

# Translation and Scale-Invariant Gesture Recognition in Complex Scenes

Alexandra Stefan<sup>1</sup>, Vassilis Athitsos<sup>2</sup>, Jonathan Alon<sup>1</sup>, and Stan Sclaroff<sup>1</sup>

<sup>1</sup> Computer Science Department, Boston University

<sup>2</sup> Computer Science and Engineering Department, University of Texas at Arlington

## ABSTRACT

Gestures are a natural means of communication between humans, and also a natural modality for human-computer interaction. Automatic recognition of gestures using computer vision is an important task in many real-world applications, such as sign language recognition, computer games control, virtual reality, intelligent homes, and assistive environments. In order for a gesture recognition system to be robust and deployable in non-laboratory settings, the system needs to be able to operate in complex scenes, with complicated backgrounds and multiple moving and skin-colored objects. In this paper we propose an approach for improving gesture recognition performance in such complex environments. The key idea is to integrate a face detection module into the gesture recognition system, and use the face location and size to make gesture recognition invariant to scale and translation. Our experiments demonstrate the significant advantages of the proposed method over alternative computer vision methods for gesture recognition.

## Categories and Subject Descriptors

H.5.2 [User Interfaces]: Input Devices and Strategies; I.4.8 [Scene Analysis]: Motion

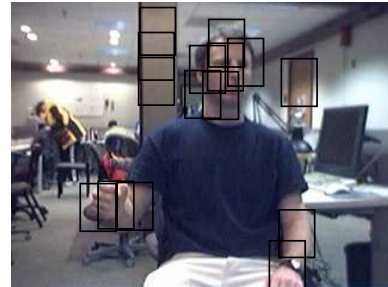
## Keywords

gesture recognition, dynamic space-time warping, DSTW

## 1. INTRODUCTION

Gestures are a natural means of communication between humans, and also a natural modality for human-computer interaction. Automatic recognition of gestures using computer vision is an important task in many real-world applications, such as sign language recognition, computer games control, virtual reality, intelligent homes, and assistive environments. Compared to many existing human-computer interfaces, hand gestures have the advantages of being easy to use, natural, and intuitive. Successful applications of hand gesture recognition include computer games control [8], human-robot interaction [22], and sign language recognition [21], to name a few. Vision-based recognition systems can give computers the capability of understanding and responding to hand gestures.

The usability of gesture recognition systems greatly depends on their ability to function reliably in common real-world environments, without requiring the user to wear special clothes or cumbersome devices such as colored markers or gloves [22]. Furthermore, in order for a gesture recognition system to be robust and



**Figure 1: Detection of candidate hand regions based on skin color. Clearly, skin color is not sufficient to unambiguously detect the gesturing hand since the face, the non-gesturing hand, and other objects in the scene have similar color. On the other hand, for this particular scene, the gesturing hand is consistently among the top 15 candidates identified by skin detection.**

deployable in non-laboratory settings, the system needs to be able to operate in complex scenes, with complicated backgrounds and multiple moving and skin-colored objects.

Most hand gesture recognition systems assume that the gesturing hand can be reliably located in every frame of the input sequence. Skin detection, motion detection, edges, and background subtraction are commonly used for identifying hand location in each frame [4, 12]. However, in many application settings the assumption that we can have a preprocessing module that reliably detects and locations in every frame is simply unrealistic. For example, in Figure 1 skin detection yields multiple hand candidates, and the top candidate is often not correct. Other visual cues commonly used for hand detection such as motion, edges, and background subtraction would also fail to unambiguously locate the hand in the image. Motion-based detection and background subtraction may fail to uniquely identify the location of the hand when the face, non-gesturing hand or other scene objects (such as walking people in the background) are moving.

Dynamic Space-Time Warping (DSTW) [2] is a gesture recognition method designed to address this problem. In contrast to most alternative methods, DSTW does not require hands to be correctly and unambiguously detected at each frame. Instead, DSTW has the significantly milder requirement that at each frame a list of multiple hand locations is available. As shown in Figure 1, even a simple skin-color detection scheme, taking a few lines of code to implement, can suffice in producing a relatively short list of candidate hand locations.

Another important consideration in gesture recognition systems is invariance. Ideally, a gesture should be recognized as long as

the gesture is visible by a camera, regardless of the position, orientation, and distance of the person performing the gesture with respect to the camera. In this paper, we make the assumption that the person performing the gesture is facing the camera, so that person orientation with respect to the camera is fixed. What we do address is translation and scale invariance. We describe a method that can recognize gestures regardless of the position and scale of the person in the scene, as long as we obtain a frontal view of the person and sufficient image resolution.

