shown in different colors. b). The 26 basic shapes used to generate model images in our database. c). Four different 3D orientations of the same hand shape.

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 for the input image.

3.1 Database

Our database contains right-hand images of 26 hand shape prototypes (Figure 1b). Each prototype is rendered from 86 different viewpoints (Figure 1c), sampled approximately uniformly from the surface of the viewing sphere. The rendering is done using a hand model and computer graphics [19]. To accommodate rotation-variant similarity measures (like the chamfer distance), 48 images are generated from each viewpoint, giving a total of 4128 different 3D orientations of the hand. Overall, the database includes 107,328 images. We refer to those images using the terms "database images," "model images," or "synthetic images."

4 Approximating the Chamfer Distance

Given an input image, the system has to identify the database images that are the closest to the input. In our system we measure distance between edge images, because edge images tend to be more stable than intensity images with respect to different lighting conditions. The chamfer distance [3] 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. Given two edge images, X and Y, the chamfer distance $C(X,Y)$ is:

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

where $\|a - b\|$ denotes the Euclidean distance between two pixel locations a and b. $C(X,Y)$ penalizes for points in either edge image that are far from any point in the other edge image. Figure 2 shows an illustration of the chamfer distance.

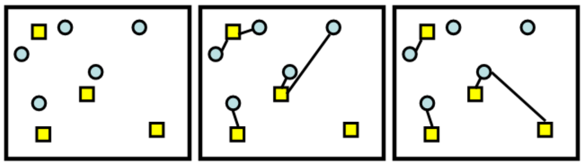

Fig. 2. An example of the chamfer distance. The left image shows two sets of points: a set of circles and a set of squares. The middle image shows a link between each circle and its closest square. The circle-to-square directed chamfer distance is the average distance between a circle and its closest square. The right image shows a link between each square and its closest circle. The square-to-circle chamfer distance is the average distance between a square and its closest circle. The chamfer distance (also known as *undirected chamfer distance*) between squares and circles is the sum of the two directed distances.

4.1 Lipschitz Embeddings

In our application, calculating the chamfer distance between the input image and all database images takes too long (over four minutes) to be used in interactive applications. However, we can obtain an efficient approximation of the chamfer distance by embedding edge images into a Euclidean space.

Embeddings of arbitrary metric spaces into a Euclidean space with an L_p norm have received increased attention in recent years [5, 8, 9]. Typically the goal is to find a *low-distortion* embedding E of an arbitrary metric space G into a 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, 8–10]. The basic intuition behind Lipschitz embeddings is that two objects that are close to each other typically have similar distances to all other objects.

An everyday example that illustrates this property is looking at distances between cities. The distance from New York to Boston is about 240 miles, and the distance from New York to Los Angeles is about 2800 miles. Suppose that we did not know these two distances. Furthermore, suppose that someone gave us, for 100 towns spread across the United States, their distances to New York, Boston and Los Angeles. What would that information tell us about the distances from New York to Boston and from New York to Los Angeles?

First we would notice that the distance from each town to New York is always within 240 miles or less of the distance between that town and Boston. On the other hand, there are some towns, like Lincoln, Nebraska, whose distances from Los Angeles and New York are very similar, and some towns, like Sacramento, whose distances to Los Angeles and New York are very different (Sacramento-Los Angeles is 400 miles, Sacramento-New York is 2800 miles). Given these distances, we could deduce that, most likely, New York is a lot closer to Boston than it is to Los Angeles.

This property, that nearby objects have similar distances to all other objects, holds for objects in any space where distances obey the triangle inequality. Using this property, we can now define an efficient approximation of the chamfer distance.

Suppose that we have chosen a *reference set* of k database edge images $R_1, R_2, ..., R_k$. Then, we can define a function E, mapping the space of edge images to the Euclidean space $\Re^k$ as follows:

$$E(G) = (C(G, R_1), C(G, R_2), ..., C(G, R_k)) \ . \tag{2}$$

where C is the chamfer distance, defined in Equation 1, and G is an edge image. The function E is also called an *embedding* of the space of edge images into $\Re^k$, and it is a special case of Lipschitz embeddings [4, 10].

We define the *approximate chamfer distance* C' between two edge images A and B to be the L_1 distance between $E(A)$ and $E(B)$:

$$C'(A, B) = \sum_{i=1}^{k} |C(A, R_i) - C(B, R_i)| \ . \tag{3}$$

The actual value of $C'(A, B)$ is not necessarily similar in scale to the value $C(A, B)$. However, $C'(A, B)$ is an approximation of $C(A, B)$ in the sense that, when $C(A, B)$ is much smaller than $C(A, G)$, then we also expect $C'(A, B)$ to be smaller than $C'(A, G)$.

