Personal and Ubiquitous Computing manuscript No.
(will be inserted by the editor)

A Database-Based Framework for Gesture Recognition

Vassilis Athitsos · Haijing Wang · Alexandra Stefan

Received: date / Accepted: date

Abstract Gestures are an important modality for human-machine communication. Computer vision modules performing gesture recognition can be important components of intelligent homes, assistive environments, and human-computer interfaces. A key problem in recognizing gestures is that the appearance of a gesture can vary widely depending on variables such as the person performing the gesture, or the position and orientation of the camera. This paper presents a database-based approach for addressing this problem. The large variability in appearance among different examples of the same gesture is addressed by creating large gesture databases, that store enough exemplars from each gesture to capture the variability within that gesture. This database-based approach is applied to two gesture recognition problems: handshape categorization and motion-based recognition of American Sign Language (ASL) signs. A key aspect of our approach is the use of database indexing methods, in order to address the challenge of searching large databases without violating the time constraints of an online interactive system, where system response times of over a few seconds are oftentimes considered unacceptable. Our experiments demonstrate the benefits of the proposed database-based framework, and the feasibility of integrating large gesture databases into online interacting systems.

Keywords gesture recognition · hand pose estimation · embeddings · American Sign Language · indexing methods · image and video databases

1 Introduction

Gestures are an important modality for human-machine communication, and robust gesture recognition can be an important component of intelligent homes, assistive environments, and human-computer interfaces in general. A key problem in recognizing gestures is that the appearance of a gesture can vary widely depending on variables such as the person

Vassilis Athitsos, Haijing Wang, Alexandra Stefan
Computer Science and Engineering Department
University of Texas at Arlington
Arlington, Texas, USA
Tel: +1-817-272-0155
Fax: +1-817-272-3784

performing the gesture, or the position and orientation of the camera. For example, the same handshape can look very different in different images, depending on the 3D orientation of the hand and the viewpoint of the camera. Similarly, in the domain of sign language recognition, the appearance of a sign can vary depending on the person performing the sign and the distance from the camera. This paper presents a database-based approach for addressing this problem of large intraclass variability. In the proposed approach, large gesture databases are used, and for each gesture class a large number of exemplars is stored in order to capture the variability among samples of that gesture class.

This database-based framework is applied to two different gesture recognition domains. The first domain is handshape categorization. Handshapes can hold important information about the meaning of the gesture, for example in sign languages, or about the intent of an action, for example in manipulative gestures or in virtual reality interfaces. In our database-based approach, a large database of tens of thousands of images is used to represent the wide variability of handshape appearance. A key advantage of the database approach is that it provides a very natural way to capture the nonparametric distribution that characterizes the appearance of each handshape class. Furthermore, databases containing tens or hundreds of thousands of images can be easily generated overnight using off-the-shelf computer graphics software.

The second gesture recognition domain where we apply the proposed approach is recognition of signs in American Sign Language (ASL). In particular, we consider the problem of searching an ASL dictionary for the meaning of a particular sign. In an automated sign lookup system, when a user wants to look up a specific sign, the user can submit a video example of that sign, and ask the system to identify database videos that are the best matches for that sign. Each database video can be annotated with information about the sign shown in that video, such as meaning, usage, or related signs. A key challenge in such a system is designing the database search module, so that the results returned to the user include the correct sign as often as possible.

Designing an accurate search module is a challenging task, that has to address the fact that the appearance of a sign depends on the person performing the sign, as well as the position and distance of the signer with respect to the camera. To address this issue, we convert the original database of 933 sign exemplars to an extended database of about 270,000 exemplars, by creating multiple copies of each exemplar in the original database, corresponding to different scaling parameters. Our experiments demonstrate that using the extended database improves retrieval accuracy, while still satisfying the time constraints of an online, interactive system.

Efficient and accurate indexing methods are important in the proposed database–based framework. In both the handshape recognition domain and the sign recognition domain, the system must identify, given a query image or video, the most similar entries in the database. The best database matches need to be identified fast enough to allow the system to be used in an interactive environment. At the same time, this database retrieval task can be very challenging, for the following reasons:

- The similarity measures that are most meaningful for image and video matching are often non-Euclidean, nonmetric, and computationally expensive. Examples of such nonmetric distance measures are the chamfer distance [7], shape context matching [9,51], and dynamic time warping (DTW) [31,33].
- The majority of database indexing methods are designed for Euclidean distance measures or metric distance measures (i.e., distance measures that obey the reflexivity, symmetry, and triangle inequality properties). Thus a relatively small number of indexing

methods are available for the nonmetric distance measures typically used for comparing hand images.

With respect to database indexing, the focus of this paper is not on proposing new indexing methods, but rather on determining the feasibility of using existing off-the-shelf indexing methods in our gesture recognition tasks. For that purpose, we consider the recently proposed BoostMap embedding method [3], and integrate that method into the retrieval modules for both our handshape recognition system and the ASL sign lookup system. A key result of our experiments is that BoostMap indeed works well in our domains and offers significant speedups compared to the naive brute-force method of comparing the query to every single database entry.

Overall, the experiments demonstrate the advantages of the proposed database-based framework for gesture recognition. In both our experimental domains, a large database is used to capture naturally the large variability in appearance between examples of the same gesture. While brute-force search in such large databases can prohibitively slow for many applications, existing indexing methods can be used to drastically improve efficiency, and thus make the proposed approach suitable for interactive applications.

2 Related Work

Gesture recognition has been an active area of research for several years. Progress in automatic recognition of gestures is useful for a diverse array of applications in areas including human computer interfaces, surveillance systems, sign language recognition, assistive environments, and healthcare. For example, some recent work proposes gesture recognition interfaces that facilitate human computer interaction for blind persons [50], or disabled persons who have difficulty performing standard motions needed to use a traditional keyboard/mouse interface [30]. As another example, the system described in [32] uses gesture recognition to facilitate the communication of hospital patients who have difficulty speaking.

A large number of methods have been proposed in the literature covering various aspects of gesture recognition. In the remainder of this section we briefly review existing methods for handshape recognition and sign recognition, and we highlight the contrasts between those methods and the solutions we propose in this paper.

Computer vision systems that estimate handshape under arbitrary 3D orientations typically do it in the context of tracking [24,36,42,49,64]. 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. The handshape recognition method described in this paper can be used (among other things) to automate the initialization and error recovery of a hand tracker.

A regression system that estimates hand pose from a single image is described in [43]. However, that method assumes that the hand silhouette is correctly identified in the input image, whereas such precise hand detection is often unrealistic to assume in a real-world application. Another regression method is presented at [16], but that method requires that the hand be simultaneously visible from multiple cameras. The database-based handshape recognition approach described here has the advantage that it only requires a single camera, and it can tolerate a certain amount of imprecision in hand detection; we still require the location of the hand to be given as an input to our system, but we do not require precise separation of the hand silhouette from the background.

Another family of methods for hand shape classification are appearance-based methods, like [19,63]. Such methods are typically limited to estimating 2D hand pose from a limited number of viewpoints. In contrast, our handshape recognition approach can handle arbitrary viewpoints.

With respect to recognition of signs and sign languages, a number of approaches have been proposed in the literature (see [39] for a recent review). Many approaches are not vision-based, but instead use input from magnetic trackers and sensor gloves, e.g., [21,37, 45,57,58,66]. Such methods achieve good recognition results on continuous Chinese Sign Language with vocabularies of about 5,000 signs [21,58,66]. On the other hand, vision-based methods, e.g., [8,13,17,20,29,48,65] use smaller vocabularies (20-300 signs) and often rely on color markers, e.g., [8,17]. The approach described in this paper is a step towards developing vision-based methods that can handle a more comprehensive vocabulary.

A key focus of this paper is on identifying efficient indexing methods for speeding up the task of finding, given a query image or video, the most similar database matches. Various methods have been employed for speeding up nearest neighbor retrieval. Comprehensive reviews on the subject include [10,26,25]. A large amount of work focuses on efficient nearest neighbor retrieval in multidimensional vector spaces using an L_p metric, e.g., [22, 34,52,61]. However, that family of approaches is not applicable in our setting, since the chamfer distance (i.e., the distance measure that we use for comparing hand images) is not an L_p measure.

A number of nearest neighbor methods can be applied for indexing arbitrary metric spaces; the reader is referred to [25] for surveys of such methods. As an example, VP-trees [67] and metric trees [53] hierarchically partition the database into a tree structure by splitting, at each node, the set of objects based on their distances to pivot objects. However, while such methods can offer theoretical guarantees of performance in metric spaces, the chamfer distance and dynamic time warping distance used in our experiments are nonmetric, and so are other measures typically used for comparing images and video to each other, such as shape context matching [9,51], and distance measures based on the Viterbi algorithm [51].

In domains with a computationally expensive distance measure, significant speed-ups can be obtained by embedding objects into another space with a more efficient distance measure. Several methods have been proposed for embedding arbitrary spaces into a Euclidean or pseudo-Euclidean space [3,4,11,18,27,60]. These methods are indeed applicable to our setting. In this paper we focus on the BoostMap embedding method [3] and we show that this method can be successfully employed in both our experimental domains, i.e., handshape recognition and sign recognition. The success of the BoostMap method in both domains illustrates the feasibility and benefits of using off-the-shelf, domain-independent indexing methods for gesture recognition tasks.