DSTW is a general method, that can achieve different types of invariance depending on the types of features that are used. However, the DSTW-based system described in [2] does not have the property of scale invariance. Also, while an alternative implementation is proposed in [2] for achieving translation invariance, that modification is costly in terms of recognition time, and can negatively affect the ability of the system to recognize gestures at interactive speeds. In contrast, the method described here achieves both translation and scale invariance, and the running time is about the same as that of the original, translation-dependent DSTW implementation.

The key idea is to integrate a face detection module into the gesture recognition system, and use the face location and size to convert features into a translation and scale invariant representation. We define a coordinate system where the origin is the center of the face, and the unit vectors are defined according to the size of the face in the image. Representing hand location and motion in this coordinate system is inherently translation and scale invariant: If the person moves to the left or to the right, the face and hands move approximately the same way. If the person moves closer or farther with respect to the camera, the face and hands scale in approximately the same way. We should point out that our method is not invariant to image plane or out-of-plane rotations; we assume that the user is standing or sitting upright and facing the camera.

It is important to note that we assume that it is easy to detect the face, whereas it is hard to detect the hands. This assumption is grounded in the current state of the art in computer vision. Mature, publicly-available real-time face detection systems have been available for several years [17, 23], whereas reliable hand detection remains an active research topic, with state-of-the-art systems being too slow for interactive applications [24]. Intuitively, faces are much easier to detect: they contain distinct-looking subparts (eyes, nose, mouth), and if the user sits or stands upright, and faces towards the camera, then the 3D orientation of the face is highly constrained. In contrast, hands are highly articulated nonrigid objects and their 3D orientation can vary widely even when the user is upright and facing the camera.

Inspired by previous vision-based HCI systems (e.g., the virtual whiteboard by Black and Jepson [3], and the virtual drawing package by Isard [9]), we evaluate our framework on a vision-based character recognition task with 10 gesture classes. Our method achieves a high level of accuracy, over 96%, as measured on 300 test sequences, collected from 10 different users. Furthermore, this accuracy corresponds to user-independent recognition, where training sequences from a user were not used at all in the process of recognizing test sequences from the same user.

## 2. RELATED WORK

In most dynamic gesture recognition systems (e.g., [6, 21]) information flows bottom-up: the video is input into the analysis module, which estimates the hand pose and shape model parameters, and these parameters are in turn fed into the recognition module, which classifies the gesture [14]. In a bottom-up framework, tracking and recognition typically fail in the absence of perfect hand

segmentation. Our method does not suffer from the bottom-up approach drawbacks, as it does not place unrealistic requirements upon the low-level task of hand detection: creating a relatively short list of candidate hand locations is sufficient, as long as the correct location is included in that list. Hand detection does not have to precisely identify the left and right hand at each frame.

Dynamic Space-Time Warping (DSTW) [2], the method that we are using in this paper, is an extension of Dynamic Time Warping (DTW). DTW was originally intended to recognize spoken words of small vocabulary [11, 15]. It was also applied successfully to recognize a small vocabulary of gestures [5, 7]. The DTW algorithm temporally aligns two sequences, a query sequence and a model sequence, and computes a matching score, which is used for classifying the query sequence. The time complexity of the basic DTW algorithm is quadratic in the sequence length, but more efficient variants have been proposed [18, 10]). In DTW, it is assumed that a feature vector can be reliably extracted from each query frame. However, this assumption is often hard to satisfy in vision-based systems, where the gesturing hand cannot be located with absolute confidence. A framework that allows for multiple detections of candidate hand regions, or more generally multiple observations, is therefore required.

In multiple hypothesis tracking (e.g., [16]) multiple hypotheses are associated with multiple observations. Each observation corresponds to a different object with a different model. In contrast, in the proposed method a single consistent hypothesis is selected among multiple distinct observations (detections), only one of which is correct. The CONDENSATION-based framework can also be applied to gesture recognition [3]. Although in principle CONDENSATION can be used for both tracking and recognition, in [3] CONDENSATION was only used for the recognition part, once the trajectory had been reliably estimated using a color marker. Even given the trajectory, system performance was reported to be significantly slower than real time, due to the large number of hypotheses that needed to be evaluated and propagated at each frame. Also, to use CONDENSATION we need to know the observation density and propagation density for each state of each class model, whereas in our method no such knowledge is necessary.

