Most Probable Longest Common Subsequence for Recognition of Gesture Character Input

Darya Frolova, Helman Stern, *Member, IEEE*, and Sigal Berman, *Member, IEEE*

***Abstract*—This paper presents a technique for trajectory classification with applications to dynamic free-air hand gesture recognition. Such gestures are unencumbered and drawn in free air. Our approach is an extension to the longest common subsequence (LCS) classification algorithm. A learning preprocessing stage is performed to create a probabilistic 2-D template for each gesture, which allows taking into account different trajectory distortions with different probabilities. The modified LCS, termed the most probable LCS (MPLCS), is developed to measure the similarity between the probabilistic template and the hand gesture sample. The final decision is based on the length and probability of the extracted subsequence. Validation tests using a cohort of gesture digits from video-based capture show that the approach is promising with a recognition rate of more than 98% for video stream preisolated digits. The MPLCS algorithm can be integrated into a gesture recognition interface to facilitate gesture character input. This can greatly enhance the usability of such interfaces.**

***Index Terms*—Classification, dynamic gestures, gesture recognition, longest common subsequence (LCS).**

I. Introduction

GESTURE TRACKING and recognition systems that allow free-air movements in 3-D space have been recently introduced in the home entertainment environment. Several systems (e.g., Wii) require a use of a handheld pointing device to help detect movement in three dimensions. Other systems are based on vision-captured gestures that require markings or physical attachments to the hand or fingers. Unlike these, unencumbered vision systems offer the freedom of gesturing in the free air on a "come as you are" basis. Unencumbered free motion vision systems place increased demands on the development of real-time robust systems capable of operating in real-life environments. One way of overcoming these disadvantages is the use of 3-D vision camera systems.

Dynamic gesture recognition systems based on single video camera output are able to recognize only planar gestures and are most accurate for gestures performed perpendicular to the camera's line of sight. Although many research studies continue attempts to create a robust dynamic gesture recognition system based on such video stream data [1], [2], the fact that the equipment does not allow the capture of all 3-D space-time gesture variations greatly restricts a potential gesture vocabulary design.

While, recently, 3-D cameras have been widely used for hand and full body tracking in 3-D space, gesture recognition methods for sign and character input still require gestures to be performed in a specific embedded 2-D plane. Most of the work in the digit/character recognition field has been done on handwritten digits using paper or a digital tablet. Character input is commonly found with touch screen devices in mobile phone and tablet computer screens.

Hand tracking provides a continuous video stream of the segmented hand but raises the problem of extracting those frames that constitute single gesture trajectories from it. Such a problem is referred to as gesture spotting [3] and is exacerbated because such trajectories may be of different lengths. One solution is warping a test trajectory to match a predetermined template of an exemplary gesture trajectory. Dynamic time warping (DTW) and hidden Markov models (HMMs) are methods that can simultaneously align such signals, using stored deterministic and statistical trajectory representations, respectively. Both methods have been applied successfully to recognize speech [4] and handwriting [5]. Currently, they are also the most used methods for recognition of gestures [2], [6]–[8]. HMMs are able to statistically model a set of samples, while DTW, as an exemplar-based matching procedure, usually requires matching with a plurality of prototypes to get comparable performance.

The precursor to DTW is the longest common subsequence (LCS) method of alignment. The LCS algorithm was developed for matching subword sequences in deoxyribonucleic acid sequences and documents, and only a few authors have employed the LCS for gesture classification [9]–[11]. The LCS method is similar to DTW which attempts to line up a test and template sequence temporally, using feature distance costs. The LCS is more robust to noise and outliers than DTW. Instead of a complete mapping between all points, in the LCS algorithm, a point without a good matching can be ignored. LCS tasks can be solved using a brute-force approach, where all possible subsequences of one of the strings are found. Once all possible strings are found, each string is tested to see if it is a subsequence of the other string [12], [13]. Dynamic programming is used to reduce the time and space complexity of the problem. It uses a 2-D array to store the size of the LCS [13]. For the case of two sequences of n and m elements, the running time and space of the algorithm are both $O(mn)$. The space can be reduced to $O(n)$ if the traceback of longest subsequence is not needed.

Manuscript received September 18, 2011; revised March 26, 2012, July 3, 2012, and August 12, 2012; accepted August 21, 2012. Date of publication October 3, 2012; date of current version May 10, 2013. This work was supported in part by Deutsche Telekom AG. This paper was recommended by Associate Editor S. Zafeiriou.

D. Frolova is with the Telekom Innovation Laboratories at Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel (e-mail: daryaf@bgu.ac.il).

H. Stern is with the Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel (e-mail: helman@bgu.ac.il).

S. Berman is with the Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer -Sheva 84105, Israel, and also with Telekom Innovation Laboratories at Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel (e-mail: sigalbe@bgu.ac.il).

Digital Object Identifier 10.1109/TSMCB.2012.2217324

In this research, a new supervised learning algorithm, which combines the advantages of the classical LCS algorithm with statistical techniques, is presented. The method, called the most probable LCS (MPLCS) algorithm, allows the classification of time-dependent 2-D curves. As such, it is applicable to hand gesture recognition of motion trajectories extracted from video input streams. Although the method can be used for any free-form trajectory, we focus on its use for 3-D space free from handwritten character recognition.

This paper is organized as follows. Section II contains an overview of the relevant research in the field of gesture character recognition. The architecture of the MPLCS algorithm is presented in Section III, followed by a description of the capture and preprocessing of a gesture trajectory training set in Section IV. The methods for probabilistic clustering and gesture template creation are described in Section V. Section VI presents the detailed description of the MPLCS method. Sections VII and VIII contain experimental results and a discussion, respectively. Conclusions are presented in Section IX.

II. Previous Art

In this section, we provide a short review of dynamic gesture recognition methods with the focus on recognition of handwritten characters. Gesture recognition is widely used in human–computer interaction (HCI). Different hardware used for HCIs which includes video cameras, magnetic tracking devices, and depth sensing cameras produces different types of data and may require different recognition methods.

Magnetic devices provide a hand gesture trajectory in a direct manner yet facilitate only encumbered operation. Camera devices facilitate unencumbered gestures, yet hand tracking should be performed to segment the hand trajectory from other motion. An example is the use of color and motion cues for hand tracking in a real-world environment [13]. A unified algorithm based on dynamic space-time warping (DSTW) that combines a trajectory recognition technique together with spatiotemporal segmentation of hand motion from a color video is presented in [1]. Here, the authors require that the users return their hands to the rest position after performing each digit. The use of 3-D sensing devices and stereoscopic sets of cameras not only improves hand tracking but also allows recognition of complex hand postures [6], [14] and reconstruction of full-hand motion and a smooth surface model of the hand [15].

Dynamic gesture recognition for HCI should be performed online (a comparison of online and offline recognition methods can be found in [16]). One of the advantages of online capturing devices (cameras and digital tablets for handwriting) is their ability to capture temporal information (e.g., velocity and acceleration). Raza *et al.* [17] use accelerometer information obtained from wireless Wii Remote device for recognition of unistroke gesture digits.