The parts of this paper discussing our database-based approach for handshape categorization are based on work that we published in three conference papers [2,6,40]. The parts of the paper describing our database-based approach for automated sign lookup are novel, and have not been published before.

3 Database-based Handshape Recognition

In handshape recognition, the goal is to recognize a set of different handshapes, such as the 20 handshapes shown on Fig. 1. In this section, we describe a system that operates on single images, as opposed to entire video sequences, or images obtained simultaneously for multiple cameras. We need to specify up front that, in a real-world system, reliable

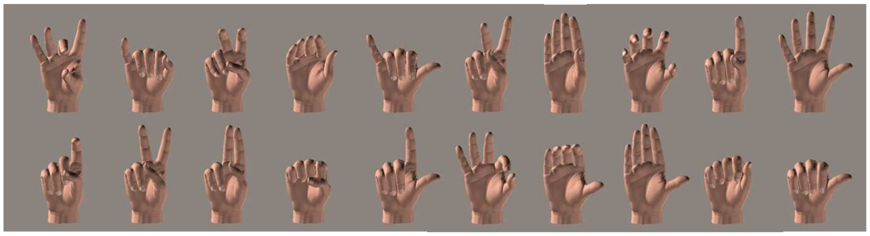

Fig. 1 The 20 handshapes used in the ASL handshape dataset.

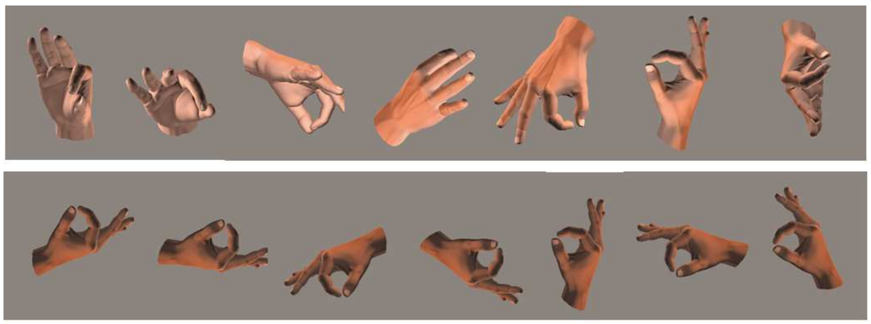

Fig. 2 Examples of different appearance of a fixed 3D hand shape, obtaining by altering camera viewpoint and image plane rotation. Top: the ASL "F" handshape rendered from seven different camera viewpoints. Bottom: the ASL "F" handshape rendered from a specific camera viewpoint, using seven different image plane rotations.

recognition of handshapes of arbitrary 3D orientation from a single image is beyond the current state of the art. At the same time, a system that operates on a single image, even if it has a relatively low classification accuracy, can be immensely useful in identifying a relatively small set of likely hypotheses. Such a set of hypotheses can subsequently be refined:

- using a hand tracker [42,24,36,47,49,64],
- using domain-specific knowledge, such as ASL linguistic constraints, or
- using knowledge of a specific protocol for human-computer communication, that can place constraints on the current handshape based on the current communication context.
- using simultaneous views from multiple cameras.

3.1 A Database of Hand Images

A key challenge in reliable handshape recognition in an intelligent home setting, or an assistive environment setting, is that the same handshape can look very different in different images, depending on the 3D orientation of the hand with respect to the camera (Fig. 2). Using a large database of hand images is a natural way to address this wide variability of the appearance of a single handshape. Since handshape appearance depends on 3D orientation, we can densely sample the space of all possible 3D orientations, and include a database image for every handshape in every one of the sampled 3D orientations.

In our system, we include 20 different handshapes (Fig. 1). Those 20 handshapes are all commonly used in American Sign Language (ASL). For each handshape, we synthetically generate a total of 4,032 database images that correspond to different 3D orientations of the hand. In particular, the 3D orientation depends on the viewpoint, i.e., the camera position on the surface of a viewing sphere centered on the hand, and on the image plane rotation. We sample 84 different viewpoints from the viewing sphere, so that viewpoints are approximately spaced 22.5 degrees apart. We also sample 48 image plane rotations, so that rotations are spaced 7.5 degrees apart. Therefore, the total number of images is 80,640 images, i.e., 20 handshapes $\times$ 84 viewpoints $\times$ 48 image plane rotations. Figure 2 displays example images of a handshape in different viewpoints and different image plane rotations. Each image is normalized to be of size 256×256 pixels, and the hand region in the image is normalized so that the minimum enclosing circle of the hand region is centered at pixel $(128, 128)$, and has radius 120. All database images are generated using computer graphics, and in particular using the Poser 5 software [14]. It takes less than 24 hours to generate these thousands of images. Image generation is a script-based automated process.

3.2 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. Examples of hand images and corresponding edge images are shown on Fig. 3.

The chamfer distance [7] 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 $D(X,Y)$ is:

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

where $\|a - b\|$ denotes the Euclidean distance between two pixel locations a and b. $D(X,Y)$ penalizes for points in either edge image that are far from any point in the other edge image. Fig. 4 shows an illustration of the chamfer distance.

The chamfer distance operates on edge images. The synthetic images generated by Poser can be rendered directly as edge images by the software. For the test images we simply apply the Canny edge detector [12].

On an AMD Athlon processor running at 2.0GHz, we can compute on average 715 chamfer distances per second. Consequently, finding the nearest neighbors of each test image using brute force search, which requires computing the chamfer distances between the test image and each database image, takes about 112 seconds. Taking 112 seconds to match the input image with the database is clearly too long for an interactive application. The need for efficiency motivates our exploration of database indexing methods. In Section 5 we describe how to use an indexing method to speed up the retrieval process.

4 Database-based Sign Recognition

The long-term goal of our work on sign recognition is to design a system that makes it easy for users and learners of American Sign Language (ASL) to look up the meaning of an

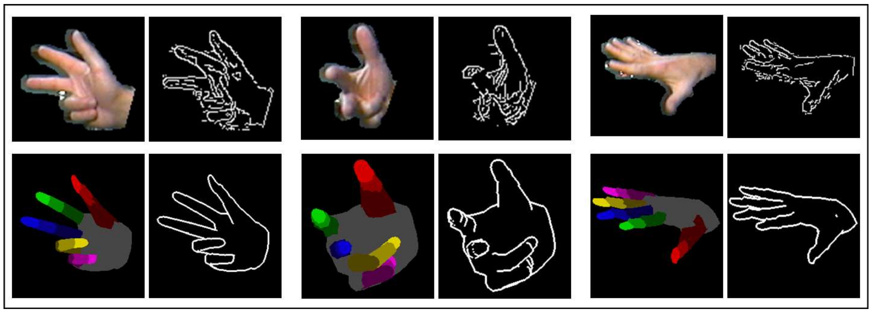

Fig. 3 Examples of real and synthetic hand images and their corresponding edge images.

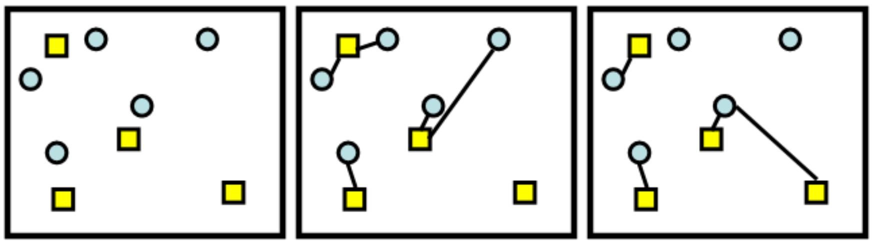

Fig. 4 An example of the chamfer distance. The left image shows two sets of points: points in the first set are shown as circles, and points in the second set are shown a squares. The middle image shows a link between each circle and its closest square. The circle-to-square directed chamfer distance is the average length of those links. The right image shows a link between each square and its closest circle. The square-to-circle chamfer distance is the average length of those links. The chamfer distance (also known as *undirected chamfer distance*) between squares and circles is the sum of the two directed distances.

unknown sign. In such a sign lookup system, when the user encounters an unknown sign, the user submits to the system a video of that sign. The user can submit a pre-existing video, if such a video is available. Alternatively, the user can perform the sign in front of a camera, and generate a query video that way.

A key component of the sign lookup project is data collection. As described in [5], we are in the process of collecting a large video dataset containing examples of almost all of the 3,000 signs contained in the Gallaudet dictionary [54]. Each sign is performed by a native signer.

Due to the large number of signs, we can only collect a small number of exemplars for each sign. The lack of a large number of training examples for sign renders several model-based recognition methods inapplicable, e.g., Hidden Markov Models [41,56]. At the same time, exemplar-based methods are readily applicable in cases with a small number of examples per class. In an exemplar-based method, processing a query involves identifying the most similar matches of the query in a database of training examples.

In our experiments, the database contains 933 examples of signs, corresponding to 921 unique sign classes. Experiments are performed in a user-independent manner, where the people performing signs in the query videos do not appear in the database videos. Figure 5 shows sample frames from four videos from this dataset.

