A Unified Framework for Gesture Recognition and Spatiotemporal Gesture Segmentation

Jonathan Alon, *Member, IEEE,* Vassilis Athitsos, *Member, IEEE,* Quan Yuan, *Member, IEEE,* and Stan Sclaroff, *Senior Member, IEEE*

Abstract—Within the context of hand gesture recognition, spatiotemporal gesture segmentation is the task of determining, in a video sequence, where the gesturing hand is located, and when the gesture starts and ends. Existing gesture recognition methods typically assume either known spatial segmentation or known temporal segmentation, or both. This paper introduces a unified framework for simultaneously performing spatial segmentation, temporal segmentation and recognition. In the proposed framework, information flows both bottom-up and top-down. A gesture can be recognized even when the hand location is highly ambiguous and when information about when the gesture begins and ends is unavailable. Thus, the method can be applied to continuous image streams where gestures are performed in front of moving, cluttered backgrounds. The proposed method consists of three novel contributions: a spatiotemporal matching algorithm that can accommodate multiple candidate hand detections in every frame, a classifier-based pruning framework that enables accurate and early rejection of poor matches to gesture models, and a subgesture reasoning algorithm that learns which gesture models can falsely match parts of other longer gestures. The performance of the approach is evaluated on two challenging applications: recognition of hand-signed digits gestured by users wearing short sleeved shirts, in front of a cluttered background, and retrieval of occurrences of signs of interest in a video database containing continuous, unsegmented signing in American Sign Language (ASL).

Index Terms—Gesture recognition, gesture spotting, human motion analysis, dynamic time warping, continuous dynamic programming.

1 Introduction

A general problem in computer vision is building robust systems, where higher-level modules can tolerate inaccuracies and ambiguous results from lower-level modules. This problem arises naturally in the context of gesture recognition: existing recognition methods typically require as input (presumably obtained from faultless lower-level modules) either the location of the gesturing hands or the start and end frame of each gesture. Requiring such input to be available is often unrealistic, thus making it difficult to deploy gesture recognition systems in many real-world scenarios. In natural settings, hand locations can be ambiguous in the presence of clutter, background motion and presence of other people or skin-colored objects. Furthermore, gestures typically appear within a continuous stream of motion, and automatically detecting where a gesture starts and ends is a challenging problem (Fig. 1).

To recognize manual gestures in video, an end-to-end computer vision system must perform both spatial and temporal gesture segmentation. Spatial gesture segmentation is the problem of determining *where* the gesture occurs, i.e., where the gesturing hand(s) are located in each video frame. Temporal gesture segmentation is the problem of determining *when* the gesture starts and ends. This paper proposes a unified framework that accomplishes all three tasks of spatial segmentation, temporal segmentation, and gesture recognition, by integrating information from lower-level vision modules and higher-level models of individual gesture classes and the relations between such classes.

Instead of assuming unambiguous and correct hand detection at each frame, the proposed algorithm makes the much milder assumption that a set of several candidate hand locations has been identified at each frame. Temporal segmentation is completely handled by the algorithm, requiring no additional input from lower-level modules. In the proposed framework, information flows both bottom-up and top-down. In the bottom-up direction, multiple candidate hand locations are detected and their features are fed into the higher-level video-to-model matching algorithm. In the top-down direction, information from the model is used in the matching algorithm to select, among the exponentially many possible sequences of hand locations, a single optimal sequence. This sequence specifies the hand location at each frame, thus completing the low-level task of hand detection.

Within the context of the proposed framework, this paper makes two additional contributions, by addressing two important issues that arise by not requiring spatiotemporal segmentation to be given as input:

- Matching feature vectors to gesture models using dynamic programming requires more computational time when we allow multiple candidate

J. Alon, Q. Yuan, and S. Sclaroff are with the Department of Computer Science, Boston University, 111 Cummington Street, Boston, MA, 02215.
V. Athitsos is with the Department of Computer Science and Engineering, University of Texas at Arlington, 416 Yates Street, Arlington, TX 76019.

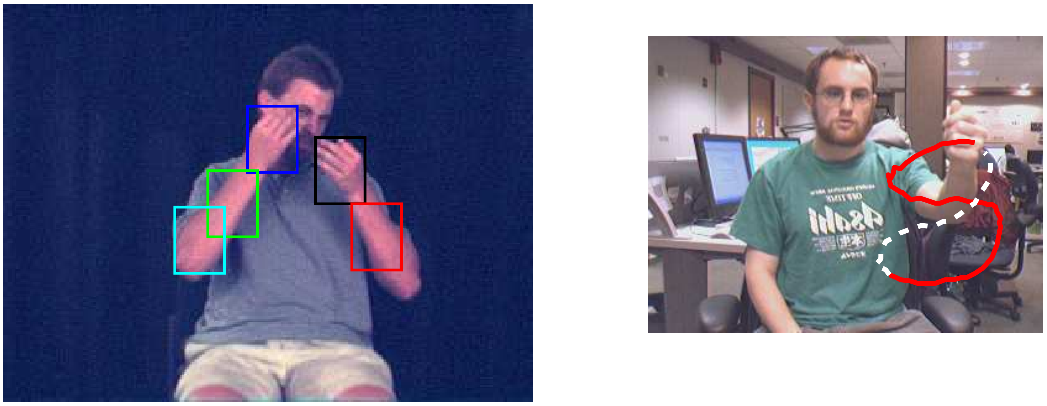


Fig. 1. Illustration of two key problems addressed in this paper. Left: low-level hand detection can often fail to provide unambiguous results. In this example, skin color and motion were used to identify candidate hand locations. Right: an example of the need for subgesture reasoning. Digit “5” is similar to a subgesture of the digit “8”. Without modeling this relation, and by forgoing the assumption that temporal segmentation is known, the system is prone to make mistakes in such cases, falsely matching the model for “5” with a subgesture of “8”.

hand locations at each frame. To improve efficiency, we introduce a discriminative method that speeds up dynamic programming by eliminating a large number of hypotheses from consideration.

- Not knowing when a gesture starts and ends makes it necessary to deal with the *subgesture problem*, i.e., the fact that some gestures may be very similar to subgestures of other gestures (Fig. 1). In our approach, subgesture relations are automatically learned from training data, and then those relations are used for reasoning about how to choose among competing gesture models that match well with the current video data.

The proposed framework is demonstrated in two gesture recognition systems. The first is a real-time system for continuous digit recognition, where gestures are performed by users wearing short sleeves, and (at least in one of the test sets) one to three people are continuously moving in the background. The second demonstration system enables users to find gestures of interest within a large video sequence or a database of such sequences. This tool is used to identify occurrences of American Sign Language (ASL) signs in video of sign language narratives that have been signed naturally by native ASL signers.

2 Related Work

The key differentiating feature of the proposed method from existing work in gesture recognition is that our method requires neither spatial nor temporal segmentation to be performed as preprocessing. In many dynamic gesture recognition systems (e.g., [1]–[7]) lower-level modules perform spatial and temporal segmentation and extract shape and motion features. Those features are passed into the recognition module, which classifies the gesture (Pavlovic et al. [8]). In such bottom-up methods, recognition will fail when the results of spatial or temporal segmentation are incorrect.

Despite advances in hand detection [9], [10] and hand tracking [11]–[15], hand detection remains a challenging task in many real-life settings. Problems can be caused by a variety of factors, such as changing illumination, low quality video and motion blur, low resolution, temporary occlusion, and background clutter. Commonly-used visual cues for hand detection such as skin color, edges, motion, and background subtraction [16], [17] may also fail to unambiguously locate the hands when the face, or other “hand-like” objects are moving in the background.

Instead of requiring perfect hand detection, we make the milder assumption that existing hand detection methods can produce, for every frame of the input sequence, a relatively short list of candidate hand locations, and that one of those locations corresponds to the gesturing hand. A similar assumption has been made in (Sato et al. [18]), and also in an earlier version of our work [19]. The key advantages of the method proposed here with respect to [18], [19] is that the proposed method: 1.) does not require temporal segmentation as preprocessing and 2.) achieves significant speed-ups by eliminating from consideration large numbers of implausible hypotheses.

An alternative to detecting hands before recognizing gestures is to use global image features. For example, Bobick et al. [20] propose using motion energy images. Dreuw et al. [21] use a diverse set of global features, such as thresholded intensity images, difference images, motion history, and skin color images. Gorelick et al. [22] use 3D shapes extracted by identifying areas of motion in each video frame. Nayak et al. [23] propose a histogram of pairwise distances of edge pixels. A key limitation of those approaches is that they are not formulated to tolerate the presence of distractors in the background, such as humans moving around. In scenes where such distractors have a large presence, the content of the global features can be dominated by the distractors.

In contrast, our method is formulated to tolerate such distractors, by efficiently identifying, among the exponentially many possible sequences of hand locations, the sequence that best matches the gesture model.

Ke et al. [24] propose modeling actions as rigid 3D patterns, and extracting action features using 3D extensions of the well-known 2D rectangle filters [25]. Two key differences between our work and that of Ke et al. are that, 1) in [24], actions are treated as 3D rigid patterns, and 2) in [24], distractors present in the 3D hyperrectangle of the action can severely affect the features extracted from that action (the method was not demonstrated in the presence of such distractors). Our method treats gestures as nonrigid along the time axis (thus naturally tolerating intra-gesture variations in gesturing speed), and, as already mentioned, is demonstrated to work well in the presence of distractors.

Several methods have been proposed in the literature for *gesture spotting* i.e., for recognizing gestures in continuous streams in the absence of known temporal segmentation [26]–[31]. One key limitation of all those methods is that they assume that the gesturing hand(s) have been unambiguously localized in each video frame.