Trajectories of interest typically appear in continuous streams of motion (gesture sequences or handwriting trajectories). In continuous video streams, a challenging problem is to segment out the portion of the stream that contains only the dynamic gesture information [18]. This problem is known as gesture spotting. Trajectory segmentation at points of minimum velocity (e.g., [8], [19], and [20]) or curvature [9] was found useful for gesture spotting. The gestures were divided into primitives (strokes) that were later grouped in sequences. Gesture segmentation based on trajectory extrema and HMM was used for classification by Li *et al.* [21]. Sclaroff *et al.* [22] used continuous dynamic programming for gesture spotting. The use of strokes to solve the problem of subgestures occurring within a complete gesture and the problem of subdigits was discussed in [1] and [22]. Conversely, the problem of ligatures (the merging of two gestures as one) was dealt with in handwriting by Hu *et al.* [23]. Lee and Kim [18] suggest a two-stage HMM-based procedure based on using a threshold model which is an ergodic model of all trained gestures. A gesture is identified only when the likelihood of the best fitting model is higher than that of the threshold model.

A character can be represented by a set of informational features which may be global or local and based on either static or dynamic properties of the character [16]. The choice of the features may depend on *a priori* information available about the data. Many character classification algorithms use global characteristics like trajectory shape or moments [16], thus reducing the number of features to a few features per trajectory. In [12], a feature vector for each gesture digit was constructed by connecting feature distributions in each direction, and template matching was used for recognition.

Many methods (e.g., alignment algorithms like LCS, DTW, and DSTW) use location information of each point, which is actual point position ($x-y$ coordinates), velocity (features were used, for example, by [19] for handwriting recognition), and, sometimes, acceleration [17]. In [24], 3-D coordinates of body markers were converted into quantized velocity vectors. HMM was used to interpret those sequences, which are directional code words characterizing the trajectory of the motion.

One of the approaches to gesture character recognition is the analysis of similarity measures between gestures represented by time series. One of the possible measures of similarity between two strings is the analysis of their common substrings, and the most popular measure is the LCS [7], [13]. Later, Wang [25] proposed to use the number of all common subsequences in order to capture the maximal common information in sequences. A strong theoretical approach of measuring a similarity of two fuzzy sets (by finding their closest common subsequence) can be found in [5]. A similarity measure that is analogous to our approach combines the deterministic and probabilistic matching scores obtained from the LCS [10].

Generative factored or coupled state models such as HMM or dynamic Bayesian networks [26], [27] assume that the observations are conditionally independent. Such restrictions make it difficult or even impossible to accommodate long-range observation dependences or multiple overlapping features. Discriminative methods such as LCS or conditional random fields (CRFs), on the other hand, avoid the independence assumption and allow nonlocal dependencies, yet they lack the ability to capture hidden states or the statistical strength of methods such as HMM. CRF does not facilitate estimation of the conditional probability of a class label of an entire sequence. In an effort to address these, Wang *et al.* [26] suggests hidden-state CRFs

(HCRFs) that enhance CRF by incorporating hidden states. The latent-dynamic CRF suggested by Morency *et al.* [31] combines the strengths of CRF and HCRF capturing both extrinsic dynamics and intrinsic substructures. In this, they facilitate recognition of unsegmented gestures. We have undertaken to enhance the LCS method which has previously not been considered as a prominent gesture recognition method. Our work provides not only a new LCS method but also a probability-based one. The new algorithm, MPLCS, can also be used for the recognition of unsegmented gestures.

Motion variability is a major problem for all human gesture recognition algorithms. Keskin *et al.* [24] treat the problem of motion variability between users by replacing variable trajectories by several alternative trajectories, each with lesser variability. Wilson and Bobick [28] developed a parametric HMM algorithm by extending the HMM model to include a global parametric variation in the output probabilities of the HMM states. This method is intended for gesture recognition of parameterized gestures, i.e., gestures with a systematic spatial motion variation and not general motion variability. Our approach is different, as we handle large variability at the point level of a trajectory in a statistical manner. Thus, instead of increasing the exemplary template models by replacing one trajectory by several trajectories, where most of the replacement trajectories are similar to the original, we retain the same trajectory and handle variations at the point level by representing it as Gaussian mixture model (GMM) with several components.

III. Architecture of the MPLCS Algorithm

The MPLCS algorithm provides a general solution to the problem of recognizing free-space handwritten character gestures. The recognition method is based on the LCS algorithm which advocates the use of the LCS as an indication of the similarity between a pair of sequences. Unlike the classical deterministic 1-D LCS, a probabilistic LCS algorithm is developed to compare 2-D probabilistic templates to 2-D input probabilistic patterns. The use of a probabilistic 2-D template rather than a deterministic 1-D template allows a more general representation of the data set by taking into account possible trajectory distortions with different probabilities.

The architecture of the MPLCS contains three main modules: Training, Storage, and Recognition. Each gesture g is represented as a separate model which consists of a set of Gaussian components Θ_g and a single template T_g. A set of training gestures is captured and used in a training procedure to learn the model parameters, which are later used in the recognition module. Learned parameters (Gaussian components Θ_g and template T_g) are stored in the Storage module of the system. The recognition process flows as follows: The input trajectory is transformed into a feature vector, which is converted into 2-D patterns according to all Gaussian components. Then, for each fixed g, a pattern is compared to the 2-D template T_g that represents gesture g. The result of this comparison is interpreted as the measure of similarity between the input trajectory and a gesture g. The final classification is performed according to the learned model classification rule. Although the MPLCS algorithm was developed for preisolated gesture digit recognition, it was also tested using a continuous gesture video stream input. The architecture is shown in Fig. 1, with more details presented in Sections IV–VI as follows.

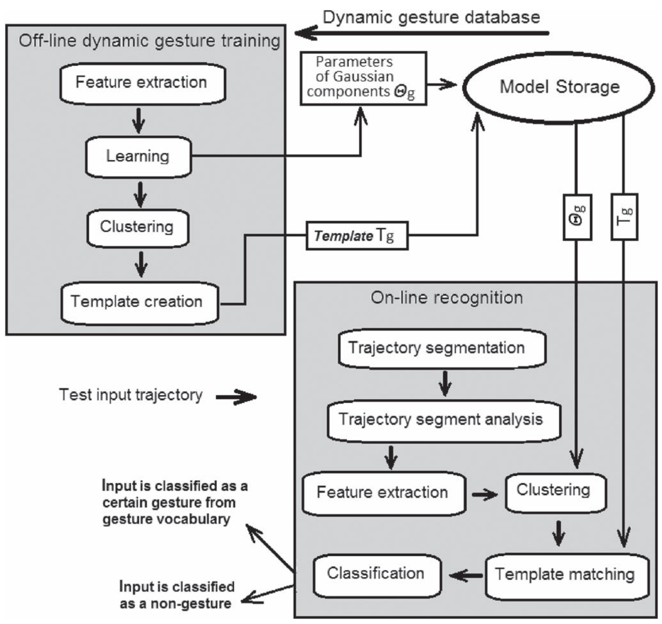


Fig. 1. The system is organized as a combination of three functional components, each containing a set of independent modules. Model learning is performed offline in the dynamic gesture training component. Models are stored in the storage component. Recognition is performed online.

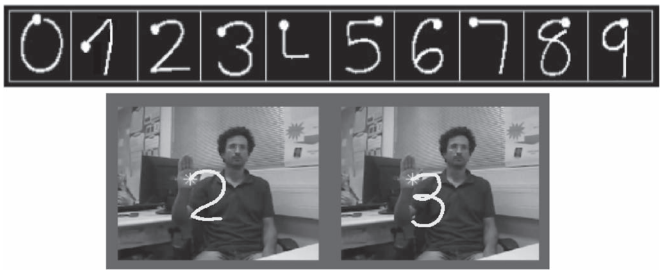


Fig. 2. (Top) Palm's Graffiti Digits [1]. (Bottom) Example digit gestures “2” and “3” as performed by a user (the dot indicates the start position).