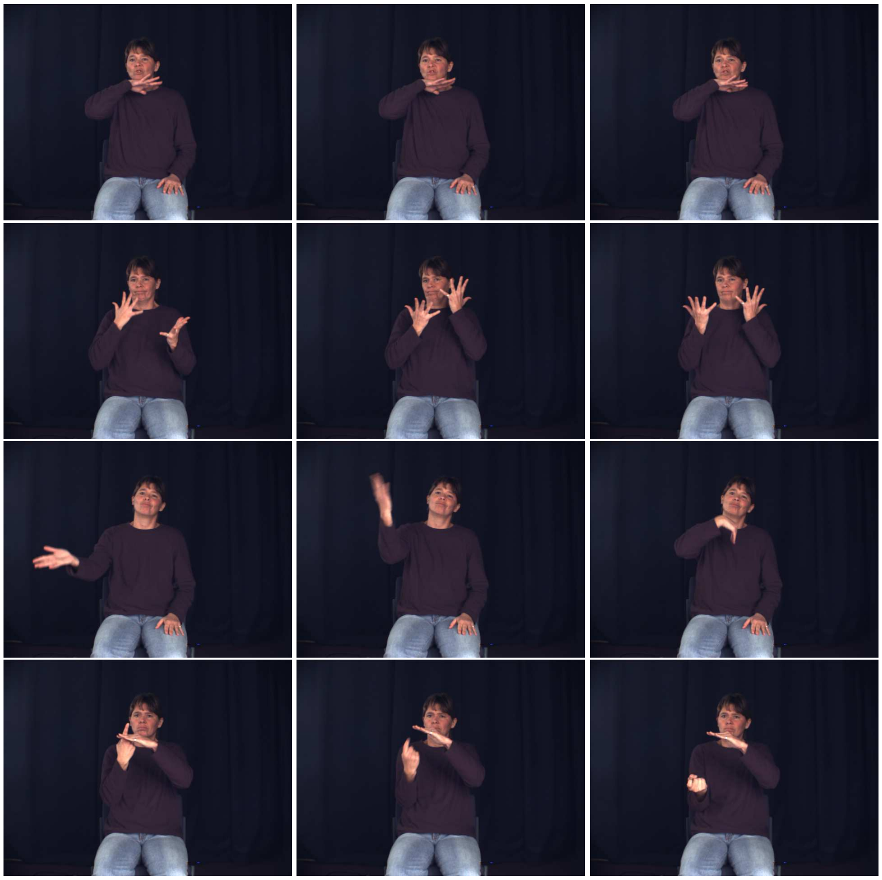

Fig. 5 Examples of sign videos from the ASL lexicon video dataset [5]. For each sign, we show, from left to right, the first frame, a middle frame, and the last frame. First row: an example of the sign DIRTY. Second row: an example of the sign EMBARRASED. Third row: an example of the sign COME-ON. Fourth row: an example of the sign DISAPPEAR.

4.1 Features for Sign Recognition

The meaning of a sign is determined by handshape, hand motion, and hand position with respect to the body. A key challenge in computer vision approaches for sign recognition is the current lack of reliable general-purpose modules for extracting handshape, hand motion, and articulated body pose, that can work in unconstrained, real-world scenes. In order to simplify the task and produce a functioning end-to-end system, in this paper we only use hand motion to discriminate between signs, leaving incorporation of hand appearance and body pose information as future work.

Furthermore, in the system described here we make the simplifying assumption that the system knows the location of the dominant hand in every frame of every database sequence and every query sequence. The location of hands in all database sequences is manually annotated. While this manual annotation is a labor-intensive process, this process is a one-time pre-processing cost that is transparent to the end user. Hand detection in the query sequence can be performed in a semi-automatic way, where the system identifies hand locations using skin and motion information [38], and the user reviews and corrects the results before submitting a query. In the near future we hope to fully automate the query process using methods that can tolerate errors and ambiguities in hand detection, such as dynamic space-time warping (DSTW) [1].

4.2 The Dynamic Time Warping Distance Measure

In order for the system to identify the most similar database matches to a query video, we need to define a distance measure between sign videos. Given the position of the dominant hand in each frame, each sign video is naturally represented as a 2D time series $((x_1,y_1),\ldots,(x_n,y_n))$, where n is the number of frames in the video, and each (x_i,y_i) represents the pixel coordinates of the centroid of the hand in the i-th frame. Consequently, comparing sign videos to each other becomes a time series matching problem.

For the purpose of measuring distance between the time-series representations of signs, we use the dynamic time warping (DTW) distance measure [15,31,33]. DTW is a popular method for matching time series, and satisfies a key requirement for a time series distance measure: the ability to tolerate temporal misalignments, so as to allow for time warps, such as stretching or shrinking a portion of a sequence along the time axis, and differences in length between time series. We now proceed to briefly describe DTW.

Let Q be the time series representation of a query video with $|Q|$ frames, and let X be the time series representation of a database video with $|X|$ frames. A warping path $W = ((w_{1,1},w_{1,2}),\ldots,(w_{|W|,1},w_{|W|,2}))$ defines an alignment between two time series Q and X, and $|W|$ denotes the length of W. The i-th element of W is a pair $(w_{i,1},w_{i,2})$ that specifies a correspondence between element $Q_{w_{i,1}}$ of Q and element $X_{w_{i,2}}$ of X. The cost $C(W)$ of warping path W that we use is the sum of the Euclidean distances between corresponding elements $Q_{w_{i,1}}$ and $X_{w_{i,2}}$:

$$C(W) = \sum_{i=1}^{|W|} \|Q_{w_{i,1}} - X_{w_{i,2}}\| \tag{2}$$

As a reminder, in our setting, $Q_{w_{i,1}}$ and $X_{w_{i,2}}$ denote respectively the center of the dominant hand in frame $w_{i,1}$ of the query video and frame $w_{i,2}$ of the database video.

For W to be a legal warping path, W must satisfy the following constraints:

- **Boundary conditions:** $w_{1,1} = w_{1,2} = 1, w_{|W|,1} = |Q|$ and $w_{|W|,2} = |X|$. This requires the warping path to start by matching the first element of the query with the first element of X, and end by matching the last element of the query with the last element of X.
- **Monotonicity:** $w_{i+1,1} - w_{i,1} \geq 0, w_{i+1,2} - w_{i,2} \geq 0$. This forces the warping path indices $w_{i,1}$ and $w_{i,2}$ to increase monotonically with i.
- **Continuity:** $w_{i+1,1} - w_{i,1} \leq 1, w_{i+1,2} - w_{i,2} \leq 1$. This restricts the warping path indices $w_{i,1}$ and $w_{i,2}$ to never increase by more than 1, so that the warping path does not skip any elements of Q, and also does not skip any elements of X between positions $X_{w_{1,2}}$ and $X_{w_{|W|,2}}$.

The optimal warping path between Q and X is the warping path with the smallest possible cost. The DTW distance between Q and X is the cost of the optimal warping path between Q and X. Given Q and X, the DTW distance between Q and X and the corresponding optimal warping path can be easily computed using dynamic programming [31].

Computing the DTW distance takes time $O(|Q||X|)$, i.e., time proportional to the product of the lengths of the two time series. If Q and X have comparable lengths, computing the DTW distance takes time quadratic to the length of the Q, and thus DTW is a computationally expensive distance measure. Furthermore, DTW is non-metric, as it does not obey the triangle inequality. On an Intel Xeon quad-core E5405 processor, running at 2.0GHz, and using only a single core, we can compute on average about 1000 DTW distances per second, when measuring DTW distances between time series corresponding to query and database videos.

4.3 Tolerating Differences in Translation and Scale

Since the only information we use in measuring sign similarity is hand position, and hand position is *not* translation invariant or scale invariant, we need to take additional steps to ensure that the matching algorithm tolerates differences in translation and scale between two examples of the same sign.

We address differences in translation by normalizing all hand position coordinates based on the location of the face in each frame. Face detection is a relatively easy task in our setting, since we can assume that the signer's face is oriented upright and towards the camera. Mature, publicly-available real-time face detection systems have been available for several years [44, 55], that work well in detecting upright, frontal views of faces. In our experiments, the face location in database sequences is manually annotated, whereas for query sequences we use the publicly available face detector developed by Rowley, et al. at CMU [44].

Differences in scale can also cause problems, as a small difference in scale can lead to large differences in hand positions, and consequently to large DTW distances. Our approach for tolerating differences in scale is to artificially enlarge the database, by creating for each database sign multiple copies, each copy corresponding to different scaling parameters. We should note that each of these multiple copies is not a new sign video, but simply a new time series, and thus the storage space required for these multiple copies is not significant.

In particular, for each time series corresponding to a database sign video, we generate 289 scaled copies. Each scaled copy is produced by choosing two scaling parameters S_x and S_y, that determine respectively how to scale along the x axis and the y axis. Each S_x and S_y can take 17 different values, spaced uniformly between 0.92 and 1.08, thus leading to a total of $17^2 = 289$ possible value for each (S_x, S_y) pair.