Taking a closer look at existing gesture spotting methods, such methods can be grouped into two general approaches: the direct approach, where temporal segmentation precedes recognition of the gesture class, and the indirect approach, where temporal segmentation is intertwined with recognition. Methods that belong to the direct approach first compute low-level motion parameters such as velocity, acceleration, and trajectory curvature (Kang et al. [32]) or mid-level motion parameters such as human body activity (Kahol et al. [33]), and then look for abrupt changes (e.g., zero-crossings) in those parameters to identify candidate gesture boundaries. A major limitation of such methods is that they require each gesture to be preceded and followed by non-gesturing intervals – a requirement not satisfied in continuous gesturing.

Indirect methods for temporal gesture segmentation detect gesture boundaries by finding, in the input sequence, intervals that give good recognition scores when matched with one of the gesture classes. Most indirect methods [26], [27], [29] are based on extensions of Dynamic Programming (DP) based methods for isolated gestures, e.g., Dynamic Time Warping (DTW) [34], [35], Continuous Dynamic Programming (CDP) (Oka [29]), various forms of Hidden Markov Models (HMMs) [6], [13], [16], [27], [36]–[38], and most recently, Conditional Random Fields [39], [40]. In those methods, the gesture endpoint is detected by comparing the recognition likelihood score to a threshold. The threshold can be fixed or adaptively computed by a non-gesture garbage model [27], [41], equivalent to silence models in speech. Our method is also an indirect method; however, unlike these related approaches, our method does not require the hand to be unambiguously localized at each frame of the video sequence.

To reduce the time complexity of DP search, several pruning methods have been proposed in the speech recognition community (Jelinek [42]). Beam-search is perhaps the most commonly-known approach, and it has been successfully adapted for sign language gesture recognition (Wang et al. [43]). While this approach can dramatically reduce the search time, it requires *ad hoc* setting of beam-search parameters. In [26], we proposed that pruning can be posed as a binary classification problem, where the classifiers are trained automatically from training data. The resulting classifiers are used to quickly rule out unlikely search paths in the DP table. In this paper we illustrate that the approach of [26] can potentially have poor generalization properties, leading to over-pruning when applied to sequences that were not used for training. We propose new classifier training and pruning algorithms that offer improved generalization properties compared to [26].

Matching the input with all gesture models (a linear scan) may still be too slow for gesture recognition systems with gesture vocabularies on the order of a few thousands (Wang et al. [43]). To reduce the search time complexity, some systems [43]–[46] break gestures into motion subunits or *movemes* that are analogous to the phonemes used in speech recognition systems. Compared to the large number of gestures, the number of subunits is relatively small, usually on the order of a few tens. As a result, the search is over a smaller space of movemes, and many gesture models can be pruned from the search early if their prefixes (first few movemes) match the input very poorly. In our experiments with natural sign language communication by native signers, we found that many sign durations tend to be only 10-20 frames at a sampling rate of 30Hz; this number of frames per moveme is insufficient to learn and represent meaningful statistics.

After a set of candidate gestures has been detected, a set of rules is applied to select the best candidate, and to identify the input subsequence with the gesture class of that candidate. Different sets of rules have been proposed: peak finding rules (Morguet and Lang [28]), spotting rules (Yoon et al. [30]), and the user interaction model (Zhu et al. [31]). A problem that occurs in practice, but is often overlooked, is the *subgesture problem*: false detection of gestures that are similar to parts of other longer gestures. To address this, Lee and Kim [27] proposed two strategies. The first introduces a maximum length for non-gesture patterns that is greater than the length of the longest gesture. The second employs heuristics to infer the user's completion intentions, such as moving the hand out of camera range or freezing the hand for a while. The first approach requires an *ad hoc* parameter setting, and the second limits the naturalness of interaction. To lift these limitations, we propose a gesture recognition algorithm that explicitly models subgesture relations between gestures, and a learning algorithm that automatically identifies the subgesture relations from training data.

Preliminary versions of this work have appeared in [19] and [26]. With respect to those versions, the work in this paper includes significant improvements:

- Here we assume neither known temporal segmentation (assumed in [19]) nor accurate hand detection (assumed in [26]).
- We recognize gestures using models that are trained with Baum-Welch, as opposed to using exemplars.
- We describe an algorithm for learning subgesture relations, as opposed to hardcoding such relations in [26].
- We improve the pruning method of [26]. The new method is theoretically shown to be significantly less prone to eliminate correct hypotheses.
- We include a much wider array of experiments, including experiments with challenging backgrounds (including two or three moving people acting as distractors), and experiments on a new computer vision application, i.e., ASL sign retrieval in a video database of native ASL signing.

3 Problem Definition and Overview

Throughout the paper, we use double-struck symbols for sequences and models (e.g., $\mathbb{M}, \mathbb{Q}, \mathbb{V}, \mathbb{W}$), and calligraphic symbols for sets (e.g., $\mathcal{A}, \mathcal{S}, \mathcal{T}$).

Let $\mathbb{M}^1, \ldots, \mathbb{M}^G$ be models for G gesture classes. Each model $\mathbb{M}^g = (M_0^g, \ldots, M_m^g)$ is a sequence of states. Intuitively, state M_i^g describes the i-th temporal segment of the gesture modeled by $\mathbb{M}^g$. Let $\mathbb{V} = (V_1, V_2, \ldots)$ be a sequence of video frames showing a human subject. We denote with $\mathbb{V}_{sn}$ the frame sequence $(V_s, \ldots, V_n)$. The human subject occasionally (but not necessarily all the time) performs gestures that the system should recognize. Our goal is to design a method that recognizes gestures as they happen. More formally, when the system observes frame V_n, the system needs to decide whether a gesture has just been performed, and, if so, then recognize the gesture. We use the term "gesture spotting" for the task of recognizing gestures in the absence of known temporal segmentation.

Our method assumes neither that we know the location of the gesturing hands in each frame, nor that we know the start and end frame for every gesture. With respect to hand location, we make a much milder assumption: that a hand detection module has extracted from each frame V_j a set $\mathcal{Q}_j$ of K candidate hand locations. We use notation Q_{jk} for the feature vector extracted from the k-th candidate hand location of frame V_j.

The overall algorithm is depicted in Fig. 2. The online gesture recognition algorithm consists of three major components: hand detection and feature extraction, spatiotemporal matching, and inter-model competition. The offline learning phase, marked by the dashed arrows in Fig. 2, consists of learning the gesture models, the pruning classifiers, and the subgesture relations between gestures.

Sec. 4 describes the hand detection and feature extraction modules used in our experiments. Multiple candidate hand regions are detected using simple skin color and motion cues. Each candidate hand region is a rectangular image subwindow, and the feature vector extracted from that subwindow is a 4D vector containing the centroid location and optical flow of the subwindow. We should note that the focus of this paper is not on hand detection and feature extraction; other detection methods and features can easily be substituted for the ones used here.

Gesture models are learned by applying the Baum-Welch algorithm to training examples (Sec. 5.1). Given such a model $\mathbb{M}^g$ and a video sequence $\mathbb{V}_{1n}$, spatiotemporal matching consists of finding, for each $n' \in \{1, \ldots, n\}$, the optimal alignment between $\mathbb{M}^g$ and $\mathbb{V}_{1n'}$, under the constraint that the last state of $\mathbb{M}^g$ is matched with frame n' (i.e., under the hypothesis that n' is the last frame of an occurrence of the gesture). The optimal alignment is a sequence of triplets (i_t, j_t, k_t) specifying that model state $M_{i_t}^g$ is matched with feature vector $Q_{j_t k_t}$. As we will see, this optimal alignment is computed using dynamic programming. Observing a new frame V_n does not affect any of the previously computed optimal alignments between $\mathbb{M}^g$ and $\mathbb{V}_{1n'}$, for $n' < n$. At the same time, computing the optimal alignment between $\mathbb{M}^g$ and $\mathbb{V}_{1n}$ reuses a lot of the computations that were performed for previous optimal alignments. The spatiotemporal matching module is described in Sec. 5.2.

Allowing for multiple candidate hand locations increases the computational time of the dynamic programming algorithm. To address that, in Sec. 6 we describe a pruning method that can significantly speed up model matching, by rejecting a large number of hypotheses using quick-to-evaluate local criteria. Deciding whether to reject a hypothesis or not is a binary classification problem, and *pruning classifiers* are learned from training data for making such decisions. These pruning classifiers are applicable at every cell of the dynamic programming table, and have the effect of significantly decreasing the number of cells that need to be considered during dynamic programming.

The inter-model competition (Sec. 7) consists of deciding, at the current frame V_n, whether a gesture has just been performed, and if so, determining the class of the gesture. By "just been performed" we mean that the gesture has either ended at frame V_n or it has ended at a recent frame $V_{n'}$. This is the stage where we resolve issues such as two or more models giving low matching costs at the same time, and whether what we have observed is a complete gesture or simply a *subgesture* of another gesture – for example, whether the user has just performed the gesture for "5" or the user is in the process of performing the gesture for "8", as illustrated in Fig. 1. Subgesture relations are learned using training data.

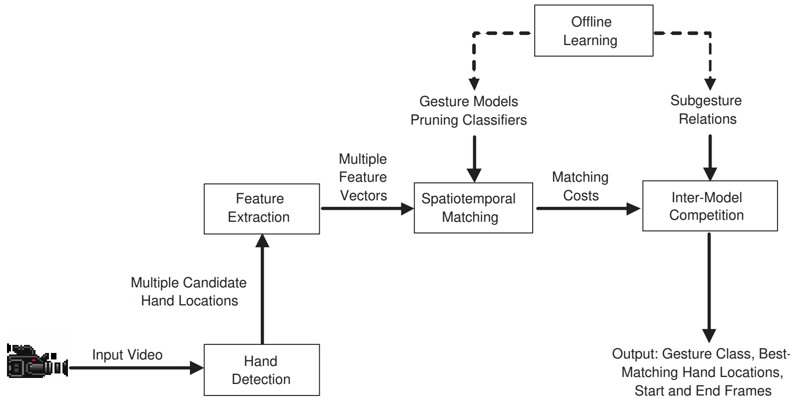


Fig. 2. System flowchart. Dashed lines indicate flow during offline learning. Solid lines indicate flow during online recognition.