The time complexity of computing the approximate distance C' between an edge image and d database edge images is $O(kn \log n + kd)$, where n is the number of edge pixels in every edge image. On the other hand, computing the chamfer distance C takes $O(dn \log n)$ time. The complexity savings are substantial when k is much smaller than d. In our system it takes more than four minutes to compute the chamfer distances between the input image and all database images. In contrast, for $k = 200$, it takes a fraction of a second to compute the corresponding approximate distances C'. To achieve this speedup we need to precompute, offline, the distances between every database image and every reference image.

An important caveat in approximating the chamfer distance C using Lipschitz embeddings is that the chamfer distance does not always obey the triangle inequality. However, our experiments indicate that only rarely do triples of edge images violate the triangle inequality, and that C' is a useful approximation of C.

5 Choosing Reference Images

In order to define the approximate chamfer distance C' we need to specify a set of k reference images R_i. In this section we discuss methods for selecting reference images among database edge images. The simplest approach is to select reference images randomly, and choose the k that leads to the best retrieval accuracy. The approaches discussed below choose reference images in a more selective way, trying to improve the approximation accuracy of the embedding.

5.1 Minimizing Average Distortion

Given the embedding E defined in Section 4.1, from the space of edge images with chamfer distance C to the Euclidean space $\Re^k$ with the L_1 distance, we would like to know how close the original chamfer distance between edge images is to the L_1 distance between the embeddings of the edge images. More generally, given an embedding F from space X with distance D_x to space Y with distance D_y, we can ask how well F preserves distances. Ideally, for any two objects $x_1, x_2 \in X$, it should hold that $D_x(x_1, x_2) = D_y(F(x_1), F(x_2))$. In that case, we can say that F perfectly preserves distances. Unfortunately, there are choices of X, D_x, Y, D_y for which no embedding can preserve distances perfectly [8].

Given a pair of objects $x_1, x_2 \in X$, we can measure how well F preserves their distance by defining the distortion $Q(x_1, x_2, F)$ as

$$Q(x_1, x_2, F) = \frac{D_y(F(x_1), F(x_2)) - D_x(x_1, x_2)}{D_x(x_1, x_2)} . \tag{4}$$

Q measures the amount of change that the embedding has caused to the distance between x_1 and x_2 as a fraction of the original distance $D_x(x_1, x_2)$.

Given m pairs of objects (a_i, b_i), we can evaluate the accuracy of the embedding by computing the *average distortion* $Q'(F)$ that F causes to those pairs:

$$Q'(F) = \frac{\sum_{i=1}^{m} |Q(a_i, b_i, F)|}{m} . \tag{5}$$

Suppose that there exists a constant c such that, for any x_1 and x_2, it holds that $D_y(F(x_1,), F(x_2)) = cD_x(x_1, x_2)$. We would consider such an embedding to be a zero-distortion embedding, because it would maintain all the important structure in the original space X, like nearest neighbors and clusters. However, for high values of c, $Q'(F)$ may be large. In order for $Q'(F)$ to give a meaningful value, we must first scale F by an appropriate constant, so that $Q'(F)$ is minimized.

Since low-distortion embeddings are desirable, a strategy for selecting reference images is to try to pick the ones that minimize average distortion. In order to make the selection task computationally feasible, we pick reference images in a greedy fashion, one by one. We pick the first reference image R_1 to be the database image which, as a single reference image ($k = 1$ in Equation 3), achieves the lowest value of Q' for a selected set of pairs of images (in our implementation, Q' is measured on 50,000 randomly chosen pairs of database images). If we have already chosen i reference images, then the $i + 1$ reference image is chosen to be the one that, when used together with the already chosen reference images, achieves the lowest average distortion. We stop choosing reference images when the average distortion stops decreasing significantly.

To improve the running time of the algorithm, we pick reference images not from the entire set of database images, but from a subset, consisting of 8944 images. The database contains 48 image plane rotations per camera viewpoint (Section 3.1). In the subset from which we choose reference images we only include 4 image plane rotations per camera viewpoint.