While, as mentioned earlier, the space required for storing these multiple copies is not significant, the time required to exhaustively compare the query to the entire database is severely affected. Since in our current system we can compute about 1000 DTW distances per second, exhaustively matching the query with the 933 entries in the original database (i.e., without including the scaled copies) takes on average a bit less than a second. On the other hand, exhaustively matching the query with the 269,637 entries in the extended database (i.e., including the scaled copies) takes on average over four minutes, which is too long for an interactive application. Fortunately, we can use existing database indexing methods, as described in the next section, to significantly reduce the search time and allow interactive use.

5 Embedding-based Retrieval

In our example applications, calculating the chamfer distance or the DTW distance between the query and all database examples takes too long (almost two minutes in the handshape recognition system, over four minutes in the sign lookup system) to be used in interactive applications. However, we can obtain an efficient approximation of these expensive distance measures by embedding query and database objects into a vector space. Using such an embedding we can drastically speed up retrieval time, with relatively small losses in accuracy. In this section we discuss how such embeddings can be constructed.

5.1 Lipschitz Embeddings

Embeddings of arbitrary spaces into a vector space are a general approach for speeding up nearest neighbor retrieval. Let $\mathbb{X}$ be a set of objects, and $D(X_1, X_2)$ be a distance measure between objects $X_1, X_2 \in \mathbb{X}$. For example, in the handshape recognition setting, $\mathbb{X}$ is the set of edge images of hands, and D is the chamfer distance. In the sign lookup setting, $\mathbb{X}$ is the set of time series corresponding to sign videos, and D is the DTW distance. An embedding $F : \mathbb{X} \to \mathbb{R}^d$ is a function that maps objects from $\mathbb{X}$ into the d-dimensional real vector space $\mathbb{R}^d$, where distances are typically measured using an L_p or weighted L_p measure, denoted as D'. Such embeddings are useful when it is computationally expensive to evaluate distances in $\mathbb{X}$, and it is more efficient to map points of $\mathbb{X}$ to vectors and compute some L_p distance between those vectors.

Given an object $X \in \mathbb{X}$, a simple 1D embedding $F^R : \mathbb{X} \to \mathbb{R}$ can be defined as follows:

$$F^R(X) = D(X, R) . \tag{3}$$

The object R that is used to define F^R is typically called a *reference object* or a *vantage object* [26]. A multidimensional embedding $F : \mathbb{X} \to \mathbb{R}^d$ can be constructed by concatenating such 1D embeddings: if $F_1, \ldots, F_d$ are 1D embeddings, we can define a d-dimensional embedding F as $F(X) = (F_1(X), \ldots, F_d(X))$.

The basic intuition behind such 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.

Suppose that we have chosen a set of d database objects $R_1, R_2, ..., R_d$ as reference objects. Then, we can define a function F, mapping X to $\mathbb{R}^d$ as follows:

$$F(X) = (D(X, R_1), D(X, R_2), ..., D(X, R_d)) . \tag{4}$$

The function F turns out to be a special case of Lipschitz embeddings [11, 35]. In our handshape recognition setting, F maps edge images to d-dimensional vectors. In our sign recognition setting, F maps time-series representations of sign videos to d-dimensional vectors.

We define the *approximate distance* D' between two objects X_1 and X_2 to be the L_1 distance between $F(X_1)$ and $F(X_2)$:

$$D'(A,B) = \sum_{i=1}^{d} |D(X_1,R_i) - D(X_2,R_i)| \ . \tag{5}$$

The actual value of $D'(A,B)$ is not necessarily similar in scale to the value $D(A,B)$. However, $D'(A,B)$ is an approximation of $D(A,B)$ in the sense that, when $D(A,B)$ is much smaller than $D(A,G)$, then we also expect $D'(A,B)$ to be smaller than $D'(A,G)$. The intuition is, again, that if A and B are close to each other, then they will also have relatively similar distances to each of the R_i's.

In the handshape recognition domain, the time complexity of computing the approximate distance D' between an edge image X and U database edge images is $O(dn \log n + Ud)$, where n is the max number of edge pixels in any edge image and d is the dimensionality of the embedding. In particular, it takes $O(dn \log n)$ time to compute $F(X)$, i.e., to compute the d chamfer distances between the edge image and each of the d reference objects, and it takes $O(Ud)$ time to compute the L_1 distance between $F(X)$ and the embeddings of all database images (which just need to be precomputed once, off-line, and stored in memory). On the other hand, computing the chamfer distance C between X and all database images takes $O(Un \log n)$ time. The complexity savings are substantial when d is much smaller than U. In our system it takes on average 112 seconds to compute the chamfer distances between the input image and all database images (for test and database images of size 256x256). In contrast, for $d = 100$, it takes 0.14 seconds to compute the corresponding approximate distances D', which is close to three orders of magnitude faster. Similar speedups are obtained in our sign recognition domain by replacing the DTW distance with the corresponding embedding-based approximate distance.

5.2 BoostMap Embeddings

A simple way to define embeddings for our purposes, i.e., for efficient matching of hand images and time series representations of sign videos, is to apply Eq. 4 for some reasonable embedding dimensionality d (values between 20 and 100 typically work well in practice), and using d reference objects R_i chosen randomly from the database. However, we can significantly optimize embedding quality using tools available from the machine learning community. In particular, embedding optimization can be casted as the machine learning problem of optimizing a binary classifier, and boosting methods such as AdaBoost [46] can be employed for embedding optimization. This is the approach taken in the BoostMap method, which is described in [3]. In our experiments we demonstrate that the BoostMap method works well for both our handshape recognition system and our sign lookup system. In this section we briefly summarize the BoostMap method, following the description in [3].

Suppose we have an embedding F with the following property: for any $Q, A, B \in \mathbb{X}$ (where $\mathbb{X}$ in our applications is either the space of edge images of hands, or the space of time series representations of signs), if Q is closer (according to the chamfer distance or DTW) to A than to B, then $F(Q)$ is closer to $F(A)$ than to $F(B)$. We can easily derive that F would also have the following property: for every query object Q, if A is the nearest neighbor of Q in the database, then $F(A)$ is the nearest neighbor of $F(Q)$ among the embeddings of all database objects. Such an embedding would lead to perfectly accurate nearest neighbor retrieval.

Finding such a perfect embedding is usually impossible. However, we can try to construct an embedding that, as much as possible, tries to behave like a perfect embedding. In other words, we want to construct an embedding in a way that maximizes the fraction of triples (Q,A,B) such that, if Q is closer to A than to B, then $F(Q)$ is closer to $F(A)$ than to $F(B)$.

More formally, using an embedding F we can define a classifier $\tilde{F}$, that estimates (sometimes wrongly) for any three objects Q,A,B if Q is closer to A or to B. $\tilde{F}$ is defined as follows:

$$\tilde{F}(Q,A,B) = \|F(Q) - F(B)\|_1 - \|(F(Q) - F(A)\|_1 \ , \tag{6}$$

where $\|X,Y\|_1$ is the L_1 distance between X and Y. A positive value of $\tilde{F}(Q,A,B)$ means that F maps Q closer to A than to B, and can be interpreted as a "prediction" that Q is closer to A than to B in the original space X. If this prediction is always correct, then F perfectly preserves the similarity structure of X.

Simple 1D embeddings, like the one defined in Eq. 3, are expected to behave as *weak classifiers*, i.e. classifiers that may have a high error rate, but at least give answers that are not as bad as random guesses (random guesses are wrong 50% of the time). Given many weak classifiers, a well-studied problem in machine learning is how to combine such classifiers into a single, strong classifier, i.e., a classifier with a low error rate. A popular choice is AdaBoost [46], which has been successfully applied to several domains in recent years.

The BoostMap algorithm [3] uses AdaBoost to construct an embedding. The input to AdaBoost is a large set of randomly picked 1D embeddings (i.e., embeddings defined by applying Eq. 3 using reference objects R picked randomly from our database), and a large set of training triples (Q,A,B) of objects, for which we know if Q is closer to A or to B (closer according to the chamfer distance, or to DTW, in our case). The output of AdaBoost is a classifier $H = \sum_{j=1}^{d} \alpha_j \tilde{F}_j$, where each $\tilde{F}_j$ is the weak classifier associated with a 1D embedding F_j, and each α_j is the weight (corresponding to importance) assigned to that 1D embedding. If AdaBoost has been successful, then H has a low error rate.

Using H, we can easily define a high-dimensional embedding F_{out} and a distance measure D' with the following property: for any triple (Q,A,B), if Q is closer to A than to B, H misclassifies that triple if and only if, according to distance measure D' (i.e., the L_1 distance measure in the embedding space) $F_{\text{out}}(Q)$ is closer to $F_{\text{out}}(B)$ than to $F_{\text{out}}(A)$. We define F_{out} and D' as follows:

$$F_{\text{out}}(x) = (F_1(x), ..., F_d(x)) \ . \tag{7}$$

$$D'(F_{\text{out}}(x), F_{\text{out}}(y)) = \sum_{j=1}^{d} (\alpha_j |F_j(x) - F_j(y)|) \ . \tag{8}$$

It is easy to prove that H and F_{out} fail on the same triples [3]. Therefore, if AdaBoost has successfully produced a classifier H with low error rate, then F_{out} inherits the low error rate of H.