IV. Dynamic Gesture Database Capture

To achieve high recognition performance, the proposed method, like other supervised learning algorithms, requires a labeled training set of gestures, preferably performed by different people in order to capture all the performance variations affected by user's age, handwriting, mood, etc. Some free-air gesture symbols are different from symbols written on paper, so existing databases of handwritten alphanumeric characters do not fully satisfy our purpose. Therefore, a database of “0”–“9” digit trajectories performed in the style of Palm's Graffiti Alphabet was created, using consistent starting points as shown in Fig. 2. The data set contains approximately 100 samples of each digit.

The output video was processed to extract the centroid positions of the hand in each frame to create its trajectory. It is desirable for simplicity of the recognition algorithm to represent

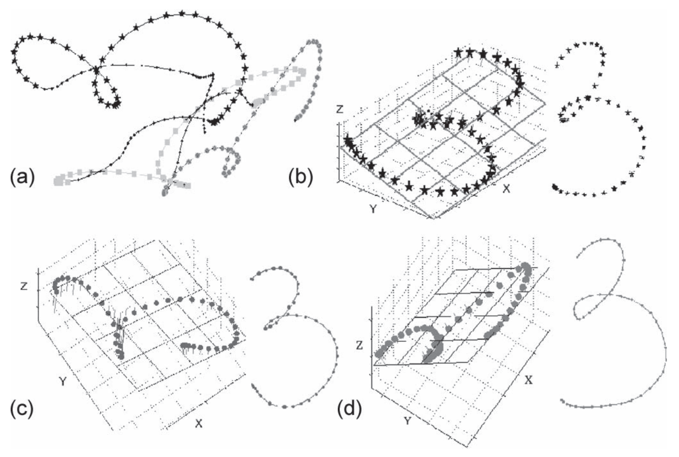


Fig. 3. (a) Original trajectories of the user's hand. The user was performing digit "3" in different directions with respect to the camera. (b)–(d) Three input trajectories were projected onto the best fitting planes.

the gesture trajectory as a sequence of (x, y) coordinates. In the case of 3-D camera, a dimensionality reduction is required.

Two types of data capture systems are considered: a single-color-based camera and a 3-D depth camera. In the case of a single video capturing device, the user's hand is segmented from the background, and positions of the centroids that represent the hand location are computed. To track the hand, a modified Camshift algorithm is used [29]. In this case, the input gesture should be drawn on the plane that is approximately parallel to the camera image plane in order to limit trajectory distortions.

In the case of a depth camera, the hand motion trajectory is described by a sequence of 3-D points. Here, the gesture may be performed from anywhere within the sensor field of view and with an arbitrary angle to the camera plane. Using the fact that the writing is naturally planar, a dimensionality reduction from 3-D to 2-D is performed. To approximate a complete trajectory by a plane, we performed a principal component analysis (PCA) using a sample data set for each of the 3-D trajectory points, to reduce the dimensionality to two. The following conditions must prevail to ensure that "most" of the trajectory does fall within a plane. This is equivalent to ensuring that most of the variation is accounted for by the first two eigenvalues. The conditions that must be satisfied are

$$e_3 \leq \tau_1 * V \qquad \frac{e_1}{e_2} \leq \tau_2.$$

Here, the eigenvalues $\{e_1, e_2, e_3\}$ and total variance V are obtained from the PCA applied to input 3-D points, and $\{\tau_1, \tau_2\}$ are real numbers determined empirically from the data set. The first condition indicates that the third eigenvalue captures less than T_1 of the total variance. The second condition guarantees that the input points belong to a plane and not a line. If both conditions are satisfied, the 3-D points are projected onto a 2-D plane (the first two components obtained by PCA). Fig. 3(a) shows an example of three sequential gesture "3" digits performed at different angles with respect to the camera plane. Their projections onto the fitting plane are shown in Fig. 3(b)–(d).

Free-space hand drawing is a feedback-free operation, so resulting trajectories may have very different shapes, orientations,

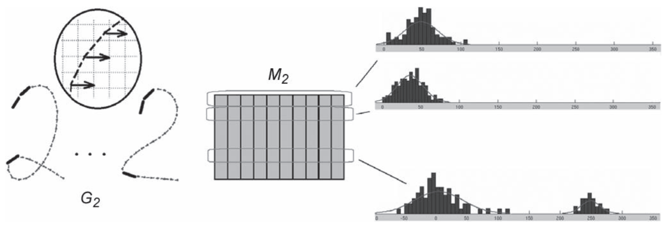


Fig. 4. Illustration of clustering of tangent angles. Adjacent tangent angles usually form distributions that may overlap (two top histograms that correspond to first and second tangent angles). Some angles may form more than one cluster (bottom histogram that corresponds to the 22nd tangent angle). Here, the vectors bifurcate into two different basic directions (where there is a sharp bend or a looped bend).

sizes, and velocities. Personal handwriting styles also have a great impact on the result.

From adjacent frames of a trajectory, local features are chosen as interpoint motion directions and represented by tangent angles (Fig. 4). These features are scale invariant (but not rotation invariant).

V. Offline Dynamic Gesture Training

Given a database of G gesture types, indexed as $g = 1, 2, \ldots, G$, this section describes the process of clustering and template creation for a given gesture type g. Each g consists of S_g samples, indexed $s = 1, 2, \ldots, S_g$. In the notation as follows, t_g^s is a trajectory sample number s, of gesture type g. Gesture samples in the database were performed in 3-D space in free style, without restrictions on speed, resulting in a different number of points (different length of t_g^s).

Step 1) Resample the trajectories $\{t_g^1 \cdots t_g^s \cdots t_g^{S_g}\}$ of the gesture type g, such that each t_g^s has an equal number L_g of equidistant points.

Step 2) Convert each sample t_g^s into a vector of ordered tangent angles $a_g^s = [\alpha_{g,1}^s \cdots \alpha_{g,m}^s \cdots \alpha_{g,L_g-1}^s]^{\mathrm{T}}$. The length of each a_g^s is $(L_g - 1)$. Let the index m represents the mth tangent angle, ordered from the start of the trajectory.

Step 3) (*Construct data matrix*) Denote by M_g a matrix of sample tangent angles for gestures of type g, where a_g^s is a tangent angle column vector for the sth sample

$$M_g = \left[a_g^1 \cdots a_g^s \cdots a_g^{Sg}\right].$$

The size of M_g is $(L_g - 1) \times S_g$. Designate $\alpha_{g,m}$ as a random variable (RV) of the mth tangent angle of gesture g. The sample data for the RV $\alpha_{g,m}$ appear in row m of M_g, which contains the set of samples $\{\alpha_{g,m}^s\}$.

Step 4) (*Gaussian Mixture Model*) The distribution of each RV $\alpha_{g,m}$ may be approximated by a GMM (see Fig. 4).

Unsupervised clustering of the data into components of the GMM is obtained by the expectation maximization algorithm [30]. The decision about correct number of components would require comparing models with different numbers of components. Bayesian information criteria, which penalize the complexity of the GMM, were used to detect the number of components (the model with 1, 2, and 3 components was tested).