The work by Sato and Kobayashi [19] is closely related to our work. In the Hidden Markov Model (HMM) framework, Sato and Kobayashi extended the Viterbi algorithm so that multiple candidate observations can be accommodated at each query frame; the optimal state sequence is constrained to pass through the most likely candidate at every time step. HMMs have found wider application for problems with large vocabulary (of words or gestures) primarily due to their ability to probabilistically encode the variability of the training data. However, DSTW can still be appropriate for smaller problems because it is simpler to implement: there is no need to worry about the HMM structure, and no training is required. Furthermore, our approach differs from [19] in that it incorporates translation and scale invariance, and is evaluated in a more challenging setting (users are wearing short sleeve shirts and the hand is not an isolated skin-colored blob).

DSTW [2] is the approach most related to the method described in this paper, which uses DSTW but proposes a different implementation than that of [2]. The key advantage of the proposed method over the DSTW implementation of [2] is that we achieve translation and scale invariance at approximately the same computational cost at which the implementation of [2] recognized gestures in a translation- and scale-dependent manner. In our experiments we demonstrate how the performance of the implementation of [2] degrades as translation and scale changes are introduced, whereas the



Figure 2: Palm's Graffiti digits.



Figure 3: Example model digits extracted using a colored glove.

performance of the method proposed here is unaffected.

### 3. OVERVIEW

Our method is an appearance-based and example-based gesture recognition method. The goal is to recognize a distinct and finite number of gesture classes. The particular gesture classes we want to recognize depend on the actual application. In our experiments we have chosen to recognize 10 gesture classes corresponding to the 0-9 numerical digits, as shown on Figures 2 and 3.

For each gesture, we have several training examples of that gesture in our database. Each training example is a video sequence. In the training examples, the person performing the gesture is wearing

a colored glove, in order to make it easy to detect hand location automatically in those examples. It is important to note that colored gloves are only used in the training examples; in the test sequences no colored gloves are used. The idea is that, in a commercial system, collection and annotation of training examples is done by the system developers, whereas the system, when deployed, recognizes gestures performed by users. It is OK for system developers to use colored gloves or other markers in order to improve system development and system accuracy, as long as the end users can gesture in a natural way that does not require the use of such visual aids.

Given the video sequence of a gesture that we want to recognize, the system first pre-processes the video sequence, so that, for each frame of the sequence, a relatively short number (5 to 20) of candidate hand locations is identified. In our experiments, we found that it is relatively easy to identify such candidate hand locations using skin detection and motion detection. The idea is that hands are skin-colored objects, and when we perform a gesture the hand moves relatively fast. Therefore, looking for hands can easily be reduced to looking for skin-colored fast-moving objects, which can be coded easily and can run efficiently.

Another type of information extracted from the input video sequence is face location. We use the publicly available face detector developed by Rowley, et al. at CMU [17]. The detector gives us the bounding rectangle of the face, from which we can readily obtain the center and size of the face. If processing time is an issue, the significantly faster Viola-Jones face detector [23] can be used in place of the CMU detector.

Every candidate hand location is a rectangle of a specific size and location. For each candidate location we extract some features. The types of features that we extract have consequences with respect to whether the overall method is translation- and scale-invariant. In order to achieve translation and scale invariance, we use two very simple features: position and motion. The position is represented simply as the  $(x, y)$  pixel co-ordinates over the centroid of the candidate hand location. The motion is represented as a 2D vector, which is an estimate of the motion between the previous frame and the current frame for that particular location. This motion estimate is simply the average optical flow of all pixels in the candidate location. Optical flow is computed using the method of Simoncelli, et al. [20].

Clearly, the position feature is both translation and scale-dependent. The motion feature is translation invariant but is still scale-dependent. However, we can make both position and motion translation and scale invariant by performing a simple coordinate transformation, moving from the image coordinate system to a face-centered coordinate system where the origin is the center of the face bounding rectangle and the unit of distance is defined based on the size of the face.

In summary then, each candidate hand location is represented using a 4D vector, containing four translation and scale-invariant numbers: two numbers for position, and two numbers for motion. If we extract  $K$  candidate hand locations for each video frame in the input sequence, after feature extraction and normalization we end up with  $K$  4D feature vectors for each frame.

For the training examples in the database, features are extracted in a similar way. The only difference is that for the training video sequences, the location of the gesturing hand is known, because users in those sequences wear color gloves. Therefore, for each video frame in the training examples, a single 4D feature vector is extracted.

Recognizing the input gesture is done using the nearest neighbor classification framework, i.e., by identifying the training example that is the most similar to the input gesture. Section 5 describes in

detail both the DSTW similarity measure and the DSTW algorithm for measuring the similarity between an input sequence and a training example. A key aspect of the DSTW algorithm is that it can evaluate similarity between two sequences despite the fact that, in the input sequence, hand location is not known (we only know that the true hand location is one of the candidate hand locations that we have identified).