5.3 Filter-and-refine Retrieval

In order to implement an end-to-end retrieval system using BoostMap, we use the well-known filter-and-refine retrieval framework [26], which works as follows:

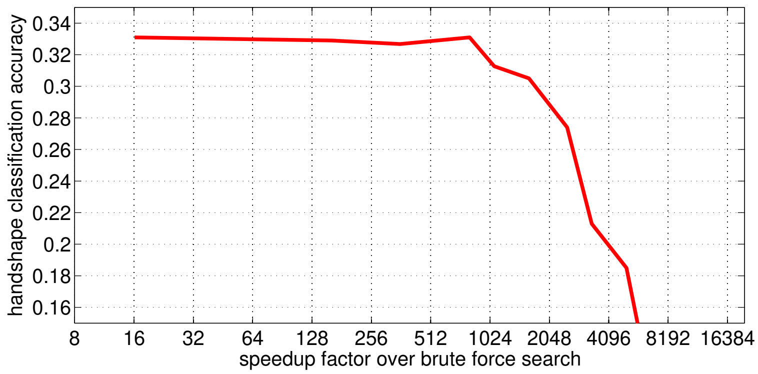

Fig. 6 Classification accuracy vs. speedup attained using BoostMap on the handshape dataset. For each accuracy, the plot shows the corresponding speedup factor obtained using BoostMap. Brute-force nearest neighbor search yields a classification accuracy of 33.1% and an average retrieval time of 112 seconds per query, corresponding to a speedup factor of 1.

- **Offline preprocessing step:** Run the BoostMap algorithm to construct an embedding. Then, compute and store the embeddings of all database objects.
- **Mapping step:** given an input image Q, compute the embedding of Q.
- **Filter step:** identify a small set of candidate nearest neighbors, by comparing $F(Q)$ with the embeddings of all database objects and selecting a small number of database objects whose embeddings are the closest to $F(Q)$.
- **Refine step:** Compute the exact distance between Q and each of the database objects selected during the filter step.
- **Output:** return the database object (among all objects considered in the refine step) with the smallest distance to the input image.

The filter step provides a preliminary set of candidate nearest neighbors in an efficient manner, that avoids computing the exact distance between the query and the vast majority of database objects. The refine step applies the exact distance only to those few candidates. Assuming that the mapping step and the filter step take negligible time (a property that is demonstrated in the experiments), filter-and-refine retrieval is much more efficient than brute-force retrieval.

5.4 Retrieval Complexity

Given a query object Q, the retrieval time for that object is simply the sum of the times that it takes respectively for the mapping step, the filter step, and the refine step. For the mapping step, we need to compute the d-dimensional embedding of Q, which takes $O(d)$ time and requires d distance measurements between the query and reference objects. For the filter step, we need to compare the embedding of the query Q to the embeddings of n database objects, which takes time $O(dn)$. For the refine step, we need to measure p distances between the query and database objects selected during the filter step, which takes $O(p)$ time. Consequently, the retrieval time complexity is $O(dn+p)$.

Measured solely in terms of the size of the database, retrieval takes time $O(n)$, assuming that at the filter step we compare the embedding of the query with the embeddings of all database objects. It is worth noting that, in terms of big-O notation, the complexity of brute force search using the original computationally expensive distance measure is also $O(n)$. However, in terms of actual running time in our experiments, the filter step is at least three orders of magnitude faster than brute-force search using the original distance measure. The factor by which the filter step is faster than brute-force search using the original distance measure is a constant factor, computed as the ratio of the time it takes to measure a single distance under the original distance measure over the time it takes to measure the distance between two vectors in the target space of the embedding.

We should also note that, as d increases, the filter step also becomes more expensive, because we need to compare vectors of increasingly high dimensionality. However, in our experiments so far, with embeddings of up to 1,000 dimensions, the filter step always takes negligible time; retrieval time is dominated by the few exact distance computations we need to perform at the embedding step and the refine step.

In cases (not encountered in our experiments) when the filter step takes up a significant part of retrieval time, one can apply vector indexing techniques [10,28,62] to speed up filtering. We should keep in mind that in the filter step we are finding nearest neighbors in a real vector space, and many indexing methods are applicable in such a setting. One of the advantages of using embeddings is exactly the fact that we map arbitrary spaces to well-understood real vector spaces, for which many tools are available. Using locality sensitive hashing (LSH) [28], for example, the complexity of the filter step can drop from $O(n)$ to $O(\log n)$.

6 Experiments

We evaluate the proposed database-based approach for gesture recognition on two experimental systems: a handshape recognition system, and an ASL sign lookup system.

Performance is evaluated using three measures: retrieval time, K-percentile accuracy, and classification accuracy. These measures are defined as follows:

- **Retrieval time:** average time it takes to process a single query.
- ***K*-percentile accuracy:** fraction of test queries for which the correct class is among the top K-percentile of classes, as ranked by the retrieval system, where K can vary depending on the experiment.
- **Classification accuracy:** fraction of test queries for which the correct class is the highest-ranked class.

In order to compute K-percentile accuracy, we look at the rankings produced by the filter-and-refine algorithm of Section 5.3, and choose for each class its highest-ranking exemplar. We then rank classes according to the rank of the highest-ranking exemplar for each class. For example, suppose that the top three database matches come from class A, the fourth and fifth match come from class B, the sixth match comes from class C, and the seventh match comes from class A again. Then, A is the highest-ranking class, B is the second highest-ranking class, and C is the third highest-ranking class.

Whether K-percentile accuracy or classification accuracy is a more appropriate measure depends on the application. For handshape recognition, there are only 20 classes to be recognized, so classification accuracy is more appropriate. On the other hand, in the sign search system, K-percentile accuracy is a more meaningful measure. Our ASL sign dataset contains 921 sign classes and, given a query, it is not strictly necessary for the correct class to be the highest-ranking class. Including the correct class in the top $K\%$ of classes, for reasonably small values of K (e.g., $K \leq 1\%$), would allow the user to identify the correct class after a quick visual inspection of the highest-ranking results.

6.1 Results on Handshape Recognition

The database of hand images used in the experiments has been constructed as described in Section 3. The test set consists of 710 images. All test images were obtained from video sequences of a native ASL signer either performing individual handshapes in isolation or signing in ASL. The hand locations were extracted from those sequences using the method described in [68]. The test images are obtained from the original frames by extracting the subwindow corresponding to the hand region, and then performing the same normalization that we perform for database images, so that the image size is 256×256 pixels, and the minimum enclosing circle of the hand region is centered at pixel $(128, 128)$, and has radius 120. Examples of test images and their corresponding edge images (edge images are used for the chamfer distance computation) are shown in Figure 3.

For each test image, filter-and-refine retrieval is performed to identify the nearest neighbor of the test image. BoostMap is used for the filter step. The test image is considered to have been classified correctly if the handshape of the nearest neighbor is the same as the handshape of the test image. The ground truth for the test images is manually provided. The total number of handshapes is 20, so our classification task consists of recognizing 20 distinct classes.

Figure 6 illustrates the results obtained on this dataset. An important thing to note here is that the classification accuracy of brute force search (i.e., before we introduce any errors caused by our indexing scheme) is only 33.1%. This accuracy rate reflects the upper limit of how well we can do using our indexing schemes: even if we have an indexing scheme that gives the same results as brute force and achieves enormous speedups, the classification accuracy is still going to be the same as that of brute-force search. At the same time, it is important to note that this accuracy rate is obtained without using any domain-specific constraints, and such constraints are oftentimes available, and highly informative, in concrete real-world applications, as discussed in Section 3.

With respect to the classification performance obtained using BoostMap, we notice that the speedup that we obtain over brute-force search is quite significant: we can get the exact same accuracy rate (33.1%) as with brute-force search, but about 800 times faster. This means that classification time is reduced from 112 seconds per query (using brute-force search) to 0.14 seconds per query. In other words, integrating an indexing scheme into the system drastically improves efficiency, with no loss in accuracy.

Besides embeddings, a simple alternative way to speed up brute-force search is to directly reduce the size of the database, by discarding a certain percentage of database objects. For example, if we only use 10% of the original database objects, brute force search becomes 10 times faster. We have run an experiment evaluating that approach, by using smaller databases of different sizes, obtained by discarding different percentages of objects from the original database. The results are shown on Table 1. The results show that, for the hands dataset, the efficiency gained by reducing the size of the database comes at a significant cost in classification accuracy. Overall, filter-and-refine retrieval using BoostMap embeddings provides far better trade-offs between accuracy and efficiency compared to simply using brute force and reducing the size of the database. For example, with BoostMap we can obtain an accuracy rate of 33.1% with a speedup factor of 800 over brute-force search. If we reduce the database size by a factor of 800, we obtain the same speedup factor of 800, but

Reduction factor	Accuracy rate
1	33.1
2	31.7
3	32.5
4	30.0
6	27.2
8	21.1
10	22.5
16	22.4
32	19.1
64	16.8
100	14.7
128	15.4
256	14.5
512	12.1
800	9.4

Table 1 Classification error rates obtained by using smaller databases. The left column shows the factor by which database size is reduced compared to the original database of 80,640 hand images. The right column shows the obtained classification accuracy rates for that database size. For comparison, using BoostMap we attain an accuracy rate of 33.1% for a speedup factor of 800.

the accuracy drops drastically to 9.4%. Even if we only reduce the size of the database by a factor of 8, the accuracy drops from 33.1% to 21.1%.