The parameters of the GMM for the RV $a_{g,m}$ are the number of Gaussian probability density functions (pdfs) K_m, their weighting factors π_m^k, and the mean μ_m^k and the standard deviation (STD) σ_m^k of each Gaussian function $k = 1 : K_m$ (here and below the subscript g have been dropped for simplicity). For $x = a_{g,m}$

$$p(\mathbf{x}) = \sum_{k=1}^{K_m} \pi_m^k \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_m^k, \boldsymbol{\sigma}_m^k\right) \qquad \forall k : \pi_m^l \geqslant 0; \sum_{k=1}^{K_m} \pi_m^k = 1.$$

The data in row m, which represents the mth tangent angle, may be approximated by more than one Gaussian component, and therefore, more than one mean and STD per row will be computed. For a given g, combine the GMM components for all the RVs $a_{g,m}$ and reindex them as $k = 1 : K_g$, where $K_g \geq (L_g - 1)$. Denote by θ_g^k the parameters of the kth component of the gesture type $g(k = 1 : K_g, g = 1 : G)$

$$\theta_g^k = \left\{\mu_g^k, \sigma_g^k\right\} \quad \Theta_g = \left\{\theta_g^k\right\} k = 1 : K_g.$$

Step 5) For each element $a_{g,m}^s$ of the tangent vector $a_g^s = [a_{g,1}^s \cdots a_{g,m}^s \cdots a_{g,Lg-1}^s]^{\mathrm{T}}$, compute the probability $p_g^s(k,m)$ that the sth sample of the mth tangent angle belongs to the Gaussian component k. Denote this probability as $p(k,m) = p_g^s(k,m)$, for $m = 1 : (L_g - 1)$, $k = 1 : K_g$. For $x = a_{g,m}^s$

$$x \sim N\left(\mu_g^k, \sigma_g^k\right) \quad p(x) = N\left(x : \mu_g^k, \sigma_g^k\right) \quad p(k,m) = p(x).$$

Denote P_g^s as a matrix of size $K_g \times (L_g - 1)$, whose (k,m) entry is $p(k,m)$ denoted earlier.

Step 6) Compute pointwise average of matrices P_g^s over s samples, and denote it by T_g. T_g is a template that represents gesture number g.

VI. Description of the MPLCS Recognition Algorithm

The general flow of the MPLCS algorithm is shown in Fig. 5.

A. Step 1

A test input trajectory t_{input} of length L_{input} is converted into a vector a_{input} of tangent angles. The length of a_{input} is $(L_{\text{input}} - 1)$. Note that the input trajectory is not resampled and L_{input} may not be equal to any of L_g, $g = 1 : G$.

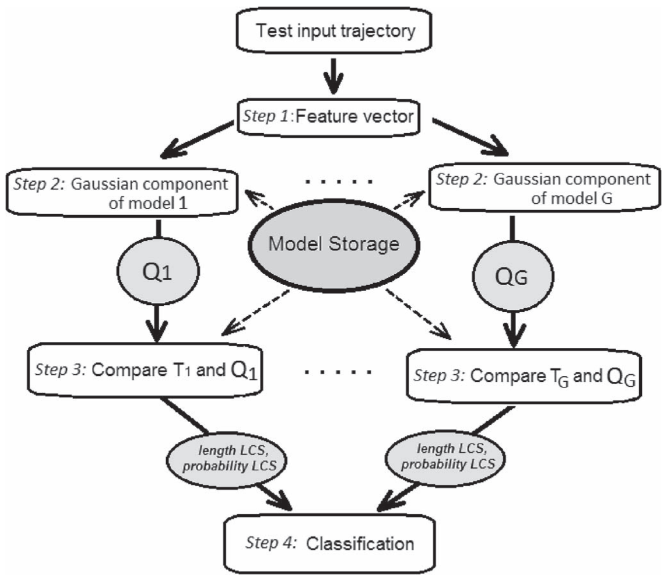

Test input trajectory
Step 1: Feature vector
Step 2: Gaussian component of model 1
Step 2: Gaussian component of model G
Model Storage
Q1
QG
Step 3: Compare T1 and Q1
Step 3: Compare TG and QG
length LCS, probability LCS
length LCS, probability LCS
Step 4: Classification

Fig. 5. The input trajectory is converted into a feature vector, which is then clustered to create a number of probabilistic patterns. Each pattern is compared to its corresponding template and two outputs—the length and the probability of the MPLCS are produced and sent to the classification unit of the algorithm.

B. Step 2

For a given t_{input}, G different matrices Q_g, $g = 1 : G$, are computed (see Fig. 5). For each g, elements of a pattern matrix Q_g are

$$Q_g(k,m) = N(x : \mu_g^k, \sigma_g^k), \text{where } x \text{ is } m\text{th element of } a_{\text{input}}.$$

Each Q_g is a matrix, whose (k,m) entry is the Gaussian posterior probability that an observation of the tangent angle $a_{g,m}^{\text{input}}$ came from a Gaussian component with parameters θ_g^k (i.e., the kth group of tangent angles of gesture g). The size of Q_g is $K_g \times (L_{\text{input}} - 1)$.

C. Step 3

The algorithm compares, for each g, Q_g with its corresponding template T_g. The number of rows in both matrices is the same and corresponds to the number of clusters K_g that characterizes a gesture g. The number of columns of T_g and Q_g denote the number of tangent angles in the template $(L_g - 1)$ and the test input trajectory $(L_{\text{input}} - 1)$. These two numbers may be different, since the length of the test trajectory may be different than the length of the prototype trajectory in the template.

Fig. 6 shows a scheme of comparison of matrices T_3 and Q_3. The goal of this comparison is to find the longest subsequence of common numbers of Gaussian components of feature angles for T_3 and Q_3. The algorithm can be explained by the following. Numbers of Gaussian components are used to quantize trajectory tangent angles. Gaussians may intersect; that is why every angle may be quantized by two or more numbers with different probabilities. A template can also be described as certain "average" trajectory quantized according to the same Gaussian component numbers (and every angle of the template can also be associated with more than one quantization

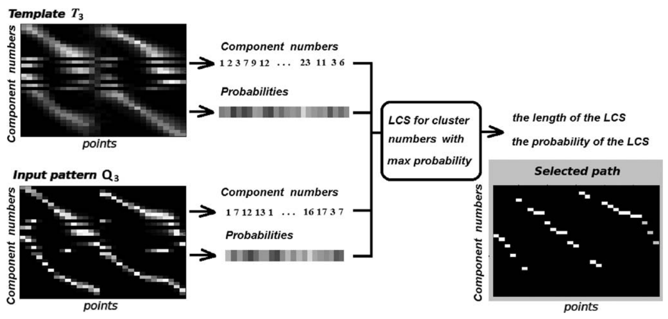


Fig. 6. (Top left) Template matrix for digit "3" is compared to (bottom left image) the input pattern. Image intensities (from 0—black to 1—white) encode the probabilities of the points to belong to the Gaussians. (Right) Longest common path with maximal probability between T_3 and Q_3.

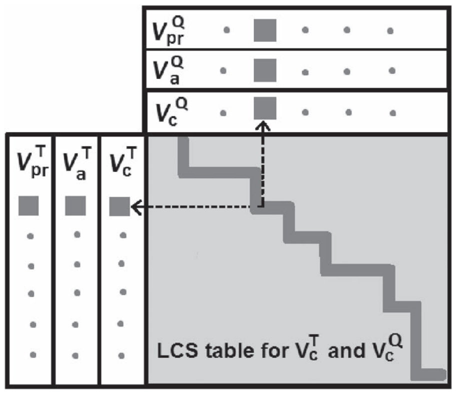


Fig. 7. The LCS table is constructed by dynamic programming. One of the LCSs is plotted by gray curve. Then, tracing back the algorithm keeps not only the common cluster numbers but also their corresponding elements from vectors V_a^{T} and V_a^Q and vectors $V_{\mathrm{pr}}^{\mathrm{T}}$ and V_{pr}^Q.

number). The goal of the algorithm is to find the LCS of quantization numbers between input trajectory and template. Probabilities that correspond to these numbers are multiplied pairwise. The procedure can return an LCS that contains two (or more) quantization numbers that correspond to the same feature angle of the input trajectory (or template). In this case, a quantization number that provides maximal probability (which is a product of corresponding probabilities of the input and template) is selected.