In the following sections we describe each of the components of our method: feature extraction is described in Section 4, and Dynamic Space-Time Warping is described in Section 5.

## 4. DETECTION AND FEATURE EXTRACTION

The proposed method uses DSTW [2], which has been designed to accommodate multiple hypotheses for the hand location in each frame (see Section 5). Therefore, we can afford to use a relatively simple and efficient hand detection scheme. In our implementation we combine two visual cues, i.e., color and motion; both requiring only a few operations per pixel. Skin color detection is computationally efficient, since it involves only a histogram lookup per pixel. Similarly, motion detection, which is based on frame differencing, involves a small number of operations per pixel.

### 4.1 Detection

The face detector [17] is applied to the first frame of the input sequence. The output of the detector is the bounding rectangle of the face. Given this output, we can readily obtain the centroid and size of the face. We also use the mean and covariance of the face skin pixels in normalized  $rg$  space to compute a skin color model that is built on the fly and is highly customized to the user performing the gesture.

The skin detector computes for every image pixel a skin likelihood term, given the skin color model that was built based on the results of face detection. The motion detector computes a mask by thresholding the result of frame differencing (frame differencing is the operation of computing, for every pixel, the absolute value of the difference in intensity between the current frame and the previous frame). If there is significant motion between the previous and current frame the motion mask is applied to the skin likelihood image to obtain the hand likelihood image. Using the integral image [23] 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. If there is no significant motion between the previous and current frame, then the  $K$  subwindows chosen at the previous frame are reused as candidate hand locations at the current frame.

A distinguishing feature of our hand detection algorithm compared to most existing methods [4] is that we do not use connected component analysis to find the largest component (discounting the face), and associate it with the gesturing hand. The 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 skin-colored objects with which the hand may overlap. As a result the hand location, which is typically represented by the largest component’s centroid, will be incorrectly estimated. In contrast, our hand detection algorithm maintains for every frame of the sequence multiple subwindows, some of which 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).

## 4.2 Feature Extraction

For every frame  $j$  of the query sequence,  $K$  candidate hand regions are found as described in the previous section. For every candidate  $k$  in frame  $j$  a 4D feature vector  $Q_{jk} = (x_{jk}, y_{jk}, u_{jk}, v_{jk})$  is extracted. The 2D position  $(x, y)$  is the region centroid, and the 2D velocity  $(u, v)$  is the optical flow averaged over that region. Optical flow is computed using the method of Simoncelli et al. [20].

In our current implementation, when we collect the model sequences, a colored glove is used to reliably detect the gesturing hand. Using such additional constraints, like colored markers, is often desirable for the offline model-building phase, because it simplifies the construction of accurate class models. It is important to stress that such markers are not used in the query sequences, and therefore they do not affect the naturalness and comfort of the user interface.

The 4D feature vector  $Q_{jk} = (x_{jk}, y_{jk}, u_{jk}, v_{jk})$  extracted from each hand location is normalized, so that it corresponds to a face-centered coordinate system, where the face is the origin and the bounding rectangle of the face becomes a square whose sides have unit length.

## 5. DYNAMIC SPACE-TIME WARPING

One of several publications that describe the DTW algorithm is [10]. In this section we will describe dynamic space time warping [2], which is an extension of DTW that can handle multiple candidate detections in each frame of the query.

Let  $M = (M_1, \dots, M_m)$  be a model sequence in which each  $M_i$  is a feature vector. Let  $Q = (Q_1, \dots, Q_n)$  be a query sequence. In the regular DTW framework, each  $Q_j$  would be a feature vector, of the same form as each  $M_i$ . However, in dynamic space-time warping (DSTW), we want to model the fact that we have multiple candidate feature vectors in each frame of the query. For example, if the feature vector consists of the position and velocity of the hand in each frame, and we have multiple hypotheses for hand location, each of those hypotheses defines a different feature vector. Therefore, in our algorithm,  $Q_j$  is a *set* of feature vectors:  $Q_j = \{Q_{j1}, \dots, Q_{jK}\}$ , where each  $Q_{jk}$ , for  $k \in \{1, \dots, K\}$ , is a candidate feature vector.  $K$  is the number of feature vectors extracted from each query frame. In our algorithm we assume  $K$  is fixed, but in principle  $K$  may vary from frame to frame.