4 Hand Detection and Feature Extraction

The spatiotemporal matching algorithm is designed to accommodate multiple hypotheses for the hand location in each frame. Therefore, we can afford to use a relatively simple and efficient preprocessing step for hand detection, that combines skin color and motion cues.

The skin detector first computes for every image pixel a skin likelihood term. For the first frames of the sequence, where a face has still not been detected, we use a generic skin color histogram (Jones and Rehg [47]) to compute the skin likelihood image. Once a face has been detected (using the detector of Rowley et al. [48]), we use the mean and covariance of the face skin pixels in normalized rg space to compute the skin likelihood image.

The motion detector computes a motion mask by thresholding the result of frame differencing. The motion mask is applied to the skin likelihood image to obtain the hand likelihood image. Using the integral image (Viola and Jones [25]) of the hand likelihood image, we efficiently compute for every subwindow of some predetermined size the sum of pixel likelihoods in that subwindow. Then we extract the K subwindows with the highest sum, such that none of the K subwindows may include the center of another of the K subwindows. Each of the K subwindows is constrained to be of size 40 rows x 30 columns. Alternatively, for scale invariance, the subwindow size can be defined according to the size of the detected face.

This way, for every frame j of the query sequence, K candidate hand regions are found. For every candidate k in frame j a feature vector Q_{jk} is extracted. In our experiments, $Q_{jk} = (x_{jk}, y_{jk}, u_{jk}, v_{jk})$, where the 2D position (x, y) is the pixel location of the region centroid, and the 2D velocity (u, v) is the optical flow averaged over that region. Optical flow is computed via block-based matching (Yuan et al. [49]).

We should emphasize that we do not use connected component analysis at all in the process of identifying candidate hand regions. The candidate hand regions are simply the 40x30 rectangles identified as described in the previous paragraph. This is in contrast with most existing methods (e.g., Chen et al. [16]), which find the single largest component, and associate it with the gesturing hand. A connected component algorithm may group the hand with the arm if the user is wearing a shirt with short sleeves, or with the face, or with any other hand-like objects with which the hand may overlap. In contrast, our hand detection algorithm maintains multiple subwindows for every frame of the video sequence, and some of these subwindows may occupy different parts of the same connected component. The gesturing hand is typically covered by one or more of these subwindows (see Figure 1).

5 Spatiotemporal Matching

In this section we describe our spatiotemporal matching algorithm, which can be applied in settings where neither spatial nor temporal segmentation of the input are available. We first describe how to learn gesture models, and then we describe how to match such models with features extracted from a video sequence.

5.1 Model Learning

In spatiotemporal matching we match feature vectors, such as the 4D position-flow vectors obtained as described in Sec. 4, to model states, according to observation density functions that have been learned for those states. Our first step is to learn those density functions, for each gesture class, using training examples for which spatiotemporal segmentation is known. Colored gloves are used – *only* in the training examples, not in the test sequences – to facilitate automatic labeling of hand locations.

Learning the observation density functions is done using a variant of the Baum-Welch algorithm [50], [51]. We manually specify the number of states for each model (as typically done with the Baum-Welch algorithm), and we also specify that transition probabilities between non-consecutive states are 0 and between consecutive states are uniform. In other words, at every transition the model is equally likely to remain at the same state or to move to the next state. We fix the transition probabilities to simplify the learning task, because we do not have sufficient training data to learn more parameters.

We note that the way we set transition probabilities, we do not allow model states to be skipped. This is simply an implementation decision. We make the assumption that the number of model states is significantly smaller than the number of frames in an occurrence of the gesture, so that skipping states is not needed. At the same time, the proposed framework can easily accomodate transition probabilities that allow for states to be skipped.

Given the above constraints we apply the standard EM procedure that is the core of the Baum-Welch algorithm. Naturally, if more training data are available, transition probabilities can also be learned, or alternative richer models can be constructed such as segmental Markov models (Ostendorf et al. [52]) or Conditional Random Fields (Lafferty et al. [39]).

5.2 Matching Video with a Model

Let $\mathbb{V} = (V_1, V_2, \ldots)$ be a sequence of video frames. We also call $\mathbb{V}$ a *query sequence* or simply a query. From each V_j we extract a *set* $\mathcal{Q}_j$ of K feature vectors: $\mathcal{Q}_j = \{Q_{j1}, \ldots, Q_{jK}\}$. Each feature vector is extracted from an image region which corresponds to a particular candidate hand location. Let $\mathbb{M}^g = (M_0^g, \ldots, M_m^g)$ be a gesture model of $m+1$ states. Each state M_i^g is associated with a Gaussian observation density (μ_i^g, Σ_i^g) that assigns a likelihood to each observation vector Q_{jk}, with μ_i^g and Σ_i^g being respectively the mean and the covariance matrix of the feature vectors observed in state M_i^g. As a reminder, in our implementation, each $Q_{j,k}$ is simply a 4D vector containing the centroid location and the optical flow for a subwindow corresponding to the k-th candidate hand region of frame j. We define the local cost $d(i,j,k) \equiv d(M_i^g, Q_{jk})$, between model state M_i^g and feature vector Q_{jk}, as the Mahalanobis distance $d(i,j,k) = (Q_{jk} - \mu_i^g)'(\Sigma^g)^{-1}(Q_{jk} - \mu_i^g)$. A special case is state M_0^g, which we treat as a "dummy" state that matches all feature vectors: $d(0,j,k) = 0$.

A warping path $\mathbb{W}$ defines an alignment between model $\mathbb{M}^g$ and a subsequence $(\mathcal{Q}_1, \ldots, \mathcal{Q}_n)$ ending at the current frame n. Formally, $\mathbb{W} = w_1, \ldots, w_T$, where each $w_t = (i_t, j_t, k_t)$ is a triple specifying that model state $M_{i_t}^g$ is matched with feature vector $Q_{j_t k_t}$. We say that w_t has two *temporal* dimensions (denoted by i and j) and one *spatial* dimension (denoted by k). A warping path $\mathbb{W}$ must satisfy the following constraints (Keogh [53]):

- *Boundary conditions:* $w_1 = (0, 1, k)$ and $w_T = (m, n, k')$. This requires the warping path to start by matching the dummy state M_0^g of the model with the first frame V_1 of the query, and end by matching the last state of the model with the last frame of the query. We require that the warping path ends at frame n, because we want to evaluate the hypothesis that current frame n is the last frame of the gesture (naturally, we have to evaluate this hypothesis every time we obtain a new current frame). No restrictions are placed on k and k'.
- *Temporal continuity:* Given $w_t = (a, b, c)$, the preceding triple $w_{t-1} = (a', b', c')$ must satisfy that $a - a' \leq 1$ and $b - b' \leq 1$. This restricts each step in the warping path to make smooth transitions (with respect to the previous step) along the two temporal dimensions.
- *Temporal monotonicity:* Given $w_t = (a, b, c)$, the preceding triple $w_{t-1} = (a', b', c')$ must satisfy that $a - a' \geq 0$ and $b - b' \geq 0$. This forces the warping path sequence to increase monotonically in the two temporal dimensions.

The matching cost $D(\mathbb{W})$ of warping path $\mathbb{W} = (i_1, j_1, k_1), \ldots, (i_T, j_T, k_T)$ is defined as

$$D(\mathbb{W}) = \sum_{t=1}^{T} d(i_t, j_t, k_t) \ , \tag{1}$$

where $d(i_t, j_t, k_t)$ is, as before, shorthand for the Mahalanobis distance between $M_{i_t}^g$ and Q_{j_t, k_t}. The optimal warping path $\mathbb{W}^*$ is simply the warping path $\mathbb{W}$ that minimizes $D(\mathbb{W})$. The optimal path $\mathbb{W}^* = (i_1^*, j_1^*, k_1^*), \ldots, (i_T^*, j_T^*, k_T^*)$ provides three important pieces of information:

- $D(\mathbb{W}^*)$ is a measure of how well the input sequence matches the model, and can be used to determine if the gesture modeled by $\mathbb{M}^g$ actually occurs in the input video $\mathbb{V}$. If we have multiple gesture classes, the matching costs between the input sequence and each class model can be compared to decide which model provides the best match.
- $\mathbb{W}^*$ specifies, for each input frame, the candidate hand location that optimizes the matching between the input sequence and the model. In other words, $\mathbb{W}^*$ specifies the optimal spatial segmentation of the gesture, under the hypothesis that the gesture has just been performed by the user.
- $\mathbb{W}^*$ also specifies the optimal temporal segmentation of the gesture (again, under the hypothesis that the gesture has just been performed by the user). The starting frame of the gesture is the first j_t^* such that $i_t^* = 1$, i.e., the first input frame that is not matched to the dummy state M_0^g. The end frame of the gesture j_T^* is constrained to equal n, i.e., the current frame.

Dynamic Programming (DP) can be used to find the minimum warping path cost efficiently. In particular, let $w_t = (i, j, k)$ be any possible warping path element. We define the cumulative cost $D(w_t)$ to be the cost of the optimal warping path matching the first $i+1$ model states $(M_0^g, \ldots, M_i^g)$ to the first j video frames $(V_1, \ldots, V_j)$, and matching state M_i^g with feature vector

Q_{jk}. The key property that allows us to use DP is that $D(w_t)$ can be defined recursively:

$$D(w_t) = \min_{w \in \mathcal{A}(w_t)} \{D(w)\} + d(w_t), \tag{2}$$

where $\mathcal{A}(w_t)$ is the set of all possible predecessors w_{t-1} that satisfy the warping path constraints, in particular continuity and monotonicity:

$$\begin{aligned} \mathcal{A}(w_t) &\equiv \mathcal{A}(i,j,k) \\ &= (\{(i,j-1),(i-1,j-1)\} \times \\ &\quad \{1,\ldots,K\}) \cup \{(i-1,j,k)\}. \end{aligned} \tag{3}$$

Note that j_t can be smaller (but not greater) than t, since we allow the same feature vector $Q_{i_t j_t}$ to match consecutive model states (in which case t increases but j_t does not increase).