Algorithm Procedure

Matrix Q_g is converted into three vectors (the length of these vectors is equal to the number of nonzero entries of Q_g): Since the matrix contains a large number of zeros, this converts the matrix entries into a vector to be associated with the columns of the LCS table. In addition, two vectors record the column and row indices of each nonzero entry. These three vectors are as follows.

1) V_{pr}^Q—the vector of nonzero entries. Remember that (k,m) entry of Q_g is Gaussian pdf of mth component of a_{input} to be associated with the sample cluster number k.
2) V_c^Q—the vector of row numbers of nonzero entries of Q_g. These row numbers correspond to the Gaussian component indices k with parameters θ_g^k.
3) V_a^Q—the vector of column numbers of nonzero entries of Q_g. These column numbers correspond to the vector a_{input} of tangent angles.

Similarly, matrix T_g is decomposed into three vectors $V_{\mathrm{pr}}^{\mathrm{T}}$, V_c^{T}, and V_a^{T}.

Dynamic programming is used to create a classical LCS table for vectors of cluster numbers V_c^{T} and V_c^Q. Traceback procedures return all tied M LCSs $\{\mathrm{LCS}_1 \cdots \mathrm{LCS}_m \cdots \mathrm{LCS}_M\}$ with length, for example, L', together with the vectors of column numbers and vectors of elementwise products of their probabilities (Fig. 7).

The LCSs found may contain elements with duplicated column numbers. For these elements, the choice of one component number per column is made by selecting the number of the Gaussian component with maximal common probability.

1) If there exists a set of column numbers $\{k_1,\ldots,k_N\}$, such that

$$V_a^{\mathrm{T}}(k_1) = \cdots = V_a^{\mathrm{T}}(k_N) \text{ or } V_a^Q(k_1) = \cdots = V_a^Q(k_N)$$

2) Then select a number k' from $\{k_1,\ldots,k_N\}$ such that

$$k' = k_i, \text{ where } i = \arg\max_j \{V_{\mathrm{pr}}^{\mathrm{T}}(k_j)^* V_{\mathrm{pr}}^Q(k_j)\}, \ j = 1:N.$$

3) Replace $(k_1,\ldots,k_N)$ with k'.

When all the longest common sequences are found and elements with duplicated column numbers $\{k_1,\ldots,k_N\}$ in T_g and Q_g are rejected, the MPLCS chooses the final LCS from the set $\{\mathrm{LCS}_1,\ldots,\mathrm{LCS}_M\}$ as the LCS with the maximal probability $\{V_{\mathrm{pr}}^{\mathrm{T}}(k_j) * V_{\mathrm{pr}}^Q(k_j)\}$. The result of the template matching procedure is a pair of numbers, length and probability of the LCS, denoted as $L(T_g,Q_g)$ and $P(T_g,Q_g)$.

D. Step 4

The bottom part of Fig. 5 shows that the length and probability for the LCSs for each gesture $g(L(T_g,Q_g),P(T_g,Q_g), g = 1:G)$ is compared with a multiclass $(g = 1:G)$ classifier to determine whether the input trajectory is recognized as a particular gesture g^*.

For classification of a gesture g, define the length and probability $L(T_gQ_g)$ and $P(T_g,Q_g)$ as the pair (L_g,P_g). Acceptance thresholds are set for L_g and P_g as $\tau(L_g)$ and $\tau(P_g)$, respectively. For an input trajectory, values of numbers (L_g,P_g) are compared to the thresholds $(\tau(L_g),\tau(P_g))$ for each gesture g. If both (L_g,P_g) are greater than $(\tau(L_g),\tau(P_g))$, for only one unique gesture g^*, then the trajectory is classified as gesture g^*. If two or more thresholds are exceeded, then there are two cases.

a) Recognized gesture is g^* if $g^* = \arg\max(L_g)$, for all g such that $L_g > \tau(L_g)$.
b) If all L_g such that $L_g > \tau(L_g)$ are equal, then recognized gesture is g^* if $g^* = \arg\max(P_g)$ for all g such that $P_g > \tau(P_g)$.

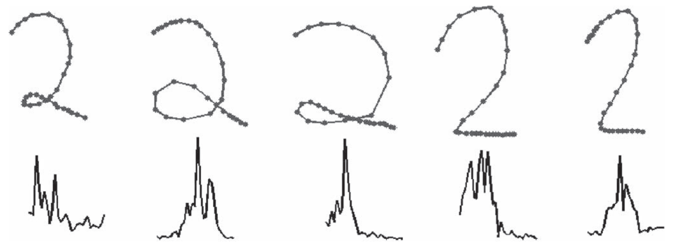


Fig. 8. (Top row) Samples of digit "2" simulated as performed with different velocities. All the samples were correctly classified as "2." (Bottom row) Corresponding velocity profiles.

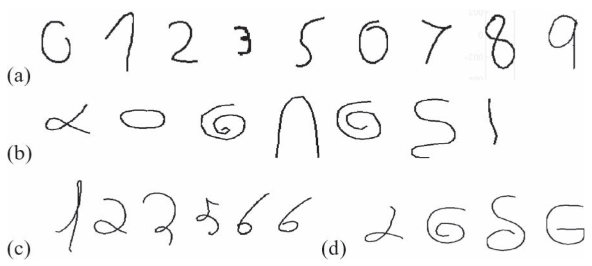


Fig. 9. Recognition results for preisolated digit data set. (a) True positive examples. (b) True negative examples. (c) False negative examples. (d) False positive examples: Nondigits that were recognized as (from left to right) "2," "6," "8," and "6."

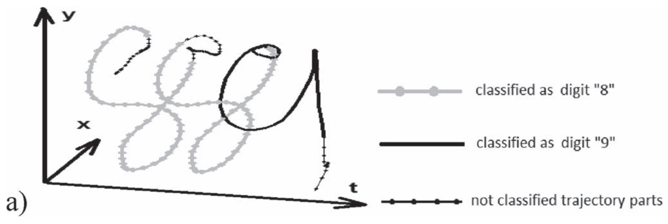


Fig. 10. Example of three correctly classified digits from a stream input trajectory that corresponds to a sequence "889," from 3-D camera.

VII. Testing Experimental Results

A set of experiments was conducted for recognition of both isolated and video-streamed input digits.[1] Also stability of the algorithm to Gaussian noise was examined.

Eight subjects (four males and four females, aged 23–35, right and left handed) participated in all the experiments. Each subject executed 15–20 repetitions of each of ten gesture digits. The repetitions were performed with pauses of a few days between the experiments. Approximately 100 samples of each gesture were used for the training. Data were collected using the PrimeSense 3-D camera.[2] The frame rate was set to 30 fps. Users were asked to draw characters on the air with no constraints on speed or style, except that the start point and trace of the digit are consistent (see Fig. 2). No postprocessing noise reduction was applied. Fifty nondigit (N/D) gestures were included in the testing set. Several were designed to be similar to the numerical digits.

A. Preisolated Digits

A 5×2 cross-validation method was used to test recognition accuracy for the preisolated data set. A confusion matrix is presented in Table I of the Appendix. Fig. 8 shows some of the gesture trajectories. The input trajectories are neither resampled nor aligned in advance and are performed with different velocities (Fig. 8).