A warping path  $W$  defines an alignment between  $M$  and  $Q$ . Formally,  $W = w_1, \dots, w_T$ , where  $\max(m, n) \leq T \leq m + n - 1$ . Each  $w_t = (i, j, k)$  is a triple, which specifies that feature vector  $M_i$  of the model is matched with feature vector  $Q_{jk}$ . We say that  $w_t$  has two *temporal* dimensions (denoted by  $i$  and  $j$ ) and one *spatial* dimension (denoted by  $k$ ). The warping path is typically subject to several constraints (adapted from [10] to fit the DSTW framework):

- **Boundary conditions:**  $w_1 = (1, 1, k)$  and  $w_T = (m, n, k')$ . This requires the warping path to start by matching the first frame of the model with the first frame of the query, and end by matching the last frame of the model with the last frame of the query. No restrictions are placed on  $k$  and  $k'$ , which can take any value from 1 to  $K$ .
- **Temporal continuity:** Given  $w_t = (a, b, k)$  then  $w_{t-1} = (a', b', k')$ , where  $a - a' \leq 1$  and  $b - b' \leq 1$ . This restricts the allowable steps in the warping path to adjacent cells along the two temporal dimensions.
- **Temporal monotonicity:** Given  $w_t = (a, b, k)$  then  $w_{t-1} = (a', b', k')$  where  $a - a' \geq 0$  and  $b - b' \geq 0$ . This forces the

warping path sequence to increase monotonically in the two temporal dimensions.

**input** : A sequence of model feature vectors  $M_i, 1 \leq i \leq m$ , and a sequence of sets of query feature vectors  $Q_j = \{Q_{j1}, \dots, Q_{jK}\}, 1 \leq j \leq n$ .

**output**: A global matching cost  $D^*$ , and an optimal warping path  $W^* = (w_1^*, \dots, w_T^*)$ .

```

// Initialization
j = 0
for i = 0 : m do
  for k = 1 : K do
    | D(i, j, k) = ∞
  end
end
D(0, 0, 1) = 0
// Iteration
for j = 1 : n do
  for i = 0 : m do
    for k = 1 : K do
      if i = 0 then
        | D(i, j, k) = ∞
      end
      else
        w = (i, j, k)
        for w' ∈ N(w) do
          | C(w', w) = τ(w', w) + D(w'),
        end
        D(w) = d(w) + min_{w' ∈ N(w)} C(w', w)
        b(w) = argmin_{w' ∈ N(w)} C(w', w)
      end
    end
  end
end
// Termination
k* = argmin_k {D(m, n, k)}
D* = D(m, n, k*)
w_T* = (m, n, k*)
// Backtrack
w_{t-1}* = b(w_t*)

```

**Algorithm 1:** The DSTW algorithm

Note that continuity and monotonicity are required only in the temporal dimensions. No such restrictions are needed for the spatial dimension; the warping path can “jump” from any spatial candidate  $k$  to any other spatial candidate  $k'$ . The transition cost  $\tau$  can be used to evaluate transitions from candidate  $k$  at frame  $j$  to candidate  $k'$  at frame  $j + 1$ .

Given warping path element  $w_t = (i, j, k)$ , we define the set  $N(i, j, k)$  to be the set of all possible values of  $w_{t-1}$  that satisfy the warping path constraints (in particular continuity and monotonicity):

$$N(i, j, k) = \{(i-1, j), (i, j-1), (i-1, j-1)\} \times \{1, \dots, K\} \quad (1)$$

We assume that we have a cost measure  $d(i, j, k) \equiv d(M_i, Q_{jk})$  between two feature vectors  $M_i$  and  $Q_{jk}$ . In our method, each of  $M_i$  and  $Q_{jk}$  is a 4D vector, with two dimensions representing position and two dimensions representing motion. Consequently, a natural choice for  $D(M_i, Q_{jk})$  is the sum of the Euclidean distance of the positions stored in  $M_i$  and  $Q_{jk}$ , and the Euclidean distance of the 2D motion vectors stored in  $M_i$  and  $Q_{jk}$ . The DSTW formulation also assumes that we have a transition cost  $\tau(w_{t-1}, w_t)$  between two successive warping path elements, but in our implementation this transition cost is ignored:  $\tau(w_{t-1}, w_t) = 0$ .

Under the above constraints and specifications, DSTW finds the optimal path  $W^*$  and the global matching score  $D^*$  as described in Algorithm 1.

## 6. REAL-TIME CONSIDERATIONS

The current system was implemented in Matlab, and it runs offline on pre-segmented digits signed by a user. However, the system can be made real-time using existing computer vision algorithms.