Using the above recursion, DP is implemented using a 3D table, indexed by (i,j,k). When the system reads a new video frame V_n, the DP algorithm computes the 2D slice of the table that corresponds to frame n, storing at cell (i,n,k) two things: 1.) the value for $D(i,n,k)$, for $i = 0,\ldots,m$ and $k = 1,\ldots,K$, and 2.) the predecessor among $\mathcal{A}(i,n,k)$ that minimized Eq. 2. After we have computed all values $D(i,n,k)$, the minimum cost among all $D(m,n,k)$ (i.e., all costs corresponding to a match between the last state of the model and the current frame n) corresponds to the optimal alignment between the model and the video sequence. Using backtracking, i.e., following predecessor links, we obtain the optimal warping path.

6 Pruning

A straightforward implementation of the matching algorithm described above may be too slow for real-time applications, especially with larger vocabularies. The time complexity is linear to the number of models, the number of states per model, the number of video frames, and the number of candidate hand locations per frame. We therefore propose a pruning method that eliminates unlikely hypotheses from the DP search, and thereby makes spatiotemporal matching more efficient. In the theoretically best case, the proposed pruning method leads to time complexity that is independent of the number of model states (vs. complexity linear to the number of states for the algorithm without pruning). The key novelty in our pruning method is that we view pruning as a binary classification problem, where the output of a classifier determines whether to prune a particular search hypothesis or not.

6.1 Pruning Classifiers

The matching algorithm computes, for each cell (i,j,k) in the DP table, the optimal warping path ending at (i,j,k). The key idea in our pruning method is that there are cases where the matching is so poor that we can safely prune out cell (i,j,k), and reject all warping paths going through it.

The decision to prune a cell can be made by a *pruning classifier* C, which can be learned from training data. Different types of classifiers can be used including: *subsequence classifiers*, which prune based on the compatibility between an input prefix and a model prefix; single *observation classifiers*, which prune based on the compatibility between a single feature vector and a single model state; and *transition classifiers*, which prune based on the compatibility between successive observations. The pruning classifier we have used in our experiments is an observation classifier that checks whether the local matching cost $d(i,j,k)$ exceeds a threshold $\tau(i)$ associated with state M_i^g:

$$C_i(Q_{jk}) = \begin{cases} 1 & \text{if } d(i,j,k) \le \tau(i) \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

where $d(i,j,k)$ is the local matching cost between the observation Q_{jk} and model state M_i^g.

6.2 Learning Pruning Classifiers

In learning the pruning classifiers, the objective is to maximize efficiency and at the same time to minimize loss in accuracy. Intuitively, we want to prune out as many hypotheses as possible, while not pruning out the true optimal alignment between the gesture model and examples from the same gesture class.

In the first method we devised (described in [26]), a threshold $\tau(i)$ is learned for each classifier C_i, as follows: the gesture model aligned via dynamic programming with all the training examples from the same class. The threshold $\tau(i)$ is set to the maximum distance among the distances between model state M_i^g and all feature vectors Q_{jk} that are matched with M_i^g. However, the thresholds thus learned may overfit the training data, and the pruning classifiers may reject a significant fraction of correct warping paths in the test data.

To illustrate the worst case, let g be a gesture class, let $\mathbb{M}^g$ be the gesture model, and suppose we have N training examples and N test examples for g. Assume that training and test examples are independent identically-distributed (iid), and that, in any sequence, any pair of observations is conditionally independent, given the model states that those observations correspond to. Let $\mathbb{S}_s$ be a test example, and $\mathbb{W}_s = (i_1,j_1,k_1),\ldots,(i_T,j_T,k_T)$ be the correct warping path between $\mathbb{S}_s$ and the model $\mathbb{M}^g$. The correct warping path will be pruned out if, for at least one triple (i,j,k), it holds that $d(i,j,k) > \tau(i)$.

Let $\tau'(i)$ be the value we would have computed for $\tau(i)$ if we had used the N test images as our training set. Since both the training and the test examples are iid, there is a 0.5 probability that $\tau'(i) > \tau(i)$, in which case for at least one test example (the one from which $\tau'(i)$ was computed), the correct warping path will be pruned out, because it will match a feature vector Q_{jk}

to state M_i^g so that $d(i,j,k) = \tau'(i) > \tau(i)$. Therefore, the probability that a random sequence $\mathbb{S}_s \in \mathcal{S}$ would be rejected because of $\tau(i)$ is at least $\frac{1}{2N}$.

At a more intuitive level, we can consider the distances between state M_i^g and all feature vectors (from both the training and the test sequences of gesture g) matched with that state under dynamic programming. If we have an equal number N of both training and test examples, there is a 0.5 probability that the largest such distance occurs in a test example, and then τ_i will cause the correct warping path for that example to be pruned. This gives a probability of at least $\frac{1}{2N}$ for any correct warping path to be pruned due to a bad match with state M_i^g.

Since we learn one threshold $\tau(i)$ for each state M_i^g, we learn a total of m such thresholds (where m is the number of states in model M_i^g), and the probability that a random test example $\mathbb{S}_s$ will be pruned by any $\tau(i)$ is at least $1-(1-\frac{1}{2N})^m$. This pruning probability can be substantial for small N and large m. To reduce overfitting and the chance of pruning the optimal path, leave-one-out cross validation is used to learn a single parameter ϵ, which we call the *expansion factor* and we add to *every* $\tau(i)$. The value for ϵ is chosen to be the minimum value such that, if every threshold $\tau(i)$ is incremented by ϵ, no optimal path is pruned out during cross-validation. Adding ϵ to each $\tau(i)$ reduces the pruning probability from $1-(\frac{2N-1}{2N})^m$ to $\frac{1}{N}$. In our digit recognition application experiments, where $N = 30$ and $m = 10$, the probability that the correct warping path for a random test sequence will be pruned out is reduced to 0.033 compared to 0.154 without the expansion factor.

6.3 Spatiotemporal Matching with Pruning

Pruning classifiers can be incorporated in the spatiotemporal matching algorithm in order to speed it up. We call the resulting algorithm Continuous Dynamic Programming with Pruning (CDPP). In this algorithm, pruning classifiers are applied to each (i,j,k) cell in the dynamic programming table. Implementing dynamic programming appropriately, so as to achieve these computational savings, involves certain subtleties. Due to space limitations we do not fully describe the dynamic programming implementation in this paper, but we provide full details on that subject in [54].

The CDPP algorithm is invoked separately for every gesture model $\mathbb{M}^g$, every time the system observes a new video frame. After the matching costs for all models have been computed, the inter-model competition algorithm is invoked to determine if a gesture has just been performed, as will be described in the next section.

7 Subgestures Relations and Inter-Model Competition

The inter-model competition stage of our method needs to decide, when multiple models match well with the input video, which of those models (if any) indeed corresponds to a gesture performed by the user. This stage requires knowledge about the subgesture relationship between gestures. A gesture g_1 is a subgesture of another gesture g_2 if g_1 is similar to a part of g_2 under the similarity model of the matching algorithm. For example, as seen in Fig. 1, in the digit recognition task the digit "5" can be considered a subgesture of the digit "8". In this section we describe how to learn subgesture relations from training data, and how to design an inter-model competition algorithm that utilizes knowledge about subgestures.

7.1 Learning Subgesture Relations

The inter-model competition algorithm requires knowledge about the subgesture relationship between gestures. Subgesture relations can be manually specified using domain knowledge. However, when subgesture relations are less obvious and when the number of gesture models is large, an automated learning method is called for.

Subgesture relations are learned automatically as follows: given a particular gesture model $\mathbb{M}^{g_1}$ and a training example $\mathbb{V}^{g_2}$ from a different gesture g_2 we run CDPP on this model-exemplar pair. If there exists a warping path $\mathbb{W} = ((1,j_s),\ldots,(m,j_e))$ for some $1 \leq j_s \leq j_e \leq n$ that was not pruned by the algorithm, then $\mathbb{M}^{g_1}$ matches a subsequence $\mathbb{V}^{g_2}_{j_s:j_e}$, and therefore g_1 is a subgesture of g_2. We repeat this process for all the remaining training examples of gesture g_2, and the number of times that $\mathbb{M}^{g_1}$ matches a subsequence of an exemplar for gesture g_2 can serve as a confidence measure of g_1 being a subgesture of g_2. Table 1 shows an example set of learned subgesture relations, and Table 2 shows the corresponding confidence scores, i.e., the number of times for each gesture pair g_1, g_2 that $\mathbb{M}^{g_1}$ matches a subsequence of an exemplar for gesture g_2.

In [54] we provide a description of the subgesture learning algorithm in pseudocode.

7.2 Inter-Model Competition

The inter-model competition algorithm consists of two steps. The first step updates the current list of candidate gesture models. The second step uses a set of rules to decide if a gesture was performed, i.e., if one of the candidate models truly corresponds to a gesture performed by the user. In order to describe the algorithm we first need the following definitions:

- *Partial path*: a warping path that matches an input subsequence $\mathbb{V}_{s:n}$ with the first i model states $M_1^g,\ldots,M_i^g$, so that frames s and n are matched respectively to states 1 and i.
- *Complete path*: a warping path matching an input subsequence $\mathbb{V}_{s:n}$ with the complete model $\mathbb{M}^g$, so that frame n is matched to the last state of the model.
- *Active path*: any partial (or complete) path that has not been pruned by CDPP. In other words, any path

stored in an unpruned DP cell that corresponds to the current frame n.

- *Active model*: a model $\mathbb{M}^g$ that has an active path. An active model $\mathbb{M}^g$ represents a hypothesis that, at the current frame n, we are observing a (possibly still incomplete) occurrence of gesture g.
- *Firing model*: a model $\mathbb{M}^g$ that has a complete path for which the path cost is below some detection acceptance threshold chosen for that model. A firing model $\mathbb{M}^g$ represents a hypothesis that, at the current frame n, we have observed the completion of an occurrence of gesture g.