An overall recognition accuracy of 98.7% for digits was obtained. Visual analysis show that confusion between the digits 0 and 6 occurred mainly because of the inaccurate closure of the digit by the user. In some cases, a 3 was misrecognized as a 2 because the bottom part of the 3 was not completed. Fig. 9(c) shows a few examples of digits that were rejected by the algorithm, because their LCS probabilities (in the case of digits "1," "2," and "6") or MPLCS lengths (in the case of "3" and "5") were less than predefined threshold. A set of trajectories that do not represent digits (but some of them were very close to digit shapes, see Fig. 9(b)–(d)) was also tested. The algorithm correctly rejected 45 out of 50 nondigit trajectories (90%).

To test user dependence, 120 gesture digits were performed by two subjects that did not participate in the database collection. The recognition accuracy on this data set was 98.3%.

[1]Researchers who would like to use the database are welcome to contact Sigal Berman (sigalbe@bgu.ac.il).

[2]www.primesense.com

We implemented HMM and linear CRF[3]-based classification methods in order to provide a comparison to the aforementioned results. For both methods, trajectories were similarly represented by tangent angles, yet they were resampled to a constant length. Training was done with a subset of gesture sequences selected according to a 5×2 cross-validation method. For the HMM, the Baum–Welch algorithm was used for training. Each HMM is a simple left–right model with two hidden states and eight Markov transition states. The states correspond to equally spaced angle segments from 0 to 360. The average recognition rate for HMM was 89.5% (19% of "2" were misclassified as "3," and 10% of "0" were misclassified as "6"). The average recognition rate for CRF was 99.1% (3% of "5" were misclassified as "8," 3% of "0" were misclassified as "6," and 2% of "6" were misclassified as "0").

B. Stream Input Data

In a second experiment, streams of digits were extracted from the 3-D camera output. An example of the sequence "889" is shown in Fig. 10 with the correct classification. A classifier was run in a sliding window of 40–50 frames (the size of the sliding window was changed dynamically to overcome a subdigit problem, which is described in the following). A confusion matrix for nonisolated data set appears as Table II in the Appendix. The algorithm correctly recognized 200 digits out of 213 (94%). The most problematic digit was "0" (14 correct matches out of 17), which was misclassified in several instances with the digit "6."

[3]Source code can be found at: http://www.cs.ubc.ca/~murphyk/Software/CRF/crf.html

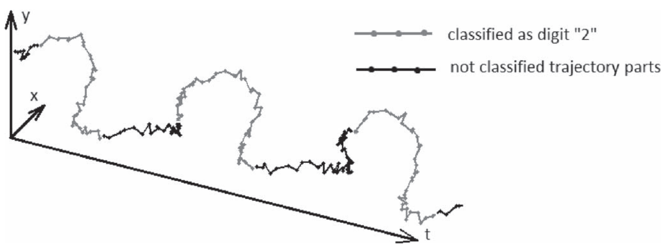


Fig. 11. Example of recognition results for stream input trajectory that corresponds to a sequence “222” obtained from 2-D Web camera.

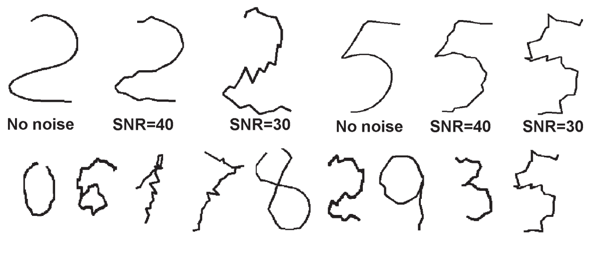


Fig. 12. Few samples of gesture database digits with additive white Gaussian noise applied. (Bottom row) SNR equals 30.

C. Noisy Data

A set of experiments was conducted to test recognition of noisy data obtained from the usual 2-D Web-camera (with image resolution of 640×480 and frame rate of 25 fps) and of preisolated digits with white Gaussian noise added.

Trajectories were extracted using a modified Camshift algorithm [21]. Fig. 11 shows an example of sequence of three digits “2” extracted from a video stream.

A Gaussian white noise with signal-to-noise ratio (SNR) of 40 and 30 was added to the database digit data set (Fig. 12). The clustering rules and templates remain the same, but the acceptance thresholds were decreased when more noise was applied. For the moderate noise level with SNR $= 40$, the average recognition accuracy was 97%; then, the accuracy decreases to 85% for the higher noise with SNR $= 30$.

The lower recognition rates for “1,” “4,” and “7” with noisy data (versus the higher recognition accuracy for clean data) may be explained by the fact that these digits have very simple shapes and, when performed in a natural way (without any noise added), they have very low variation. This is why the templates learned from nonnoisy data handle only a limited number of variations. At the same time, templates learned for more complex digits, like “5” or “8,” are more robust to noise because the training data contain samples performed in different ways.

VIII. Discussion

Our experimental results showed better accuracies for precut gestures (98.7%) than those extracted from streaming video data (94%). This is a higher accuracy than that obtained (89.5%) from the HMM using precut gestures. We did not perform actual tests of the HMM with streaming gestures as the accuracy would be lower since the gesture input has more noise in it. That is to say, the HMM with video-streamed gestures will be less than or equal to 89.5%, which would

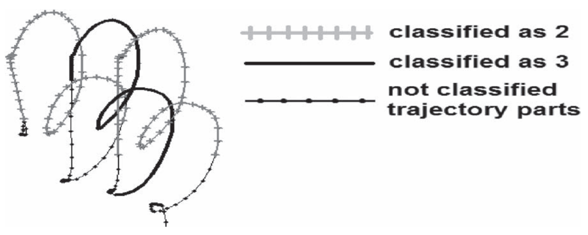


Fig. 13. Without a subdigit problem consideration, two of three digits “3” were classified as “2.”

be worse than the 94% obtained by the MPLCS algorithm for video-streamed gestures. For linear CRF, the classification accuracies obtained for precut resampled gestures (99.1%) are a bit higher than those of the MPLCS (98.7%) for gestures of varying lengths (not resampled). Training of CRFs for large vocabularies can become very computationally expensive [32]. In contrast, training computation cost of MPLCS is affected by the length of the gesture trajectory rather than by the size of the vocabulary. Thus, for large vocabularies, MPLCS may be preferable.

In the absence of pauses between digit gestures, digit spotting is problematic, as all the information in a sliding window is analyzed which leads to higher misclassification rates. To improve recognition rates, the problem of subdigits should be considered. This problem arises when a part of a digit is similar to another digit, e.g., “3” and “2,” “2” and “7,” “9” and “0,” etc. Fig. 13 shows the problem of subdigit recognition.

In the sequence “333,” only the middle “3” was recognized correctly, because a sliding window of the fixed size captured only a part of the digit. The initial size of the sliding window was equal to the mean trajectory length plus three STDs. When a digit that can be a part of another digit was detected, the sliding window was expanded by 1.5 to try and detect the larger digit. For the case shown in Fig. 13, after digit “2” was recognized, the window size was increased, and the larger trajectory portion was analyzed to correctly detect digit “3.”

To overcome the problems of streaming input data, we suggest a procedure as follows. For a real-time recognition system, the input stream will not be segmented *a priori*. In streaming gestures, motion naturally slows down at the beginning and at the end of each gesture character and at places with high curvature. A velocity profile can be used to segment the input trajectory into motion primitives, known as strokes. Fig. 14 shows a segmentation of the sequence “123” into strokes according to local velocity minima. The uncertainty in the choice of sliding window size for continuous streams can thus be avoided. In the following, we describe a procedure for streaming video gestures using the idea of stroke segmentation.