Currently, the processing time is 1.08 seconds per frame. Almost all gestures in our test set are between 30 and 90 frames long. In particular, identifying the  $K$  candidate hand regions using skin detection takes 0.33 seconds, and optical flow measurements take 0.75 seconds. The times are measured on a 2GHz Opteron processor. The size of an input frame is  $240 \times 320$ .

Skin detection involves a single table lookup per pixel. To find the  $K$  candidate hand regions we convolve the skin likelihood image with a  $40 \times 30$  uniform mask in order to compute the likelihood sum for each  $40 \times 30$  window. This corresponds to making 2400 operations per pixel. Using integral images [23] we can compute the likelihood sums about two orders of magnitude faster.

Optical flow is computed using Simoncelli’s, et al. method [20] with a three-level pyramid. We found that using only one level is 10 times faster, but we still have not evaluated the accuracy of the results. As an alternative, we can use one of the existing real-time optical flow algorithms. (e.g., [6]).

In summary, we believe that fitting hand detection and optical flow estimation within a real-time framework is relatively straightforward to achieve. We should point out that DSTW itself takes only a small fraction of the total computational time (0.9 seconds per gesture, not per frame) and does not hinder real-time implementation. In effect, DSTW facilitates real-time implementation, by allowing the use of a computationally cheap hand detection module, which is allowed to produce several false detections at each frame.

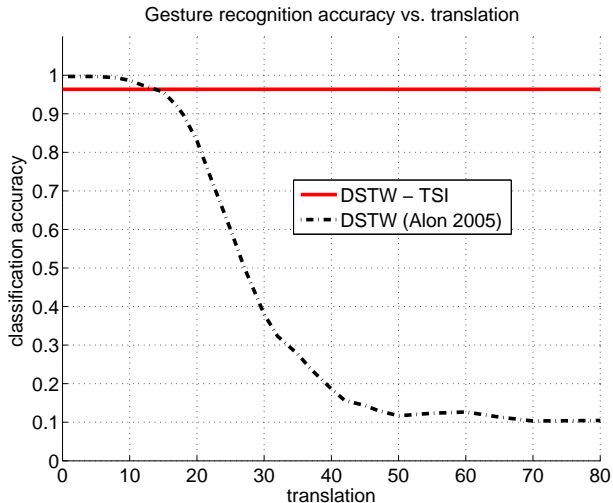
## 7. EXPERIMENTS

To test the proposed method we implemented a hand-signed digit recognition system. For the experiments we have collected video clips of 10 users gesturing the 10 digits in the style of Palm’s Graffiti Alphabet [13] (Figure 2). The video clips were captured with a Logitech 3000 Pro camera using an image resolution of  $240 \times 320$ . The following number and type of sequences were collected for each user:

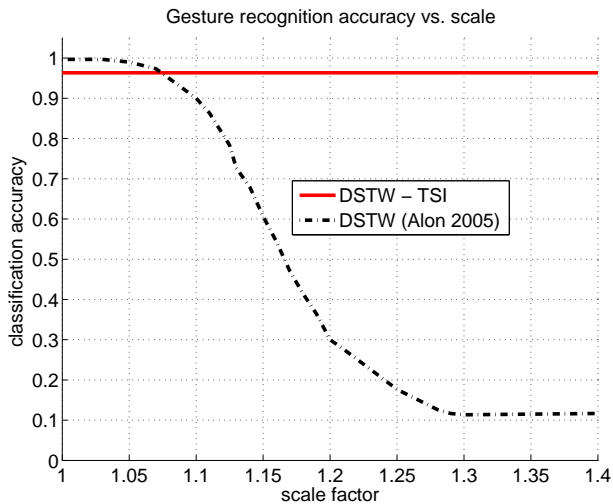
- Training examples: 30 digit exemplars (three per class) per user were stored in the database (See Figure 3). In those exemplars, the users were wearing color gloves.
- Test gestures: 30 digit exemplars (three per class) per user were used as queries. In those exemplars, the users were wearing short-sleeve t-shirts, to highlight the ability of both our method and the original DSTW of [2] to operate under conditions where precise hand localization is hard to achieve. Most gesture recognition methods require users to wear long-sleeve shirts, so that the hands are easily localized using skin color.

In total, there were 300 training video sequences in our database (3 per user per class), and 300 test sequences (3 per user per class) on which recognition accuracy was measured. It is important to note that recognition was performed in a *user-independent* manner: in recognizing a sequence from a specific user, only the 270 training examples collected from the other users were used.