The inter-model competition algorithm is invoked once for each input frame n. One input to this algorithm is a table of subgesture relations. This table is automatically constructed from training data, offline, as described in Sec. 7.1. Another input is a current list of candidate gestures. Before the first video frame is observed, this list of candidate models is initialized to be empty. The third input to the algorithm is the state of all DP tables at the current frame.

At an intuitive level, after all the matching costs have been updated for the current frame n, the best firing model (if such a model exists) defines a gesture hypothesis h' that is considered for inclusion in the list of candidate gestures. For every new candidate h' we record its class $g_{h'}$, the frame at which it has been detected (which we call the "end frame" for that gesture), the corresponding start frame, and the optimal matching cost (normalized, by dividing the original cost by the path length). Decisions on candidates are not made immediately; a candidate gesture may be considered for several frames, before it is finally accepted or rejected. At every frame, existing candidates are considered for removal from the list. Finally, the system may decide, according to a set of rules, that one of the candidate gestures has indeed been performed.

The inter-model competition algorithm is described below. We describe two versions of the algorithm, that differ on whether they use the table of subgesture relations or not. We use the acronym CDPP (Continuous Dynamic Programming with Pruning) for the version that does not use the table, and we use the acronym CDPPS (CDPP with Subgesture Reasoning) for the version that takes subgesture relations into account.

1) Find the best firing model $\mathbb{M}^{g'}$, i.e., the model with the lowest cost for the current frame, and define the corresponding candidate gesture h'. The significance of this step is that, if multiple models are firing at the current frame n, we only generate a candidate gesture h' for the model with the lowest matching cost; we ignore the rest of the firing models.
2) For all candidates $h_i \neq h'$ (recall that a candidate may be considered for several frames before getting accepted or rejected) perform the following tests:
   a) CDPPS only: if g' (the gesture class associated with candidate h') is a supergesture of g_{h_i} then delete h_i. This rule enforces that a gesture class g_{h_i} (e.g., the digit "5") cannot be recognized when a supergesture of g_{h_i} (e.g., the digit "8") is also a candidate.
   b) CDPPS only: if g' is a subgesture of g_{h_i} then delete h'. The reasoning here is the same as in the previous step.
   c) If the start frame of h' occurred before the end frame of h_i, then delete the worst scoring among h' and h_i. This rule enforces that no two candidates overlap with each other. If the current candidate h' is overlapping with a previous candidate h_i, only the one with the lowest matching cost among the two survives.

After the list of candidates has been updated, if the list of candidates is nonempty then a candidate h_i may be "spotted", i.e., accepted as a true occurrence of gesture g_{h_i}, if all current active paths started after the end frame of h_i. In other words, h_i is accepted if and only if all hypotheses overlapping with h_i have been rejected (and h_i has still not been rejected).

Once a gesture has been recognized, the list of candidates is reset to empty, all DP tables are cleared, and the entire process (spatiotemporal matching and inter-model competition) starts again, from the first video frame after the end frame of the recognized gesture.

8 Extensions

8.1 Translation and Scale Invariance

To gain invariance to translation and scale we can normalize the features with respect to the location and scale of a face detection window. Since recent face detection algorithms (e.g., Viola and Jones [25]) have reached a relatively high level of accuracy and efficiency, the location and size of a face detection window can serve as a good estimate of the location and scale of a gesture, assuming the gesture is performed at the same location and scale with respect to the face.

8.2 Matching Two-Handed Gestures

For ease of explanation the spatiotemporal matching algorithm was described for one-handed gestures. One way to model two-handed gestures is to have two separate models, one per hand, where the models can be either coupled (Brand et al. [36]) or parallel (Vogler and Metaxas [37]). An alternative is to concatenate the two hand observation vectors while maintaining a consistent order, namely, first the left hand's feature followed by the right (Starner et al. [6]).

In our work the hands' identities are not known in the first frame, and multiple ($K \geq 2$) hand hypotheses are maintained throughout the input sequence. To address the multiple hypothesis problem, we maintain K^2 (concatenated) feature vectors reflecting all ordered combinations of left/right hand hypotheses, $Q_{jk} \equiv (Q^l_{j,k_1}, Q^r_{j,k_2})$, where $1 \leq k \leq K^2$, and $1 \leq k_1, k_2 \leq K$. The CDPP algorithm will then find the optimal sequence of two-hand hypotheses.

9 Complexity

The time complexity of the spatiotemporal matching algorithm is $O(Kmn)$, where K is the number of candidate hand locations at every input frame, and m and n are the number of model states and number of video frames respectively. The complexity of matching two-handed gestures is $O(K^2mn)$. During matching, only the matrices corresponding to the current and previous frame are needed to be stored in memory, and therefore the space complexity is a factor of n less than the corresponding time complexity. Alternatively, a circular buffer of fixed size can be kept for every model to allow the system to trace back the optimal path.

10 Experiments

We evaluate the proposed gesture recognition framework in two settings: a gesture recognition setting, where users perform gestures corresponding to the 10 digits (Figs. 3 and 4), and an ASL sign retrieval setting. In our experiments the users and signers can wear short sleeved shirts, the background may be arbitrary (e.g., an office environment in the digit gesture dataset) and even contain other moving objects, and hand-over-face occlusions are allowed.

The proposed system is quantitatively evaluated in terms of accuracy and computation time. The computation time is measured both as the average processing time per frame, and the average number of operations per frame, where the number of operations is simply the number of Dynamic Programming cells visited by the matching algorithm. All experiments were conducted on a 2GHz Opteron-based Windows PC with 2GB of RAM.

10.1 Continuous Digit Recognition

This experiment is intended to evaluate the performance of the entire gesture recognition system that combines spatial and temporal segmentation. Users are asked to make gestures chosen from the 10 Palm Graffiti Digits, as shown in Figs. 3 and 4. The video clips for the digits dataset were captured with a Unibrain Firewire camera at 30Hz using an image size of 240×320. Each digit gesture video clip depicts a user gesturing the ten digits in sequence (although the system does not know that the digits are performed in sequence), and the goal of the system is to recognize the digits as they are gestured.

We should note that the order in which digits appear in a sequence does not make any difference in system accuracy. The gesture models are trained using segmented examples, so no information about the preceding digit is available. Also, the hand returns to the rest position after performing each digit, so there is no coarticulatory effect between gestures.

The training set used for learning the digit models consisted of 30 video sequences, 3 sequences from each of 10 users. In each sequence, the user signed once each of the 10 digits, for a total of 30 training examples per digit class. Users wore colored gloves in the training sequences. The 30 training sequences were used in the offline learning stage, to learn models, pruning classifiers, and subgesture relations.

We used two test sets for evaluating performance on digit recognition: an "easy" set and a "hard" set. In each test sequence, the user signed once each of the 10 digits, and wore short sleeves (and, naturally, no colored gloves). For both test sets, experiments were performed in a user-independent fashion, where the system recognizes test gestures of a particular user using digit models which were learned from training examples collected from other users.

The easy test set contains 30 short sleeve sequences, three from each of 10 users. The hard test set contains 14 short sleeve sequences, two from each of seven users. The sequences in the hard test set contain distractors, in the form of one to three humans (besides the gesturing user) moving back and forth in the background (Fig. 5). The presence of such distractors makes these sequences quite challenging for methods assuming reliable hand tracking and methods relying on global features. In the easy test set there are no distractors.

Given a test frame, K candidate hand regions of size 40×30 were detected and corresponding feature vectors were extracted as described in Sec. 4. Single observation classifiers (Sec. 6.1 Eq. 4) were used for pruning and expansion factors were learned as described in Sec. 6.2. The resulting matching costs were fed into the inter-model competition algorithm, which decided whether or not a gesture has ended at that time.

To measure the system's accuracy on this data set, we report the number of correct detections and the number of false positives. We define that a correct detection occurs when the class label is correct, and 50% of the duration of the estimated gesture overlaps with the ground truth gesture and vice versa, as proposed by Williams [55]. A false positive is any detection result that is not correct.

The inter-model competition algorithm used Subgesture Table 1 to detect gestures in the video stream. The subgesture table was automatically learned from training data as described in Section 7.1. From inspection of the table it is evident that the learning algorithm successfully discovered meaningful subgesture relations.

The number of model states was selected in proportion to the trajectory length of the digit prototype. The digit prototype was the training example for that digit that had the smallest sum of DTW distances to all other training examples for that digit. In the optimal setting

Fig. 3. Palm's Graffiti Digits.

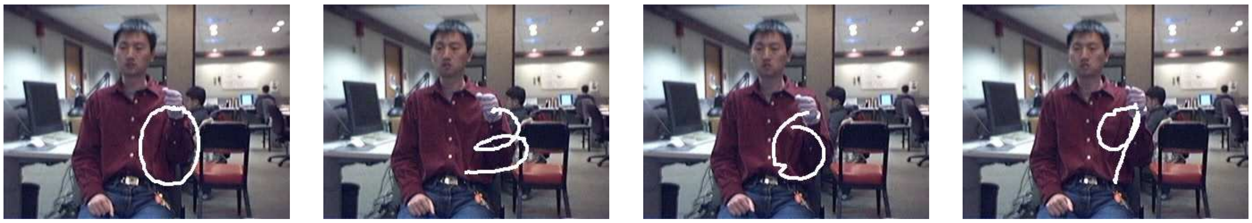


Fig. 4. Example digit gestures for classes "0", "3", "6", and "9", as performed by a user.

Fig. 5. Example frames from video sequences in our hard test set, showing users in the foreground performing digit gestures, and other people moving around in the background.

Subgesture	Supergestures
"1"	{"7","9"}
"3"	{"2","7"}
"4"	{"5","8","9"}
"5"	{"8"}
"7"	{"2","3"}
"9"	{"5","8"}

TABLE 1

Subgesture table learned from training data.