Database trajectories are automatically segmented into strokes, which are trajectory parts between velocity minima. Short strokes should be excluded from analysis. The use of the absolute value of tangent angle rather than the tangent angle itself as a trajectory feature gives a user more freedom in character performance (for example, it allows the user to perform a digit “0” both clockwise and counterclockwise without using an additional template).

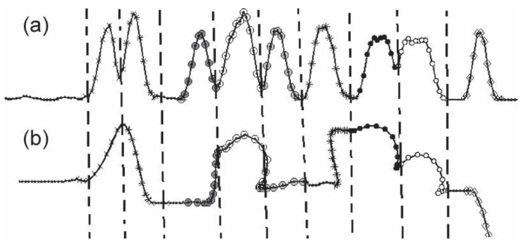


Fig. 14. (a) Velocity profile of the sequence "123" with vertical lines indicating rough segmentation according to local minima; (b) input "123" trajectory with marks indicated segments corresponding to segments in (a).

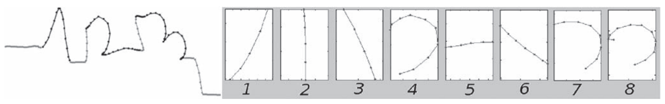


Fig. 15. Sequence "123" followed with the set of strokes extracted.

Each type of each stroke is converted into a template as described in Section V. Each digit may consist of one to three strokes, with some common to a few digits. Since strokes are shorter than digits, templates are smaller.

After the input stream is segmented into strokes according to local velocity minima (see Fig. 15 for an example) each stroke is converted into set of probabilistic matrices, as described in Section V. Then, the MPLCS algorithm is used to compare probabilistic matrices and corresponding templates. Classified strokes are then combined into characters.

IX. CONCLUSION

The MPLCS method is a supervised learning algorithm, which combines learning techniques and a modified LCS algorithm. It allows recognition of free-space hand gestures: digits, characters, etc. The gesture may be performed in any plane with respect to the camera's image plane, for users standing, sitting, or lying within the camera field of view. The experimental results show good recognition rates for gesture digits and low false positive rates. These rates were 98.7% and 94% for preisolated and video-streamed gestures, respectively. This is higher than the recognition accuracy of 89.5% with HMM for preisolated digits. We did not perform actual tests of the HMM procedure with streaming gestures as it can be inferred that the accuracies would be lower since the gesture input has more noise in it. For resampled preisolated gestures, the recognition accuracy of linear CRF was 99.1%, a bit higher than that obtained by the MPLCS for gesture of varying lengths. The computation cost of training the MPLCS is less affected by vocabulary size; thus, it is expected to have an advantage over CRF for large gesture vocabularies.

Tests for preisolated gestures with random noise show the degree of degradation as the SNR is decreased. One of the shortcomings of the approach is the requirement on the user to provide consistent character start points and path directions. This shortcoming is easily avoided by designing multiple templates (a future work task). We also offer a procedure for possible improvement of the recognition accuracy of the MPLCS from streaming video using the idea of gesture primitives.

APPENDIX

TABLE I
CONFUSION MATRIX FOR PREISOLATED DIGIT DATA SET (IN PERCENT). ROWS CORRESPOND TO THE GROUND TRUTH LABELS, AND COLUMNS CORRESPOND TO THE ESTIMATED CLASS LABELS. THE "NOT A DIGIT" ("N/D") ROW SHOWS THE RESULT OF TESTING OF NONDIGIT GESTURE TRAJECTORIES. RECOGNITION RATE FOR DIGITS IS 98.7%

	0	1	2	3	4	5	6	7	8	9	N/D
0	98.5	0	0	0	0	0	1.5	0	0	0	0
1	0	99.5	0	0	0	0	0	0.5	0	0	0
2	0	0	98.8	1	0	0	0	0	0	0	0.2
3	0	0	1	99	0	0	0	0	0	0	0
4	0	0	0	0	98.7	0	0	0	0	0	1.3
5	0	0	0	0	0	97.5	0	0	0	0	2.5
6	1	0	0	0	0	0	99	0	0	0	0
7	0	1	0	0	0	0	0	99	0	0	0
8	0	0	0	0	0	0	0	0	97.9	0	2.1
9	1	0	0	0	0	0	0	0	0	98.6	0.4
N/D	0	1	3	0	1.5	0.5	3	0.5	0.5	0	90

TABLE II
CONFUSION MATRIX FOR NONISOLATED DIGIT STREAM INPUT DATA SET. ROWS CORRESPOND TO THE GROUND TRUTH LABELS, AND COLUMNS CORRESPOND TO THE ESTIMATED CLASS LABELS. THE LAST ROW INDICATES THE TOTAL AMOUNT OF GESTURES USED IN THE TEST. THE "N/D" COLUMN INDICATES THE NUMBER OF TRAJECTORIES CLASSIFIED AS NONGESTURES. RECOGNITION RATE FOR DIGITS IS 94%. NONDIGIT GESTURE TRAJECTORIES WERE NOT TESTED HERE

	0	1	2	3	4	5	6	7	8	9	N/D
0	14	0	0	0	0	0	2	0	0	0	1
1	0	24	0	0	0	0	0	1	0	0	0
2	0	0	26	0	0	0	0	0	0	0	0
3	0	0	1	26	0	0	0	0	0	0	0
4	0	0	0	0	22	0	0	0	0	0	1
5	0	0	0	0	0	18	0	0	1	0	1
6	1	0	0	0	0	0	20	0	0	0	0
7	0	1	0	0	0	0	0	17	0	0	0
8	1	0	0	0	0	0	0	0	17	0	1
9	0	0	0	0	0	0	0	1	0	16	0
Total	17	25	26	27	23	20	21	18	19	17	

ACKNOWLEDGMENT

The authors would like to thank T. Godo for his help with the video capture tasks of this project.

REFERENCES

[1] J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff, "A unified framework for gesture recognition and spatiotemporal gesture segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 9, pp. 1685–1699, Sep. 2009.
[2] F. Chen, C. Fu, and C. Huang, "Hand gesture recognition using a real-time tracking method and hidden Markov models," *Image Video Comput.*, vol. 21, no. 8, pp. 745–758, Aug. 2003.
[3] A. Just and S. Marcel, "A comparative study of two state-of-the-art sequence processing techniques for hand gesture recognition," *Comput. Vis. Image Understand.*, vol. 113, no. 4, pp. 532–543, Apr. 2009.
[4] H. An and D. Kim, "Hand gesture recognition using 3D depth data," in *Proc. of 10th Postech-Kyutech Joint Workshop*, Kyutech, Japan, 2010.
[5] G. Andrejkova, "The similarity of two strings of fuzzy sets," *Kybernetika*, vol. 36, no. 6, pp. 671–687, 2000.

[6] Y.-K. Ahn, M.-W. Kim, Y.-C. Park, K.-S. Choi, W.-C. Park, H.-M. Seo, and K.-M. Jung, "Virtual gesture screen system based on 3D visual information and multi-layer perceptron," *World Acad. Sci. Eng. Technol.*, vol. 59, pp. 405–408, Nov. 2009.

[7] L. Bergroth, H. Hakonen, and T. Raita, "A survey of longest common subsequence algorithms," in *Proc. 7th Int. Symp. String Process. Inf. Retriev.*, 2000, pp. 39–48.