The experiments with reduced database sizes also show that, in certain cases, a smaller database size leads to slightly better results than a larger database size. For example, using one eighth of the original database objects the accuracy is 21.1%, and using one sixteenth of the original database objects the accuracy is 22.4%. Given that the test size is 710 images, it is not clear whether such small increases in accuracy are accidental artifacts or whether some database objects actually act as distractors and hurt classification accuracy. The condensing method [23] could be used, in theory, to identify such distractors.

Overall, the experiments show the need for more research, to design image matching methods that are more accurate that the chamfer distance (some recent progress on that topic is reported in [59]). At the same time, the experiments also illustrate the effectiveness of BoostMap as an indexing method. BoostMap yields a classification time that is almost three orders of magnitude faster than that of brute-force search, thus making it feasible to search a large database of hand images in real time.

6.1.1 Discussion of Handshape Recognition Results

The handshape recognition accuracy that we report in the system is clearly not sufficiently high for deployment as a standalone module in unconstrained real-world environments. At the same time, it is important to note that handshape recognition in cluttered images under arbitrary 3D orientation is still a largely unsolved problem. To the best of our knowledge, so far no competing methods have been quantitatively evaluated on real hand images for the task of handshape recognition under arbitrary 3D orientation.

Furthermore, we believe that the handshape recognition rates we report correspond, in some sense, to a worst-case scenario, where no prior information is available as to what 3D orientations and handshapes are most likely to be observed. As discussed in Section 3, our system can be a useful module in a larger hand tracking or gesture recognition system, by identifying a relatively small number of initial hypotheses, that can further be refined using

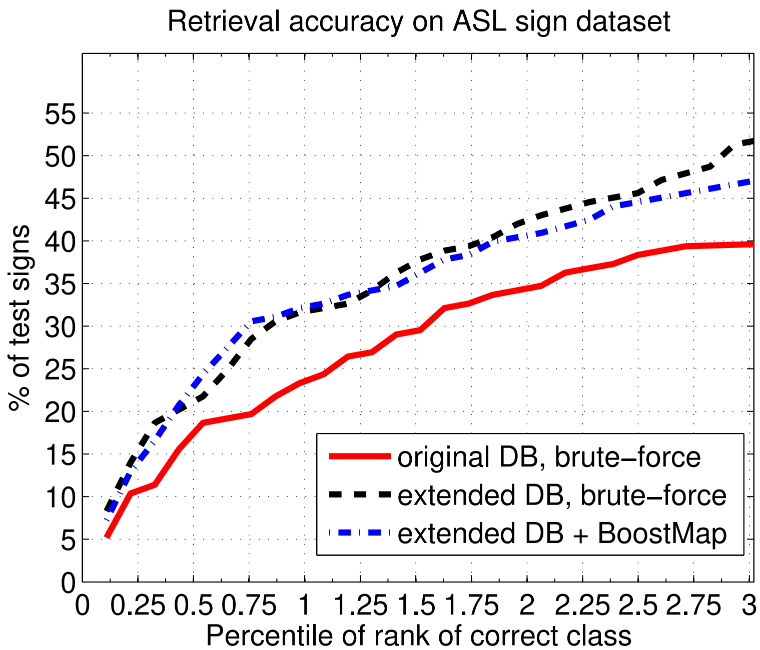

Fig. 7 *K*-percentile accuracy plot for the ASL sign dataset, for brute-force search in the original database of 933 time series, brute-force search in the extended database of 269,637 time series, and embedding-based retrieval in the extended database. The x-axis corresponds to values of *K*, between 0 and 3%. For each such value of *K*, we show the percentage of test signs for which the correct sign class was ranked in the highest *K*-percentile among all 921 classes. For example, using embedding-based retrieval in the extended database, for 32.6% of the queries the correct class was ranked in the top 1.1% of all classes, i.e., in the top 10 out of all 921 classes.

domain-specific knowledge, information from multiple consecutive frames, or information from multiple cameras.

6.2 Results on ASL Sign Retrieval

The query and database videos for these experiments have been obtained from the ASL Lexicon Video Dataset [5]. Our test set consists of 193 sign videos, with all signs performed by two native ASL signers. The video database contains 933 sign videos, corresponding to 921 unique sign classes (we had two videos for a few of the sign classes). The database signs were performed also by a native ASL signer, who was different from the signers performing in the test videos.

Each query and database video was converted to a time series, as described in Section 4.1. From the original database of 933 time series we created an extended database of 269,637 time series, by creating multiple scaled copies of each original time series, as described in Section 4.3.

Figure 7 illustrates the retrieval accuracy obtained on this dataset using brute-force search on the original database of 933 time series, using brute-force search on the extended database of 269,637 time series, and using filter-and-refine retrieval (with BoostMap used

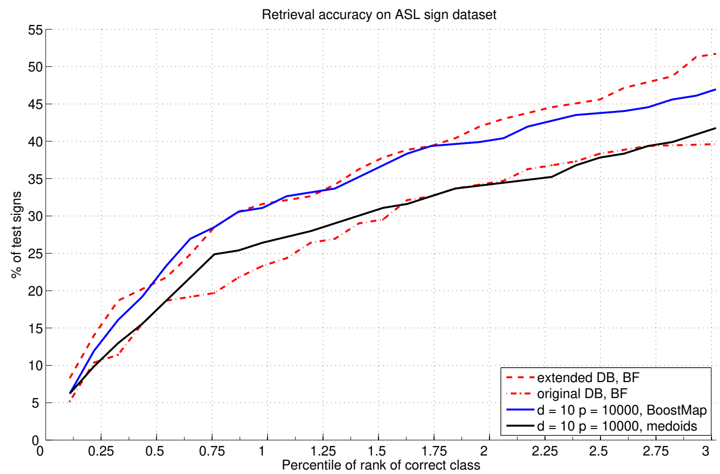

Fig. 8 *K*-percentile accuracy plot for the ASL sign dataset, for brute-force search in the original database of 933 time series, brute-force search in the extended database of 269,637 time series, and embedding-based retrieval in the extended database with two 10-dimensional embeddings, with filter-and-refine parameter p set to 10,000. The first of the two 10-dimensional embeddings was trained using BoostMap, and the second one was defined using as reference objects 10 medoids, identified using an iterative K-medoid algorithm. The x-axis corresponds to values of K, between 0 and 3%. For each such value of K, we show the percentage of test signs for which the correct sign class was ranked in the highest K-percentile among all 921 classes.

in the filter step), with a 100-dimensional embedding, and a refine step that compares the query to the top 10,000 matches obtained from the filter step.

In Figure 7 we focus on K-percentile accuracy with K up to 3%. Our rationale is that if, for a query, the correct class is not ranked in the top 3% of all classes, the retrieval result is unlikely to be useful to the user, because it is unlikely that the user will be willing to visually inspect that many retrieval results in order to identify the correct match. In our current database of 921 sign classes, the top 3% corresponds to 28 classes. When, as is our goal [5], the database is extended to include almost all of the 3,000 signs included in the Gallaudet dictionary [54], the top 3% of all classes will correspond to 90 classes, which will be rather cumbersome for a user to visually inspect.

As Figure 7 shows, extending the database with multiple scaled copies of each time series improves accuracy significantly. For example, as shown in the figure, using brute force search in both the original and the extended database, we obtain the following results: the fraction of test signs for which the correct class is ranked in the top 1.1% of all classes (i.e., in the top 10 out of 921 classes) is 24.4% using the original database and 32.1% using the extended database. Similarly, the fraction of test signs for which the correct class is ranked in the top 2.2% of all classes (i.e., in the top 20 out of 921 classes) is 36.3% using the original database and 43.8% using the extended database.

At the same time, as mentioned in Section 4.2, our current system can compute about 1000 DTW distances per second. Therefore, brute-force search on the original database takes on average a bit less than a second per query, whereas on the extended database it takes on average more than four minutes per query. Here is where incorporating an indexing method can make a big difference. In Figure 7 we include results obtained using embedding-