Subgesture confidences for all pairs of gestures										
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	1
2	0	0	0	0	0	0	0	0	0	0
3	0	0	15	0	0	0	0	2	0	0
4	0	0	0	0	0	37	0	0	88	26
5	0	0	0	0	0	0	0	0	197	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	71	62	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	67	0	0	15	0

TABLE 2

Subgesture confidences computed as described in Section 7.1. Rows correspond to subgestures and columns correspond to supergestures.

K	Easy dataset				Hard dataset			
	No SR		SR		No SR		SR	
	DR	FP	DR	FP	DR	FP	DR	FP
1 (Oka [29])	68.3	24	75.6	2	8.5	3	8.5	3
2	79.0	37	91.0	1	23.5	4	23.5	4
3	78.0	51	93.6	4	37.8	7	40.0	4
4	78.6	54	94.6	5	50.7	7	53.5	3
5	77.6	58	94.6	5	62.8	14	67.8	7
6	77.3	59	94.6	5	64.2	22	71.4	11
7	77.0	60	92.6	6	69.2	24	79.2	9
8	77.0	60	90.6	7	67.8	32	81.4	9
10	78.6	57	91.0	6	66.4	42	82.8	14
12	77.0	64	88.0	10	63.5	52	82.8	15
14	77.3	62	84.6	12	62.1	56	82.1	17
15	77.0	61	83.3	11	65.7	51	85.0	13
16	76.6	60	80.6	11	61.4	56	80.7	16

TABLE 3

Detection rate (DR) percentage, and number of false positives (FP) for different numbers K of candidate hand regions for frame, for the easy and hard digit datasets. SR indicates that subgesture relations were used during the inter-model competition. Note that our method with $K = 1$ becomes the CDP method of Oka ([29]).

(determined using cross-validation), one model state corresponded to three frames of the prototype.

Table 3 illustrates the detection accuracy rate and number of false positives for each test set. For the easy dataset, the best result without subgesture reasoning was attained for $K = 2$, with a detection rate of $237/300 = 79.0\%$ and 37 false positives. Using subgesture reasoning, the best result was attained for $K = 4, 5, 6$, with a detection rate of $284/300 = 94.6\%$, and only 5 false positives.

For the hard dataset, the best result without subgesture reasoning was attained for $K = 7$, with a detection rate of $97/140 = 69.2\%$ and 24 false positives. Using subgesture reasoning, the best result was attained for $K = 15$, with a detection rate of $119/140 = 85.0\%$, and 13 false positives. Table 4 shows the confusion matrix for the hard dataset.

We note that using subgesture reasoning improves performance significantly for almost all K and both test sets, by increasing detection rates and decreasing false positives. The optimal K value for the hard test set was 15, compared to 6 for the easy test set, illustrating that the presence of distractors caused the correct hand regions to be ranked significantly lower than in the easy test set. At the same time, we notice that performance degrades gracefully in the presence of distractors: the optimal detection rate drops from 94.6% to 85.0% between the easy test set and the hard test set. To the best of our knowledge, methods assuming reliable hand tracking, or using global features, have not been demonstrated on scenes where the presence of distractors is comparable to the presence of distractors in our hard test set.

We note that using $K = 1$ (i.e., assuming reliable hand tracking), our method reduces to the continuous dynamic programming method (CDP) of Oka [29]. As can be seen from the $K = 1$ row of Table 3, CDP yields a relatively low detection rate of 75.6% for the easy test set. More remarkably, for the hard test set, CDP leads to a really low detection rate of 8.5%. Any $K \neq 1$ produces significantly higher detection rates than CDP. These results emphasize the advantages of our method: with a very simple hand detection module, but allowing multiple candidate hand regions per frame, we can overcome to a large extent the challenges posed by the distractors present in the hard test set.

A speed-up factor of 4.2 (in terms of runtime) was obtained using our pruning method, and the processing time for spatiotemporal matching and inter-model competition was 0.6 ms per frame. The speed-up obtained by pruning in this experiment is a somewhat pessimistic estimate. Pruning depends on several factors including:

1) The similarity between gesture prefixes. In our experiment all digits start at the top, and therefore it is difficult to prune based on position early on.
2) The ratio between gesture time and total time. A small ratio means that the periods in between gestures are relatively long. During these periods there is more chance of pruning since the non gesture patterns would tend to match the models very poorly. To verify the influence of the gesture time ratio, we conducted an experiment where we varied the periods of rest between gestures. Fig. 6 shows that the amount of pruning increases as the rest time in between gestures increases, leading to a speedup of a factor of 10 for cases where gestures occupy a small fraction of the input sequence.

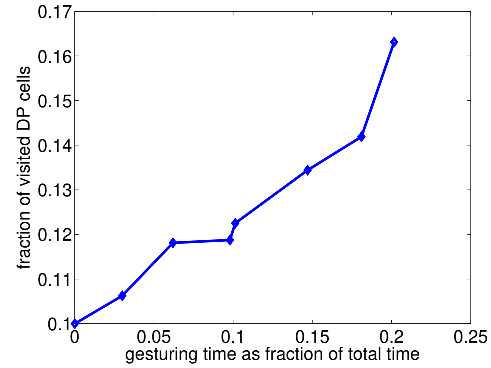


Fig. 6. Fraction of visited cells (out of total DP cells) as a function of fraction of gesturing time (out of total time).

On the easy test set, not using pruning, for $K = 6$ the detection rate drops from 94.6% to 84.3%, and the number of false positives increases from 5 using pruning to 7 without pruning. This result illustrates that our pruning method not only decreases the running time, but also significantly improves the accuracy of the system, by pruning away hypotheses that otherwise could lead to recognition errors.

As a last experiment on the digits dataset, we have found that decreasing the number of states per model leads to decreasing accuracy. For the easy test set, and $K = 6$, as the number of prototype digit frames per model state increases from 3 to 4, 6, 8, and 10, the detection rate drops respectively from 94.6% to $92.0\%, 82.0\%, 67.3\%$, and 57.7%.

All training and test sequences from the digit recognition dataset, as well as ground truth files for all those sequences, are available for viewing/downloading at: `http://cs-people.bu.edu/athitsos/digits`.

10.2 ASL Sign Retrieval

In this experiment we consider a gesture-finding tool, where the system assists the user in efficiently finding occurrences of an ASL sign in video sequences. ASL has no standard written form, and video is the standard form of recording ASL narrative. Retrieving occurrences of signs in such video can enable content-based search of narratives that is analogous to keyword-based search of text documents.

We have experimented with two data sets. The first data set, used for a preliminary evaluation, is a video clip of an ASL story. The story is 26,805 frames long at 30 frames per second. In terms of signs, the story is 1071 signs long. In this experiment we find occurrences of 3 ASL signs: “BETTER”, “HERE”, and “WOW”. The

Confusion matrix										
Hard dataset, without using subgesture relations										
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	9	0	0	0	0	0	0	0	0	0
1	1	13	2	1	1	3	0	0	3	5
2	0	0	7	1	0	0	0	0	0	0
3	0	0	0	7	0	0	0	0	0	0
4	0	0	0	0	14	9	0	0	1	3
5	0	0	0	1	0	2	0	0	5	0
6	3	0	0	0	0	0	14	0	0	0
7	0	0	5	5	0	0	0	14	0	0
8	0	0	0	0	0	0	0	0	5	0
9	1	0	0	0	0	0	0	0	0	8
miss	5	1	7	7	0	12	0	0	9	6

Confusion matrix										
Hard dataset, using subgesture relations										
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	10	0	0	0	0	0	0	0	0	0
1	0	13	0	1	1	0	0	0	0	1
2	0	0	12	2	0	0	0	1	0	0
3	0	0	0	11	0	0	0	1	0	0
4	0	0	0	0	14	0	0	0	0	0
5	0	0	0	0	0	10	0	0	0	0
6	3	0	0	0	0	0	13	0	0	0
7	0	0	0	0	0	0	0	12	0	0
8	0	0	0	0	0	1	0	0	14	0
9	1	0	0	0	0	0	0	0	0	11
miss	4	1	2	3	0	4	1	2	0	3

TABLE 4

Confusion matrices for $K = 15$, for the hard digit dataset. Rows correspond to estimated class labels, and columns correspond to ground truth labels. For the purpose of computing these matrices, for each detected gesture we find the nearest-in-time ground-truth gesture, and we increment the matrix entry defined by the class labels of those two gestures. The "miss" row counts, for each gesture class, the number of occurrences of that class with which no detection result could be associated.

average sign lengths are: 14 frames for "BETTER", 14 frames for "HERE", and 31 frames for "WOW". In the story there are 5 examples of the sign "BETTER", 9 examples of "HERE", and 4 examples of "WOW". Given the model of the gesture of interest, we associate each frame of the input sequence with the matching cost of the optimal warping path going through that frame. Then, a threshold is chosen manually, in a way that guarantees that all occurrences of the sign of interest are below that threshold.

Since we only consider the threshold guaranteeing 100% recall of the signs of interest, our performance measure is the *retrieval ratio*, which is defined as the ratio between the number of frames retrieved using that threshold and the total number of frames. For each detection result, all frames from the warping path of that result are counted as retrieved. In the absence of a gesture-finding tool, the user would simply have to review the entire video, which is equivalent to a retrieval ratio of 1.0. Using our gesture-finding tool, the retrieval ratio for the signs "BETTER," "HERE," and "WOW" is 0.19, 0.11, and 0.07 respectively. We should note that, based on the number of occurrences of the query signs and the average length of those occurrences, the theoretically optimal result for this experiment would be 0.0026 for "BETTER," 0.0047 for "HERE," and 0.0046 for "WOW."

A problem with this data set is that we could only find three signs for which we had enough examples to perform off-line training. To perform a more comprehensive quantitative evaluation, we have used a second data set of video clips of ASL signs. In every clip a native ASL signer gestures 24 signs: 7 one-handed signs and 17 two-handed signs, as listed in Table 5. Each sign belongs to one of the five types [56] shown in Table 6. Each sign starts with a particular hand shape and ends with a particular hand shape, as shown in Table 5.