[8] H. Bezine, M. A. Alimi, and N. Derbel, "An explanation for the feature of a handwriting trajectory movement controlled by a beta elliptic model," in *Proc. 7th Int. Conf. Doc. Anal. Recognit.*, 2003, pp. 1228–1232.

[9] H. Chen, O. E. Agazzi, and C. Y. Suen, "Piecewise linear modulation model of handwriting," in *Proc. Int. Conf. Doc. Anal. Recognit.*, 1997, pp. 363–367.

[10] C. Choi, J.-H. Ahn, and H. Byun, "Visual recognition of aircraft marshalling signals using gesture phase analysis," in *Proc. IEEE Intell. Veh. Symp.*, Eindhoven, Netherlands, Jun. 4–6, 2008, pp. 853–858.

[11] K. Tanaka, "Gesture recognition with a focus on important actions by using a path searching method in weighted graph," *Int. J. Comput. Sci.*, vol. 6, no. 2, pp. 14–19, 2009.

[12] J. Hao and T. Shibata, "Digit-writing hand gesture recognition by hand-held camera motion analysis," in *Proc. 3rd Int. Conf. Signal Process. Commun. Syst.*, 2009, pp. 1–5.

[13] D. S. Hirschberg, "Algorithms for the longest common subsequence problem," *J. ACM*, vol. 24, no. 4, pp. 664–675, Oct. 1977.

[14] S. Malassiotis, N. Aifanti, and M. G. Strintzis, "A gesture recognition system using 3D data," in *Proc. 1st Int. Symp. 3D Data Process. Visual. Transm.*, Jun. 2002, pp. 190–193.

[15] G. Dewaele, F. Devernay, and R. Horaud, "Hand motion from 3D point trajectories and a smooth surface model," in *Proc. Eur. Conf. Comput. Vis.*, 2004, pp. 495–507.

[16] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in online handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 8, pp. 787–808, Aug. 1990.

[17] S. A. Raza, M. W. Ahmed, T. M. Madani, M. Tahir, M. I. Khan, and M. Ashraf, "Preliminary evaluation of 3D unistroke gestures—An accelerometer-based approach," in *Proc. Int. Conf. Intell. Inf. Technol.*, 2010, pp. 634–638.

[18] H. Lee and J. Kim, "An HMM-based threshold model approach for gesture recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 10, pp. 961–973, Oct. 1999.

[19] M. Kherallah, L. Haddad, A. M. Alimi, and A. Mitiche, "On-line handwritten digit recognition based on trajectory and velocity modeling," *Pattern Recognit. Lett.*, vol. 29, no. 5, pp. 580–594, Apr. 2008.

[20] P. Mermelstein and M. Eden, "Experiments on computer recognition of connected handwritten words," *Inf. Control*, vol. 7, no. 2, pp. 255–270, Jun. 1964.

[21] X. Li, R. Plamondon, and M. Parizeau, "Model-based on-line handwritten digit recognition," in *Proc. 14th Int. Conf. Pattern Recognit.*, 1998, pp. 1134–1136.

[22] S. Sclaroff, M. Betke, and G. Kollios, "Tracking, analysis and recognition of human gestures in video," in *Proc. 8th Int. Conf. Doc. Anal. Recognit.*, 2005, pp. 806–810.

[23] J. Hu, M. K. Brown, and W. Turin, "HMM based online handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 10, pp. 1039–1045, Oct. 1996.

[24] C. Keskin, A. T. Cemgil, and L. Akarun, "DTW based clustering to improve hand gesture recognition," in *Proc. 2nd Workshop Hum. Behav. Understand.*, Amsterdam, The Netherlands, Nov. 2011, pp. 72–81.

[25] H. Wang, "All common subsequences," in *Proc. IJCAI*, 2007, pp. 635–640.

[26] S. B. Wang, A. Quattoni, L. P. Morency, D. Demirdjian, and T. Darrell, "Hidden conditional random fields for gesture recognition," in *Proc. Comput. Vis. Pattern Recognit.*, 2006, pp. 1521–1527.

[27] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden Markov models for complex action recognition," in *Proc. Comput. Vis. Pattern Recognit.*, 1997, pp. 994–999.

[28] A. Wilson and A. Bobick, "Parametric hidden Markov models for gesture recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 9, pp. 884–900, Sep. 1999.

[29] H. Stern, D. Frolova, and S. Berman, "Hand gesture recognition for TV remote control using tree-based ensemble and LCS classifiers," in *Proc. World Congr. Comput. Sci., Comput. Eng., Appl. Comput.*, Las Vegas, NV, Jul. 2010.

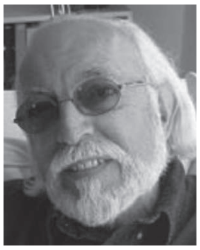
[30] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Stat. Soc. Ser. B (Methodol.)*, vol. 39, no. 1, pp. 1–38, 1977.

[31] L.-P. Morency, A. Quattoni, and T. Darrell, "Latent-dynamic discriminative models for continuous gesture recognition," in *Proc. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–8.

[32] C. Sutton and A. McCallum, "An introduction to conditional random fields for relational learning," in *Introduction to Statistical Relational Learning*, L. Getoor and B. Taskar, Eds. Cambridge, MA: MIT Press, 2007.

Darya Frolova received the B.Sc. and M.Sc. degrees in applied mathematics from Kharkov National University, Kharkov, Ukraine, and the Ph.D. degree in computer from the Weizmann Institute of Science, Rehovot, Israel.

She is a Postdoctoral Fellow with Telekom Innovation Laboratories at Ben-Gurion University of the Negev, Beer-Sheva, Israel. Her publications have appeared in European Conference on Computer Vision, International Conference on Computer Vision, and IEEE Conference on Computer Vision and Pattern Recognition. Her research interests are machine learning, computer vision, 3-D shape reconstruction, hand gesture recognition, and differential equations.



Helman Stern (M'98) received the B.Sc. degree in electrical engineering from Drexel University, Philadelphia, PA, the M.S. degree in engineering administration from The George Washington University, Washington, DC, and the Ph.D. degree in operations research from the University of California, Berkeley.

He has taught at the University of California, Rensselaer Polytechnic Institute, Troy, NY, State University of New York, Albany, and San Francisco State University, San Francisco, CA. He was a Co-primary Investigator for the remote control gesture recognition project with Telekom Innovation Laboratories at Ben-Gurion University of the Negev. He is with the Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer-Sheva, Israel and teaches in the areas of intelligent systems and machine vision. His research interests are human robotic cooperative learning, robotic search, machine learning, computer vision, hand gesture vocabulary design and recognition, and human–humanoid gesture dialog.

Prof. Stern is a member of the IEEE Computer Society.

Sigal Berman (M'91) received the B.Sc. degree in electrical and computer engineering from the Technion—Israel Institute of Technology, Haifa, Israel, and the M.Sc. degree in electrical and computer engineering and the Ph.D. degree in industrial engineering from Ben-Gurion University of the Negev, Beer-Sheva, Israel.

She is a Lecturer and the Head of the intelligent system M.Sc. track in the Department of Industrial Engineering and Management, Ben-Gurion University of the Negev. She is also with Telekom Innovation Laboratories at Ben-Gurion University of the Negev, where she was the Primary Investigator of the gesture recognition for remote control user interface project. Her research interests include robotics and motor control, computer-integrated manufacturing, and human machine interfaces.

Dr. Berman is a member of the IEEE Systems, Man, and Cybernetics Society.