Given a query frame,  $K = 5$  candidate hand regions were detected as described in Section 4. For every candidate hand region



**Figure 4: Classification accuracy versus amount of translation for the method proposed here (denoted as DSTW-TSI, where TSI stands for translation and scale invariance), and the method of Alon, et al. [2]. The x-axis is the amount of translation, in pixels, applied to both the  $x$  and the  $y$  dimensions of the test video sequences, The y-axis is the accuracy rate obtained by each method.**

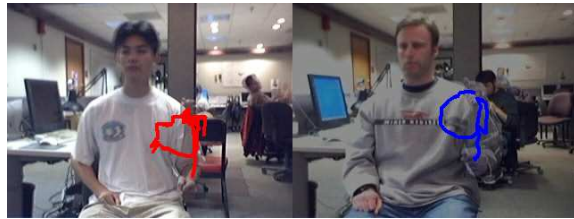


**Figure 5: Classification accuracy versus amount of scale for the method proposed here (denoted as DSTW-TSI), and the method of Alon, et al. [2]. The x-axis is the scale factor, applied to both the  $x$  and the  $y$  dimensions of the test video sequences, The y-axis is the accuracy rate obtained by each method.**

in every query frame, a feature vector was extracted and normalized as described in Section 4. The query digit was then matched with the model exemplars in the database (leaving out the training exemplars from the user performing the query digit). The class of the query was estimated using the one nearest neighbor (1-NN) rule. Examples of a correct match and a false match are shown in Figures 6 and 7 respectively.

The purpose of our experiments is to demonstrate the scale and

Query and model trajectories



**Figure 6: Example query trajectory (left) and corresponding model trajectory (right) for a correct match between two users signing the digit 9.**

translation invariance properties of the method proposed in this paper, and to compare the performance of the proposed method to that of the original DSTW implementation of in [2]. To apply a certain amount  $R_t$  of translation to the input gestures, we simply add  $R_t$  to the  $x$  and  $y$  coordinates of the position of each candidate hand location. To apply a certain amount  $R_s$  of scaling to the input gestures, we simply multiply the  $x$  and  $y$  coordinates of each candidate hand location by  $R_s$ , and we also multiply the 2D motion vector of each candidate hand location by  $R_s$ .

In Figures 4 and 5 we show how the classification accuracy of each method varies as we artificially translate and scale the test video sequences. Naturally, as the method proposed in this paper is invariant to translation and scale, the classification accuracy of our method is not affected at all by this artificial translation and scaling. In contrast, the DSTW method of [2] can only tolerate relatively small amount of translation and scaling, with performance rapidly deteriorating as larger amounts of translation and scaling are used.

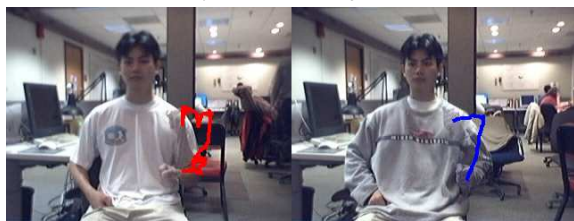
In terms of actual numbers, the DSTW implementation of Alon et al. [2], without the coordinate transformation proposed in this paper, achieves a 99.7% accuracy rate if no translation and scale transformations are applied to the input gestures. However, performance rapidly deteriorates as translation and scale transformations are performed. Applying a translation of 25 pixels in each dimension, or a scaling factor of 1.15 in each dimension, is sufficient to bring the accuracy rate down to 60%. Applying a translation of 35 pixels in each dimension, or a scaling factor of 1.21 in each dimension, brings the accuracy rate down to 28%.

In contrast, the DSTW implementation proposed in this paper achieves an accuracy rate of 96.3%, corresponding to recognizing correctly 289 out of the 300 test gestures. Applying any amount of translation or scaling to the input sequences does not affect the accuracy of the system. By comparing these results to the results of Alon, et al. [2], we see that the proposed method for translation and scale invariance greatly enhances the robustness of the system, by removing the cumbersome restriction that users be positioned precisely to match the translation and scale characteristics of the training exemplars.

## 8. DISCUSSION AND FUTURE WORK

We have presented a method for achieving translation and scale-invariant gesture recognition in complex scenes, in which hand locations cannot be extracted automatically. We have used the Dynamic Space-Time Warping framework (DSTW) of [2] to address the difficulty of achieving unambiguous hand detection. In DSTW, instead of requiring hand location to be precisely detected at each frame, it is sufficient to identify for each frame a relatively short list of candidate hand locations. Those candidate hand locations can easily be generated using simple and well-known skin and mo-

Query and model trajectories



**Figure 7: Example confusion between query digit 3 (left) and model digit 7 (right). In the final segment of the query digit 3 the elbow rather than the hand is falsely matched with the hand of model digit 7.**

tion detection methods.