The video sequences for the two ASL datasets were captured at 60Hz using an original image size of 480×640 pixels, which was downsampled to 240×320 pixels for the experiments. We collected ten sequences where the signer wears two colored gloves: a green glove on the dominant (right) hand, and a purple glove on the nondominant hand (left). Those sequences were used for learning the sign models. As test sequences, we collected ten sequences where the signer wears a short sleeved shirt, thus making it difficult to localize hands unambiguously at each frame.

The results are shown in Table 7. The performance measures we report are the number of false positives generated when the detection threshold is set to detect all ten occurrences of each sign (i.e., 100% detection rate), and the retrieval ratio, as defined above. The total number of frames is 32,060. For most signs the number of false positives is relatively low and the proportion of frames retrieved by the system out of the total number of frames is also relatively small. Two one-handed signs ("PAST," and "TELL") and four two-handed signs ("BIG," "FINISH," "MANY," and "WOW") generated too many false positives. The reason for that is that those signs have little apparent (2D) motion and/or their learned models are too simple (contain only a few states). It is interesting to note that we obtained a retrieval ratio of 0.07 and 0.25 for "WOW" in the first and second data set respectively. A different signer performed "WOW" in each of the two data sets. In the

One-handed signs			
Sign	Type	Handshapes	Average Length
AND	0	$5 > 0$	19
KNOW	X	B-L	23
MAN	X	5	33
OUT	0	$5 > 0$	17
PAST	0	B-L	18
TELL	0	1	13
YESTERDAY	X	A(10)	36

Two-handed signs			
Sign	Type	Handshapes	Average Length
ARRIVE	2	B-L/B-L	22
BIG	1	bent-L/bent-L	15
BORN	2	B-L/B-L	34
CAR	1	S/S	63
DECIDE	1	F/F	15
DIFFERENT	1	1/1	15
FINISH	1	5/5	14
HERE	1	B-L/B-L	64
MANY	1	$S > 5$	16
MAYBE	1	B-L/B-L	62
NOW	1	Y/Y	11
RAIN	1	bent 5/ bent 5	36
READ	3	2/B-L	22
TAKE-OFF	2	B-L/B-L	24
TOGETHER	1	A/A (10/10)	13
WHAT	1	5/5	42
WOW	1	bent 5	45

TABLE 5

ASL signs used in experiments: 7 one-handed signs and 17 two-handed signs. Sign types are based on Battison [56] (see Table 6). In the Handshapes column, start and end shapes are depicted as *start* > *end*, dominant and nondominant hands are depicted as *dominant* / *nondominant*.

Sign Type	Description
Type 0	One handed, **not** contacting the body
Type X	One handed, contacting the body (but **not** the other hand)
Type 1	Two handed, both moving, synchronous or alternating movements
Type 2	Two handed, one active, one passive, **same** handshape
Type 3	Two handed, one active, one passive, **different** handshape

TABLE 6

Classification of signs based on number of hands [56].

One-handed signs		
Sign	# False Positives	Retrieval Ratio
AND	26	1/51
KNOW	6	1/149
MAN	47	1/22
OUT	2	1/138
PAST	637	1/8
TELL	390	1/37
YESTERDAY	0	1/112

Two-handed signs		
Sign	# False Positives	Retrieval Ratio
ARRIVE	0	1/139
BIG	249	1/33
BORN	4	1/137
CAR	0	1/64
DECIDE	7	1/120
DIFFERENT	0	1/572
FINISH	37	1/85
HERE	1	1/47
MANY	164	1/38
MAYBE	13	1/29
NOW	65	1/78
RAIN	35	1/48
READ	0	1/159
TAKE-OFF	0	1/324
WHAT	8	1/77
WOW	253	1/4

TABLE 7

Results of spotting one-handed signs and two-handed signs for the second ASL video dataset. The false positive rate is given for 100% detection rate. The retrieval ratio is the fraction between the number of retrieved frames and the total number of frames.

first data set, the examples of "WOW" exhibited more motion, thus making it easier to identify them compared to the second experiment.

Subgesture relations were not used in these sign retrieval experiments. For the first of the two datasets we have used, there was insufficient training data available to learn the subgesture table. That said, subgesture relations are useful when available. In the general case, the ASL sequences on which sign retrieval can be performed use an unrestricted vocabulary of hundreds or thousands of signs. In many cases, a trained model is not available for a large number of those signs, and learning subgesture relations requires the availability of such models.

In our ASL retrieval experiments there were five signs with repetitive motion: "WOW," "CAR," "MAYBE," "RAIN," and "WHAT". The models for those signs were trained in the same way the other models were trained, even though different training examples of a particular sign had a different number of repetitions. Identifying more appropriate training methods for such signs remains a topic for future investigation.

11 Discussion and Conclusions

This paper presented a novel gesture spotting algorithm that is accurate and efficient, is purely vision-based, and can robustly recognize gestures, even when the user gestures without any aiding devices in front of a complex background. The proposed algorithm can recognize gestures using a fairly simple hand detection module that yields multiple candidates. The system does not break down in the presence of a cluttered background, multiple moving objects, multiple skin-colored image regions, and users wearing short sleeve shirts. It is indicative that, in our experiments on the hard digit dataset, the proposed algorithm increases the correct detection rate tenfold, from 8.5% to 85%, compared to the continuous dynamic programming method of Oka [29], which assumes reliable hand detection results.

A promising direction for improving both accuracy and efficiency is to use a cascade approach, where efficient but less accurate features are used for spatiotemporal matching, and expensive but more accurate features are used for higher-level gesture recognition and verification. We have performed a preliminary evaluation of this approach, by adding to the proposed system a verification module that is based on hand appearance in specific states of the model. This led to promising results, as detailed in [54]. For example, by applying the verification classifier to all instances of recognition of the sign "NOW", the number of false positives decreased from 65 to only 12, without rejecting any true appearances of the gesture. This result is very encouraging, indicating that verification and, more generally, a cascaded approach, can play a significant role in improving system performance.

It has been demonstrated that language models improve the accuracy of continuous sign language recognition systems [6], [43], [44], [46]. Even relatively simple language models, such as unigram (Bauer et al. [44]), bigram (Bauer et al. [44]), and Part of Speech (Starner et al. [6]) models, provide very useful priors about gesture sequences and are often combined with data likelihood in a Bayesian framework to provide the posterior probabilities of hypothesized gesture sequences. Although language modeling is likely to improve the results of our ASL sign retrieval tool this is not the main focus of our paper. Furthermore, some vision-based gesture recognition systems for HCI may not be associated with a language model, allowing instead command gestures to appear in arbitrary order.

Template-based approaches like CDP (Oka [29]) or DTW [34], [35] have the advantage that only one example is needed, but they lack a statistical model for variations. On the other hand, higher accuracy is expected when using more expressive dynamic models, such as segmental Markov models (Ostendorff et al. [52]) or Conditional Random Fields [39], [40], which however require a large amount of training data. We have instead proposed a hybrid approach, where we estimate a Gaussian model for the observation probabilities (like an HMM), but employ the uniform transition probability model of DTW. Such an approach has also been used for online handwritten character recognition (Bahlmann and Burkhardt [50]). Overall, we believe that the simple dynamic models used in this paper are a good compromise between template-based methods and more expressive dynamic models, given the moderate amount of training data available in our data sets.

An interesting topic for future work is formulating a more general inter-model competition algorithm. In this work, a few simple rules coupled with learned subgesture relations have been demonstrated to perform quite well. At the same time, it will be interesting to explore formulations based on model selection methods, such as MDL, that can integrate information from model matching costs and subgesture relations.

Acknowledgments

This work was supported by NSF grants CNS-0202067, IIS-0329009, and IIS-0705749.

References

[1] A. Corradini, "Dynamic time warping for off-line recognition of a small gesture vocabulary," in *Proc. IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, 2001, pp. 82–89.

[2] R. Cutler and M. Turk, "View-based interpretation of real-time optical flow for gesture recognition," in *Proc. Third IEEE International Conference on Automatic Face and Gesture Recognition*, 1998, pp. 416–421.

[3] T. Darrell and A. Pentland, "Space-time gestures," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1993, pp. 335–340.

[4] M. Gandy, T. Starner, J. Auxier, and D. Ashbrook, "The gesture pendant: A self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring," in *International Symposium on Wearable Computing*, 2000, pp. 87–94.

[5] K. Oka, Y. Sato, and H. Koike, "Real-time fingertip tracking and gesture recognition," *IEEE Computer Graphics and Applications*, vol. 22, no. 6, pp. 64–71, 2002.

[6] T. Starner, J. Weaver, and A. Pentland, "Real-time American Sign Language recognition using desk and wearable computer based video," *Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1371–1375, December 1998.

[7] M. H. Yang, N. Ahuja, and M. Tabb, "Extraction of 2D motion trajectories and its application to hand gesture recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 8, pp. 1061–1074, 2002.

[8] V. Pavlovic, R. Sharma, and T. Huang, "Visual interpretation of hand gestures for human-computer interaction: A review," *Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 677–695, July 1997.

[9] Y. Cui and J. Weng, "Appearance-based hand sign recognition from intensity image sequences," *Computer Vision and Image Understanding*, vol. 78, no. 2, pp. 157–176, May 2000.

[10] E. Ong and R. Bowden, "A boosted classifier tree for hand shape detection," in *Proc. Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, 2004, pp. 889–894.

[11] M. Isard and A. Blake, "CONDENSATION - conditional density propagation for visual tracking," *Intenational Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[12] M. Kolsch and M. Turk, "Fast 2D hand tracking with flocks of features and multi-cue integration," in *IEEE Workshop on Real-Time Vision for Human-Computer Interaction*, 2004, pp. 158–165.

[13] N. Stefanov, A. Galata, and R. Hubbold, "Real-time hand tracking with variable-length Markov Models of behaviour," in *Real Time Vision for Human-Computer Interaction*, 2005, pp. III: 73–73.