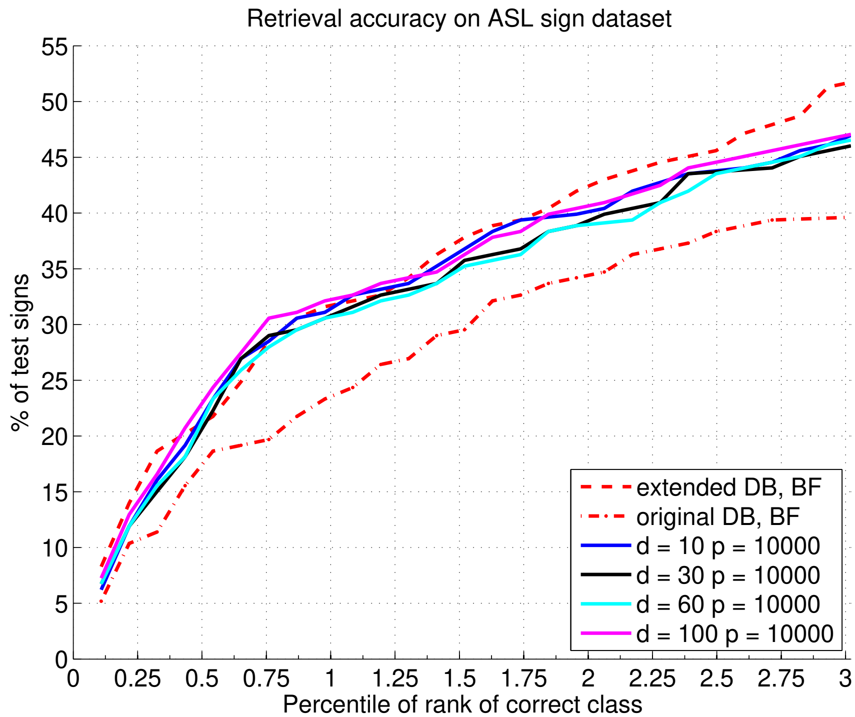

Fig. 9 *K*-percentile accuracy plot for the ASL sign dataset, for brute-force search in the original database of 933 time series, brute-force search in the extended database of 269,637 time series, and embedding-based retrieval in the extended database with embeddings of different dimensionality, with filter-and-refine parameter *p* set to 10,000. The x-axis corresponds to values of *K*, between 0 and 3%. For each such value of *K*, we show the percentage of test signs for which the correct sign class was ranked in the highest *K*-percentile among all 921 classes.

based indexing on the extended database. In particular, we use a 100-dimensional embedding, and the refine step evaluates DTW distances between the query and the top 10,000 matches identified using the embedding. In total, embedding-based retrieval evaluates 100 DTW distances to compute the embedding of the query, and 10,000 DTW distances during the refine step, thus reducing retrieval runtime per query from over 4 minutes to about 10 seconds. We believe that a retrieval time of 10 seconds, while leaving room for improvement, is still within acceptable limits for an online interactive system.

In evaluating a similarity indexing method, a key question is how much accuracy is lost by using indexing instead of brute-force search. Figure 7 shows that, with respect to *K*-percentile accuracy, for *K* values ranging between 0.4% and 1.3%, embedding-based retrieval is not only faster but also more accurate than brute-force search. Given that our test set size is only 193 sign videos, the slightly improved accuracy may well be accidental, as our method only aims to get close to the accuracy of brute-force search, and not to surpass that accuracy. For *K* ranging between 0 and 2.5%, the difference in accuracy between embedding-based retrieval and brute force is rather small. The difference becomes more pronounced for *K* ranging between 2.5% and 3%, but those are ranges in which the system becomes increasingly less useful to the user; arguably, in an extended search lookup system

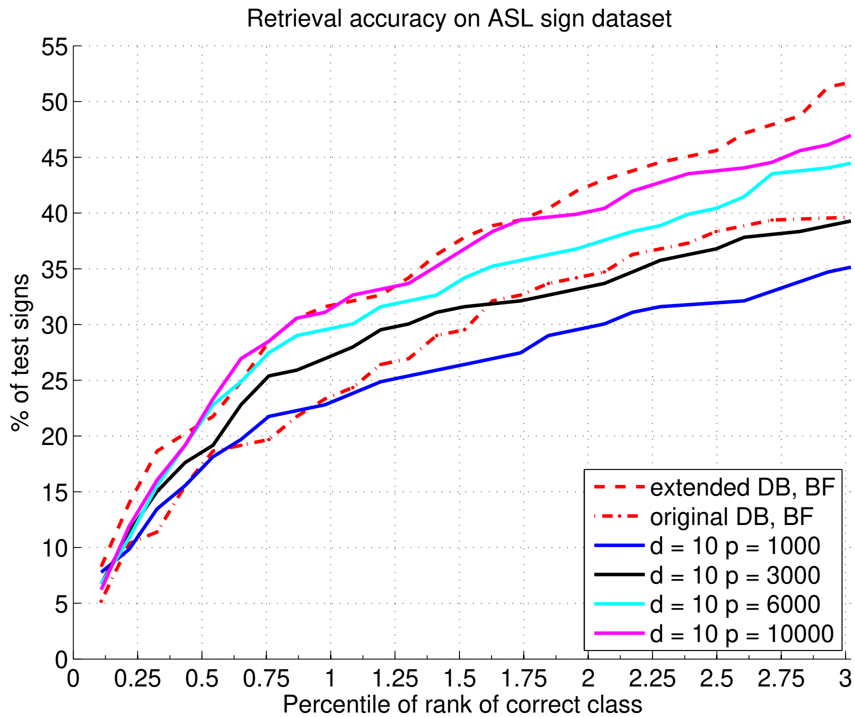

Fig. 10 *K*-percentile accuracy plot for the ASL sign dataset, for brute-force search in the original database of 933 time series, brute-force search in the extended database of 269,637 time series, and embedding-based retrieval in the extended database with 10-dimensional embeddings, for different values of the filter-and-refine parameter p. The x-axis corresponds to values of K, between 0 and 3%. For each such value of K, we show the percentage of test signs for which the correct sign class was ranked in the highest K-percentile among all 921 classes.

covering the 3000 sign classes of the the Gallaudet dictionary [54], the most important values of K for measuring K-percentile accuracy are in the range between 0 and 1%. In that range, embedding-based retrieval works quite well in our experiments.

In Figure 8 we compare performance obtained using a 10-dimensional BoostMap embedding versus performance obtained using a 10-dimensional embedding where the reference objects were selected to be the 10 medoids (among all database objects) identified using a standard iterative K-medoid algorithm. We see that the BoostMap embedding, where the reference objects were selected using AdaBoost, significantly outperforms the embedding that uses medoids.

It is also interesting to see how performance depends on system parameters, namely the dimensionality d of the embedding, and the number p of distance evaluations at the refine step of the retrieval process. Figures 9 and 10 show how K-percentile accuracy varies vs. d and vs. p respectively. It is interesting to note that there are no major differences in performance between embeddings of dimensions 10, 30, 60, and 100. This indicates that increasing the dimensionality above 100 is not likely to improve performance. At the same time, as expected, we see that varying p between 1000 and 10000 drastically affects the obtained K-percentile accuracy, with higher values of p leading to better accuracy, which naturally comes at the cost of slower retrieval time.

6.2.1 Discussion of Sign Retrieval Results

As in the handshape recognition results, we note that the results we have obtained on the sign retrieval system are not at a level that would make the system ready for deployment as a standalone module. At the same time, we believe that the results we have obtained are quite promising, especially given that we only use hand motion information, ignoring hand appearance and position with respect to other parts of the face and torso. Also, we have obtained the results using dynamic time warping, a relatively simple similarity measure, that considers only correspondences between frames, and does not take into account higher-level information like motion pattern over multiple frames, or repeated patterns of motion. We believe that extracting and using additional information from the videos, as well as using more sophisticated similarity measures, can significantly improve accuracy.

It is worth noting that, even with this relatively simple system, for about 32% of the queries, the system ranks the correct result within the top 1% of all classes. While visually inspecting 1% of all signs can be somewhat cumbersome (1% would correspond to 30 out of the 3000 signs in the Gallaudet dictionary [54]), it would still be an acceptable cost for many users, given the current lack of straightforward methods for looking up the meaning of a sign. In that sense, we believe that our current, relatively simple system, still works reasonably well for about one third of the queries, and we hope to make that fraction significantly higher as we continue working towards improving the system.

7 Discussion and Conclusions

This paper has presented a database-based framework for gesture recognition in the context of human computer interaction in real-world applications. We have shown that using large databases of exemplars is a feasible and promising method for capturing the wide range of variability in the appearance of each individual gesture class. We have described in detail how to apply the proposed framework on two specific gesture recognition domains: a handshape recognition system and an ASL sign retrieval system.

A key issue that this paper has addressed is the ability to search large gesture databases fast enough for interactive applications, given the large number of database objects that need to be matched with each query. We have described how to apply BoostMap, an embedding-based indexing method, in order to achieve efficient retrieval in both our applications. Our experiments demonstrate that BoostMap is an effective indexing method, that reduces retrieval time by more than an order of magnitude in both replications, thus allowing retrieval to be performed at interactive speeds. Furthermore, we have shown that the drastic improvements in running time obtained using BoostMap incur only small decreases in recognition accuracy.

While the accuracy rates we have attained in our experiments are still not quite satisfactory, it is important to note that our database-based approach has produced quantitative results based on real datasets, both for handshape recognition under arbitrary 3D orientation, and for large vocabulary sign retrieval. The ability to tackle these hard gesture recognition problems and to produce quantitative results is a key advantage of the proposed database-based framework, where a large database can naturally capture the wide variations of the gestures we want to recognize. The challenge remains to build on top of our results, so as to create gesture recognition systems that are ready for real-world deployment, and that address real user needs, such as the ability to look up the meaning of an unknown ASL sign, or the ability to help disabled persons interact with a computer or communicate with other people. We hope to address that challenge in our ongoing and future work.