5.2 Minimizing Worst-Case Distortion

An alternative to minimizing average distortion is minimizing worst-case distortion. We can define the scaling $S(x_1, x_2, F)$ that an embedding F causes to the distance between x_1 and x_2 as:

$$S(x_1, x_2, F) = \frac{D_y(F(x_1), F(x_2))}{D_x(x_1, x_2)} \ . \tag{6}$$

Then, given m pairs of objects (a_i, b_i), we can find the pairs that attain the highest and lowest values of S, $s_{\max}$ and $s_{\min}$ respectively, and we can define the worst-case distortion $S'(F)$ as

$$S'(F) = \frac{s_{\max}}{s_{\min}} \ . \tag{7}$$

The worst-case distortion is a commonly used measure for evaluating the distortion of embeddings. The term "distortion" is often used for this measure in the literature [8, 10], but we have opted for the term "worst-case distortion" to distinguish it from the measure of "average distortion" introduced in Section 5.1.

If F preserves distances perfectly (or up to scale), then $S'(F) = 1$. Computing $S'(F)$ provides a statistical guarantee, that it is very uncommon to find a triple of objects x_1, x_2, x_3 whose pairwise distances are distorted by F by a factor greater than $S'(F)$.

It is easy to modify the algorithm outlined in Section 5.1, so that it chooses reference images that minimize worst-case distortion, as opposed to average distortion. At each step, we add to the reference set the image that, combined with the previously chosen reference images, yields the smallest worst-case distortion.

5.3 Using a Training Set

Another possible criterion for picking reference images is to directly try to maximize retrieval accuracy. The retrieval accuracy attained by different choices of reference images can be measured on a training set of real hand images. To choose reference images that maximize retrieval accuracy, we can again use a modified version of the algorithm described in Section 5.1. When we choose each reference image, we simply choose the one that leads to the best retrieval accuracy on the training set. The next section discusses measures of retrieval accuracy.

In our implementation, we have used 276 images of real hands as the training set for this method. Those images are not part of the database (which only includes synthetic images) and were not included in the test set of 426 real hand images that we used in the experiments described in Section 7.

6 Accuracy Evaluation

To evaluate retrieval accuracy, we use a test set of real images of hands. Each image in this set depicts one of the 26 hand shapes used in generating the database, in an arbitrary 3D orientation. For each test image we manually establish pseudo-ground truth, using the rendering software to identify the hand shape and 3D orientation under which the model hand looks the most similar to the test image. We consider a database image V to be a *correct match* for a test image I if the hand shapes in V and I are the same, and the 3D hand orientations of V and I differ by less than 30 degrees. On average, for each test image, there are 30.4 correct matches in the database. Our measure of retrieval accuracy for a given test image I is the rank of the *highest-ranking correct match*. For example, if, for a given input image, the top 9 database matches are incorrect but the 10th match is correct, then the rank of the highest ranking correct match is 10.

To evaluate retrieval accuracy on a set of test images, we use the median rank of the highest ranking correct match. For example, if we have a test set of 100 images, and the median rank of the highest ranking correct match is 34, then we know that at least 50 of those test images had a correct match of rank at most 34.

Our evaluation method is discussed in more detail in [1].

7 Experiments

We have implemented the four different methods discussed in Section 5. Table 1 shows the accuracy attained by each method for specific values of k (the number of reference images used). The accuracy was measured on a test set of 426 images, some of which can be seen in Figure 3. We can see that no dramatic improvements in accuracy are achieved by using any of the four methods. Overall, minimizing the average distortion seems to lead to better accuracy.

Since minimizing average distortion seems to give the best accuracy, we checked whether the accuracy attained by a set of reference images could be reliably predicted by the average distortion corresponding to those images. The answer turned out to be negative. Table 2 shows the average distortion and accuracy attained by random picking and by minimizing average distortion. Using 4 reference images selected by minimizing the average distortion we attain an

Table 1. Accuracy attained by the four different methods of choosing reference images, for different numbers of reference images. The measure of accuracy is the median rank of the highest ranking correct matches retrieved for a set of 426 test images.

# of reference images	4	8	16	32	64	128
random	628	636	527	459	437	463
average distortion	1009	408	453	343	345	349
worst-case distortion	846	570	452	416	477	490
training	916	793	651	492	421	424

Table 2. Average distortions and accuracies attained by two different methods of choosing reference images: picking random images and choosing images that minimize average distortion. We see that the average distortion attained by a set of reference images cannot reliably predict the corresponding accuracy.

# of reference images	4	8	16	32	64	128
random: distortion	.29	.27	.26	.20	.20	.18
random: accuracy	628	636	527	459	437	463
average dist.: distortion	.21	.16	.14	.13	.12	.12
average dist.: accuracy	1009	408	453	343	345	349

Table 3. Comparison of two methods for choosing reference images: choosing randomly (method RAND), and choosing images that minimize average distortion (method MAD). In each trial, 16,32 or 64 reference images were chosen randomly from a set of candidate images, and an equal number of reference images were chosen from the same set of candidates using the method of minimizing average distortion. Classification accuracy is measured as in Table 1.