The contribution of this paper consists in incorporating translation and scale invariance to DSTW. These invariances are achieved by incorporating information from a face detection module, and performing a coordinate transformation that defines the origin and unit vectors according to the location and size of the face. It is important to note that this translation and scale invariance is achieved without incurring any significant additional computational cost over the original translation and scale-dependent DSTW implementation of [2].

There are several interesting issues that were not covered in this paper. First of all, as the method was implemented primarily in Matlab, the system does not quite achieve real-time performance. At the same time, the image processing and pattern matching operations are computationally cheap, and we believe that an optimized C++ implementation can easily achieve real-time performance. Also, it is worth mentioning that, although in our implementation the start and end frames of each gesture were assumed to be known, this assumption can be easily lifted within the framework of DSTW, as described in [1].

In terms of future work, a key type of invariance that we have not addressed is viewpoint invariance. In our method we have assumed that the user is facing the camera, so that we have a frontal view of the gesture. If the user is facing in another direction, a multi-camera system may be used to still obtain a frontal or near-frontal view of the gesture. Alternatively, an articulated human body tracking method can be used to estimate the 3D motion of the user, which can then easily be converted to a frontal view representation. However, reliable tracking of 3D human articulated motion remains a challenging problem.

Finally, another interesting direction is to apply and extend the proposed method for the purpose of recognizing sign language. We are particularly interested in developing tools that let users identify content of interest in video databases of sign language content. The user independence, translation invariance, and scale invariance of the proposed method, as well as the ability of our method to operate in the absence of reliable hand detection, are desirable features that make our method a promising starting point for building such tools.

## 9. REFERENCES

[1] J. Alon, V. Athitsos, and S. Sclaroff. Accurate and efficient gesture spotting via pruning and subgesture reasoning. In *IEEE Workshop on Human Computer Interaction*, pages 189–198, 2005.

[2] 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.

[3] M. Black and A. Jepson. Recognizing temporal trajectories

using the condensation algorithm. In *Automatic Face and Gesture Recognition*, pages 16–21, 1998.

[4] F. Chen, C. Fu, and C. Huang. Hand gesture recognition using a real-time tracking method and Hidden Markov Models. *Image and Video Computing*, 21(8):745–758, August 2003.

[5] A. Corradini. Dynamic time warping for off-line recognition of a small gesture vocabulary. In *Recognition, Analysis and Tracking of Faces and Gestures in Real-time Systems (RATFG-RTS)*, pages 82–89, 2001.

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

[7] T. Darrell and A. Pentland. Space-time gestures. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 335–340, 1993.

[8] W. Freeman. Computer vision for television and games. In *Recognition, Analysis and Tracking of Faces and Gestures in Real-time Systems (RATFG-RTS)*, page 118, 1999.

[9] M. Isard and A. Blake. ICONDENSATION: Unifying low-level and high-level tracking in a stochastic framework. In *European Conference on Computer Vision (ECCV)*, pages 893–908, 1998.

[10] E. Keogh. Exact indexing of dynamic time warping. In *International Conference on Very Large Data Bases*, pages 406–417, 2002.

[11] J. B. Kruskal and M. Liberman. The symmetric time warping algorithm: From continuous to discrete. In *Time Warps*. Addison-Wesley, 1983.

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

[13] Palm. Graffiti alphabet. <http://www.palmone.com/us/products/input/>.

[14] V. Pavlovic, R. Sharma, and T. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(7):677–695, July 1997.

[15] L. Rabiner and B. Juang. *Fundamentals of speech recognition*. Prentice Hall, 1993.

[16] C. Rasmussen and G. Hager. Probabilistic data association methods for tracking complex visual objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(6):560–576, June 2001.

[17] H. 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.

[18] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. In *IEEE Transactions on Acoustics, Speech, and Signal Processing*, volume 34(1), pages 43–49, 1978.

[19] 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 *International Conference on Pattern Recognition (ICPR)*, volume 2, pages 515–519, 2002.

[20] E. Simoncelli, E. Adelson, and D. Heeger. Probability distributions of optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 310–315, 1991.

- [21] T. Starner and A. Pentland. Real-time american sign language recognition from video using hidden markov models. In *IEEE International Symposium on Computer Vision*, pages 265–270, 1995.
- [22] J. Triesch and C. von der Malsburg. Robotic gesture recognition. In *Gesture Workshop*, pages 233–244, 1997.
- [23] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 511–518, 2001.
- [24] 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.