Acknowledgements This work has been supported by National Science Foundation grants IIS-0705749 and IIS-0812601, as well as by a University of Texas at Arlington startup grant to Professor Athitsos, and University of Texas at Arlington STARS awards to Professors Chris Ding and Fillia Makedon. We also acknowledge and thank our collaborators at Boston University, including Carol Neidle, Stan Sclaroff, Joan Nash, Ashwin Thangali, and Quan Yuan, for their contributions in collecting and annotating the American Sign Language Lexicon Video Dataset.

References

1. J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff. Simultaneous localization and recognition of dynamic hand gestures. In *IEEE Motion Workshop*, pages 254–260, 2005.
2. V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. Filtering methods for similarity-based multimedia retrieval. In *International Workshop on Audio-Visual Content and Information Visualization in Digital Libraries (AVIVDiLib)*, 2005.
3. V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. Boostmap: An embedding method for efficient nearest neighbor retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(1):89–104, 2008.
4. V. Athitsos, M. Hadjieleftheriou, G. Kollios, and S. Sclaroff. Query-sensitive embeddings. *ACM Transactions on Database Systems (TODS)*, 32(2), 2007.
5. V. Athitsos, C. Neidle, S. Sclaroff, J. Nash, A. Stefan, Q. Yuan, and A. Thangali. The American Sign Language lexicon video dataset. In *IEEE Workshop on Computer Vision and Pattern Recognition for Human Communicative Behavior Analysis (CVPR4HB)*, 2008.
6. V. Athitsos and S. Sclaroff. Estimating hand pose from a cluttered image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 432–439, 2003.
7. 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 *International Joint Conference on Artificial Intelligence*, pages 659–663, 1977.
8. B. Bauer and K.F. Kraiss. Towards an automatic sign language recognition system using subunits. In *Gesture Workshop*, pages 64–75, 2001.
9. S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.
10. C. Böhm, S. Berchtold, and D. A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3):322–373, 2001.
11. J. Bourgain. On Lipschitz embeddings of finite metric spaces in Hilbert space. *Israel Journal of Mathematics*, 52:46–52, 1985.
12. J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.
13. Y. Cui and J. Weng. Appearance-based hand sign recognition from intensity image sequences. *Computer Vision and Image Understanding*, 78(2):157–176, 2000.
14. Curious Labs, Santa Cruz, CA. *Poser 5 Reference Manual*, August 2002.
15. T.J. Darrell, I.A. Essa, and A.P. Pentland. Task-specific gesture analysis in real-time using interpolated views. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 18(12):1236–1242, 1996.
16. T. E. de Campos and D. W. Murray. Regression-based hand pose estimation from multiple cameras. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 782–789, 2006.
17. J. Deng and H.-T. Tsui. A PCA/MDA scheme for hand posture recognition. In *Automatic Face and Gesture Recognition*, pages 294–299, 2002.
18. C. Faloutsos and K. I. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *ACM International Conference on Management of Data (SIGMOD)*, pages 163–174, 1995.
19. W.T. Freeman and M. Roth. Computer vision for computer games. In *Automatic Face and Gesture Recognition*, pages 100–105, 1996.

20. K. Fujimura and X. Liu. Sign recognition using depth image streams. In *Automatic Face and Gesture Recognition*, pages 381–386, 2006.
21. W. Gao, G. Fang, D. Zhao, and Y. Chen. Transition movement models for large vocabulary continuous sign language recognition. In *Automatic Face and Gesture Recognition*, pages 553–558, 2004.
22. Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *International Conference on Very Large Databases*, pages 518–529, 1999.
23. P. E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(3):515–516, 1968.
24. T. Heap and D. Hogg. Towards 3D hand tracking using a deformable model. In *Automatic Face and Gesture Recognition*, pages 140–145, 1996.
25. G. R. Hjaltason and H. Samet. Index-driven similarity search in metric spaces. *ACM Transactions on Database Systems*, 28(4):517–580, 2003.
26. G.R. Hjaltason and H. Samet. Properties of embedding methods for similarity searching in metric spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):530–549, 2003.
27. G. Hristescu and M. Farach-Colton. Cluster-preserving embedding of proteins. Technical Report 99-50, CS Department, Rutgers University, 1999.
28. P. Indyk. *High-dimensional Computational Geometry*. PhD thesis, Stanford University, 2000.
29. T. Kadir, R. Bowden, E. Ong, and A. Zisserman. Minimal training, large lexicon, unconstrained sign language recognition. In *British Machine Vision Conference (BMVC)*, volume 2, pages 939–948, 2004.
30. M. Kavakli. Gesture recognition in virtual reality. *International Journal of Arts and Technology (IJART)*, 1(2):215–229, 2008.
31. E. Keogh. Exact indexing of dynamic time warping. In *International Conference on Very Large Data Bases*, pages 406–417, 2002.
32. C. Keskin, K. Balci, O. Aran, B. Sankur, and L. Akarun. A multimodal 3d healthcare communication system. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, pages 1–4, 2007.
33. J. B. Kruskall and M. Liberman. The symmetric time warping algorithm: From continuous to discrete. In *Time Warps*. Addison-Wesley, 1983.
34. C. Li, E. Chang, H. Garcia-Molina, and G. Wiederhold. Clustering for approximate similarity search in high-dimensional spaces. *IEEE Transactions on Knowledge and Data Engineering*, 14(4):792–808, 2002.
35. 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.
36. S. Lu, D. Metaxas, D. Samaras, and J. Oliensis. Using multiple cues for hand tracking and model refinement. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 443–450, 2003.
37. J. Ma, W. Gao, J. Wu, and C. Wang. A continuous Chinese Sign Language recognition system. In *Automatic Face and Gesture Recognition*, pages 428–433, 2000.
38. J. Martin, V. Devin, and J.L. Crowley. Active hand tracking. In *Automatic Face and Gesture Recognition*, pages 573–578, 1998.
39. S. C. W. Ong and S. Ranganath. Automatic sign language analysis: A survey and the future beyond lexical meaning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):873–891, 2005.
40. M. Potamias and V. Athitsos. Nearest neighbor search methods for handshape recognition. In *Conference on Pervasive Technologies Related to Assistive Environments (PETRA)*, 2008.
41. L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77:2, 1989.
42. 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.
43. R. Rosales, V. Athitsos, L. Sigal, and S. Sclaroff. 3D hand pose reconstruction using specialized mappings. In *IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 378–385, 2001.
44. H.A. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 38–44, 1998.
45. H. Sagawa and M. Takeuchi. A method for recognizing a sequence of sign language words represented in a Japanese Sign Language sentence. In *Automatic Face and Gesture Recognition*, pages 434–439, 2000.
46. R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.
47. 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.

48. T. Starner and A. Pentland. Real-time American Sign Language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, 1998.
49. B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1372–1384, 2006.
50. I. Sturm, M. Schiewe, W. Köhlmann, and H. Jürgensen. Communicating through gestures without visual feedback. In *Conference on Pervasive Technologies Related to Assistive Environments (PETRA)*, 2009.
51. A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 127–133, 2003.
52. E. Tuncel, H. Ferhatosmanoglu, and K. Rose. VQ-index: An index structure for similarity searching in multimedia databases. In *Proc. of ACM Multimedia*, pages 543–552, 2002.
53. J. Uhlman. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, 40(4):175–179, 1991.
54. C. Valli, editor. *The Gallaudet Dictionary of American Sign Language*. Gallaudet U. Press, Washington, DC, 2006.
55. P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, 2001.
56. C. Vogler and D. N. Metaxas. Parallel hidden markov models for american sign language recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pages 116–122, 1999.
57. C. Vogler and D. N. Metaxas. Handshapes and movements: Multiple-channel american sign language recognition. In *Gesture Workshop*, pages 247–258, 2003.
58. C. Wang, S. Shan, and W. Gao. An approach based on phonemes to large vocabulary Chinese Sign Language recognition. In *Automatic Face and Gesture Recognition*, pages 411–416, 2002.
59. J. Wang, V. Athitsos, S. Sclaroff, and M. Betke. Detecting objects of variable shape structure with hidden state shape models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(3):477–492, 2008.
60. X. Wang, J. T. L. Wang, K. I. Lin, D. Shasha, B. A. Shapiro, and K. Zhang. An index structure for data mining and clustering. *Knowledge and Information Systems*, 2(2):161–184, 2000.
61. R. Weber and K. Böhm. Trading quality for time with nearest-neighbor search. In *International Conference on Extending Database Technology: Advances in Database Technology*, pages 21–35, 2000.
62. D. A. White and R. Jain. Similarity indexing: Algorithms and performance. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 62–73, 1996.
63. Y. Wu and T.S. Huang. View-independent recognition of hand postures. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 88–94, 2000.
64. Y. Wu, J.Y. Lin, and T.S. Huang. Capturing natural hand articulation. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 426–432, 2001.
65. M. Yang and N. Ahuja. Recognizing hand gesture using motion trajectories. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 466–472, 1999.
66. G. Yao, H. Yao, X. Liu, and F. Jiang. Real time large vocabulary continuous sign language recognition based on OP/Viterbi algorithm. In *International Conference on Pattern Recognition*, volume 3, pages 312–315, 2006.
67. P.N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 311–321, 1993.
68. Q. Yuan, S. Sclaroff, and V. Athitsos. Automatic 2D hand tracking in video sequences. In *IEEE Workshop on Applications of Computer Vision*, pages 250–256, 2005.