Number of reference images:	16	32	64
Number of trials:	26	26	26
Times RAND outperformed MAD:	6	2	2
Times MAD outperformed RAND:	20	24	24
(Mean, std) of accuracy using RAND:	(621, 170)	(584, 117)	(541, 114)
(Mean, std) of accuracy using MAD:	(482, 60)	(438, 54)	(413, 46)

average distortion of .21, which is smaller than the average distortion attained by choosing 4, 8, or 16 reference images randomly. However, the classification accuracy is better using the randomly chosen reference images. Therefore, average distortion by itself is not an accurate predictor of classification accuracy.

In order to evaluate the degree to which minimizing average distortion outperforms picking reference images randomly, we ran multiple experiments to compare these two methods. First, we created 26 mutually disjoint sets of candidate reference images. Each of those sets contained 344 images, and consisted of database images generated using only one of the 26 hand shapes. Then, from each set of candidate reference images, and for each value of k among $\{16, 32, 64\}$, we picked two reference sets of size k: a set of images chosen randomly, and a set of images chosen by minimizing average distortion. We compared the classi-

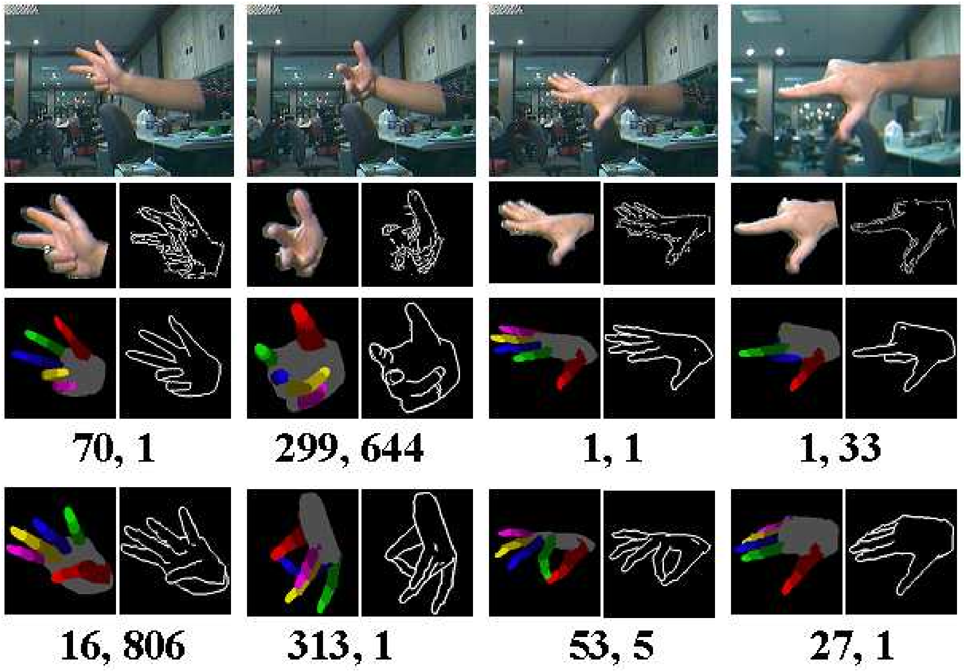

Fig. 3. Examples of input images and results. First row: example input images. Second row: segmented hand images and the corresponding edge images. Third row: examples of correct matches (synthetic hand images and edge images). Fourth row: the ranks of the correct matches using the approximate chamfer distance, and using the combination of the two distances. Fifth row: examples of incorrect matches. Sixth row: ranks of the incorrect matches using the approximate chamfer distance and using the combination of the two distances.

fication accuracy obtained by using each of those two reference sets. Overall, we compared 78 pairs of reference sets. Table 3 shows the results. In the majority of the comparisons (68 out of 78), the reference set chosen by minimizing average distortion outperformed the corresponding reference set of equal size that was chosen randomly from the same set of candidates.

We also compared the accuracy of the approximate chamfer distance with that of the exact chamfer distance. For the approximate chamfer distance, using 128 reference images chosen by minimizing average distortion, the median of the highest correct match is 349, and retrieval takes less than a second. If we use the exact chamfer distance, the median of the highest ranking correct match is 38, and retrieval takes about five minutes.