[14] B. Stenger, A. Thayananthan, P. Torr, and R. Cipolla, "Filtering using a tree-based estimator," in *Proc. IEEE International Conference on Computer Vision*, 2003, pp. 1063–1070.

[15] E. Sudderth, M. Mandel, W. Freeman, and A. Willsky, "Visual hand tracking using nonparametric belief propagation," in *Proc. IEEE CVPR Workshop on Generative Model Based Vision*, 2004.

[16] F. Chen, C. Fu, and C. Huang, "Hand gesture recognition using a real-time tracking method and Hidden Markov Models," *Image and Video Computing*, vol. 21, no. 8, pp. 745–758, August 2003.

[17] J. Martin, V. Devin, and J. Crowley, "Active hand tracking," in *Proc. Third IEEE International Conference on Automatic Face and Gesture Recognition*, 1998, pp. 573–578.

[18] Y. Sato and T. Kobayashi, "Extension of Hidden Markov Models to deal with multiple candidates of observations and its application to mobile-robot-oriented gesture recognition," in *Proc. International Conference on Pattern Recognition*, 2002, pp. II: 515–519.

[19] J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff, "Simultaneous localization and recognition of dynamic hand gestures," in *Proc. IEEE Workshop on Motion and Video Computing*, 2005, pp. II: 254–260.

[20] A. Bobick and J. Davis, "The recognition of human movement using temporal templates," *Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 257–267, March 2001.

[21] P. Dreuw, T. Deselaers, D. Keysers, and H. Ney, "Modeling image variability in appearance-based gesture recognition," in *ECCV Workshop on Statistical Methods in Multi-Image and Video Processing*, 2006, pp. 7–18.

[22] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 12, pp. 2247–2253, 2007.

[23] S. Nayak, S. Sarkar, and B. Loeding, "Unsupervised modeling of signs embedded in continuous sentences," in *IEEE Workshop on Vision for Human-Computer Interaction*, 2005.

[24] Y. Ke, R. Sukthankar, and M. Hebert, "Efficient visual event detection using volumetric features," in *IEEE International Conference on Computer Vision (ICCV)*, vol. 1, 2005, pp. 166–173.

[25] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2001, pp. I:511–518.

[26] J. Alon, V. Athitsos, and S. Sclaroff, "Accurate and efficient gesture spotting via pruning and subgesture reasoning," in *Proc. IEEE ICCV Workshop on Human Computer Interaction*, 2005, pp. 189–198.

[27] H. Lee and J. Kim, "An HMM-based threshold model approach for gesture recognition," *Pattern Analysis and Machine Intelligence*, vol. 21, no. 10, pp. 961–973, October 1999.

[28] P. Morguet and M. Lang, "Spotting dynamic hand gestures in video image sequences using Hidden Markov Models," in *Proc. IEEE International Conference on Image Processing*, 1998, pp. 193–197.

[29] R. Oka, "Spotting method for classification of real world data," *The Computer Journal*, vol. 41, no. 8, pp. 559–565, July 1998.

[30] H. Yoon, J. Soh, Y. Bae, and H. Yang, "Hand gesture recognition using combined features of location, angle and velocity," *Pattern Recognition*, vol. 34, no. 7, pp. 1491–1501, July 2001.

[31] Y. Zhu, G. Xu, and D. Kriegman, "A real-time approach to the spotting, representation, and recognition of hand gestures for human-computer interaction," *Computer Vision and Image Understanding*, vol. 85, no. 3, pp. 189–208, March 2002.

[32] H. Kang, C. Lee, and K. Jung, "Recognition-based gesture spotting in video games," *Pattern Recognition Letters*, vol. 25, no. 15, pp. 1701–1714, November 2004.

[33] K. Kahol, P. Tripathi, and S. Panchanathan, "Automated gesture segmentation from dance sequences," in *Proc. Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, 2004, pp. 883–888.

[34] T. Darrell, I. Essa, and A. Pentland, "Task-specific gesture analysis in real-time using interpolated views," *Pattern Analysis and Machine Intelligence*, vol. 18, no. 12, pp. 1236–1242, 1996.

[35] J. B. Kruskall and M. Liberman, "The symmetric time warping algorithm: From continuous to discrete," in *Time Warps, String Edits and Macromolecules*, J. B. Kruskal and D. Sankoff, Eds. Addison-Wesley, 1983, pp. 125–162.

[36] M. Brand, N. Oliver, and A. Pentland, "Coupled Hidden Markov Models for complex action recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1997, pp. 994–999.

[37] C. Vogler and D. Metaxas, "Parallel Hidden Markov Models for American Sign Language recognition," in *Proc. IEEE International Conference on Computer Vision*, 1999, pp. 116–122.

[38] A. Wilson and A. Bobick, "Parametric Hidden Markov Models for gesture recognition," *Pattern Analysis and Machine Intelligence*, vol. 21, no. 9, pp. 884–900, September 1999.

[39] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random

fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. 18th International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2001, pp. 282–289.

[40] A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell, "Hidden conditional random fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 10, pp. 1848–1852, 2007.

[41] H. Yang, A. Park, and S. Lee, "Robust spotting of key gestures from whole body motion sequence," in *Proc. Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, 2006, pp. 231–236.

[42] F. Jelinek, *Statistical Methods for Speech Recognition*. MIT Press, 1997.

[43] C. Wang, W. Gao, and S. Shan, "An approach based on phonemes to large vocabulary Chinese sign language recognition," in *Proc. Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, 2002, pp. 393–398.

[44] B. Bauer, H. Hienz, and K. Kraiss, "Video-based continuous sign language recognition using statistical methods," in *Proc. International Conference on Pattern Recognition*, 2000, pp. Vol II: 463–466.

[45] R. Bowden, D. Windridge, T. Kadir, A. Zisserman, and M. Brady, "A linguistic feature vector for the visual interpretation of sign language," in *Proc. Eighth European Conference on Computer Vision*, 2004, pp. Vol I: 390–401.

[46] C. Vogler, "American Sign Language recognition: Reducing the complexity of the task with phoneme-based modeling and parallel Hidden Markov models," Ph.D. dissertation, University of Pennsylvania, 2003.

[47] M. Jones and J. Rehg, "Statistical color models with application to skin detection," *Intenational Journal of Computer Vision*, vol. 46, no. 1, pp. 81–96, January 2002.

[48] H. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23–38, January 1998.

[49] Q. Yuan, S. Sclaroff, and V. Athitsos, "Automatic 2D hand tracking in video sequences," in *Proc. IEEE Workshop on Applications of Computer Vision*, 2005, pp. 250–256.

[50] C. Bahlmann and H. Burkhardt, "The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping," *Pattern Analysis and Machine Intelligence*, vol. 26, no. 3, pp. 299–310, March 2004.

[51] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[52] M. Ostendorf, V. Digalakis, and O. Kimball, "From HMMs to segment models: A unified view of stochastic modeling for speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 5, pp. 360–378, 1996.

[53] E. Keogh, "Exact indexing of dynamic time warping," in *International Conference on Very Large Data Bases*, 2002, pp. 406–417.

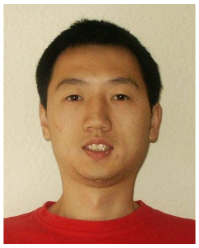
[54] J. Alon, "Spatiotemporal gesture segmentation," Ph.D. dissertation, Dept. of Computer Science, Boston University, Technical Report BU-CS-2006-024, 2006.

[55] G. Williams, "A study of the evaluation of confidence measures in automatic speech recognition," University of Sheffield, Tech. Rep. CS-98-02, 1998.

[56] R. Battison, *Lexical Borrowing in American Sign Language*. Linstok Press, 1978.

Jonathan Alon received the BSc degree in physics and computer science from Tel Aviv University in 1994, and both the MA degree and PhD degree in computer science from Boston University in 2001 and 2006. In September 2006 he joined the image processing group at NegevTech as an algorithms engineer, developing methods for visual inspection for the semiconductor industry. His research interests include computer vision and pattern recognition. His published work is in the areas of gesture recognition, efficient database retrieval, and character recognition.

Vassilis Athitsos received the BS degree in mathematics from the University of Chicago in 1995, the MS degree in computer science from the University of Chicago in 1997, and the PhD degree in computer science from Boston University in 2006. In 2005-2006 he worked as a researcher at Siemens Corporate Research, developing methods for database-guided medical image analysis. In 2006-2007 he did a postdoc at the Computer Science department at Boston University. Since Fall 2007 he is an assistant professor at the Computer Science and Engineering department at the University of Texas at Arlington. His research interests include computer vision, machine learning, and data mining. His recent work has focused on efficient similarity-based retrieval, gesture and sign language recognition, human motion analysis, shape modeling and detection, and medical image analysis.



Quan Yuan received the BS and MS degree in computer science and engineering from Harbin Institute of Technology in 2001 and 2003 respectively. Since September 2003, he has been a PHD candidate in the Computer Science Department at Boston University. His research interest include computer vision and machine learning. His recent work has focused on object recognition, gesture recognition and medical image analysis. He is a student member of IEEE.

Stan Sclaroff received the PhD degree from the Massachusetts Institute of Technology in 1995. He is currently a professor of computer science and the chair of the Department of Computer Science at Boston University. He founded the Image and Video Computing research group at Boston University in 1995. In 1996, he received a US Office of Naval Research (ONR) Young Investigator Award and a US National Science Foundation (NSF) Faculty Early Career Development Award. Since then, he has coauthored numerous scholarly publications in the areas of tracking, video-based analysis of human motion and gesture, deformable shape matching and recognition, as well as image/video database indexing, retrieval, and data mining methods. He has served on the technical program committees of more than 80 computer vision conferences and workshops. He has served as an associate editor for the IEEE Transactions on Pattern Analysis, 2000-2004 and 2006-present. He is a senior member of the IEEE.