A reasonable compromise in terms of speed and accuracy is to employ a two-step retrieval approach: use the approximate chamfer distance to select some top candidate matches, and then use the exact chamfer distance to rank those matches. In an experiment, we used the approximate chamfer distance to pick 5,000 candidate matches, and then we applied the chamfer distance to sort those matches. The median of the highest ranking correct match after the second step was 44, and the retrieval time was 15 seconds. We see that by combining the two methods we get significant gains in retrieval efficiency (15 seconds versus 5 minutes), with only a small loss in retrieval accuracy (44 versus 38).

In [1] and [2] we describe additional similarity measures, that can be used to further improve classification accuracy and retrieval efficiency.

8 Discussion and Future Work

Overall, the approximate chamfer distance, obtained by embedding edge images into a Euclidean space, is a useful approximation of the chamfer distance, that can be used to significantly reduce retrieval time. Several open question remain on how to imrpove the approximation accuracy. We are particularly interested in the following questions:

- Is there a fundamental limit to the minimum distortion that can be achieved by the approximate chamfer distance, with respect to the original chamfer distance, regardless of the number and choice of reference images?
- Are there other edge images, that are not included in the database, and that may not even be images of hands, that might be more suitable reference images?

An additional issue we are particularly interested in is exploring the consequences of the fact that the chamfer distance does not obey the triangle inequality. There is a more computationally expensive distance measure, called minimum-weight bipartite perfect matching, that actually obeys the triangle inequality and that intuitively has many similarities to the chamfer distance. It will be interesting to investigate whether an approximation of bipartite matching using Lipschitz embeddings would lead to lower distortion and/or better classification accuracy than the approximate chamfer distance.

At the same time, we are working on improving the overall accuracy of the system, by designing more accurate similarity measures. We are also interested in integrating our system with a 3D hand tracker, in order to achieve a fully-automated 3D hand pose estimation and tracking system.

9 Conclusions

Our system estimates the 3D hand pose of an image by retrieving the most similar images from a large database. The chamfer distance is a useful measure for evaluating image similarity. An efficient approximation of the chamfer distance can be obtained using Lipschitz embeddings. Combining the approximate chamfer distance with the exact chamfer distance, in a two-step retrieval process, significantly reduces retrieval time, while maintaining most of the retrieval accuracy of the exact chamfer distance. Ongoing work focuses on improving the attained approximation, and exploring the theoretical limits of this method.

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. V. Athitsos and S. Sclaroff. Estimating hand pose from a cluttered image. In *CVPR*, 2003.
3. H.G. Barrow, J.M. Tenenbaum, R.C. Bolles, and H.C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *IJCAI*, pages 659–663, 1977.
4. 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.I. 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.T. Freeman and M. Roth. Computer vision for computer games. In *Automatic Face and Gesture Recognition*, pages 100–105, 1996.
7. T. Heap and D. Hogg. Towards 3D hand tracking using a deformable model. In *Face and Gesture Recognition*, pages 140–145, 1996.
8. 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.
9. G. Hristescu and M. Farach-Colton. Cluster-preserving embedding of proteins. Technical Report 99-50, Computer Science Department, Rutgers University, 1999.
10. 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.
11. B. Moghaddam and A. Pentland. Probabilistic visual learning for object detection. Technical Report 326, MIT, June 1995.
12. C. Nölker and H. Ritter. Parametrized SOMs for hand posture reconstruction. In *IJCNN*, pages 4139–4144, 2000.
13. 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.
14. R. Rosales, V. Athitsos, L. Sigal, and S. Sclaroff. 3D hand pose reconstruction using specialized mappings. In *ICCV*, volume 1, pages 378–385, 2001.
15. J. Segen and S. Kumar. Shadow gestures: 3D hand pose estimation using a single camera. In *CVPR*, pages 479–485, 1999.
16. 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.
17. B. Stenger, P.R.S. Mendonça, and R. Cipolla. Model based 3D tracking of an articulated hand. In *CVPR*, volume 2, pages 310–315, 2001.
18. J. Triesch and C. von der Malsburg. Robotic gesture recognition. In *Gesture Workshop*, pages 233–244, 1997.
19. Virtual Technologies, Inc., Palo Alto, CA. *VirtualHand Software Library Reference Manual*, August 1998.
20. Y. Wu and T.S. Huang. View-independent recognition of hand postures. In *CVPR*, volume 2, pages 88–94, 2000.
21. Y. Wu, J.Y. Lin, and T.S. Huang. Capturing natural hand articulation. In *ICCV*, volume 2, pages 426–432, 2001.