Most discriminating segment – Longest common subsequence (MDSLCS) algorithm for dynamic hand gesture classification

Helman Stern, Merav Shmueli, Sigal Berman*

Department of Industrial Engineering and Management, Deutsche Telekom Labs at BGU, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel

Abstract

In this work, we consider the recognition of dynamic gestures based on representative sub-segments of a gesture, which are denoted as most discriminating segments (MDSs). The automatic extraction and recognition of such small representative segments, rather than extracting and recognizing the full gestures themselves, allows for a more discriminative classifier. A MDS is a sub-segment of a gesture that is most dissimilar to all other gesture sub-segments. Gestures are classified using a MDSLCS algorithm, which recognizes the MDSs using a modified longest common subsequence (LCS) measure. The extraction of MDSs from a data stream uses adaptive window parameters, which are driven by the successive results of multiple calls to the LCS classifier. In a preprocessing stage, gestures that have large motion variations are replaced by several forms of lesser variation. We learn these forms by adaptive clustering of a training set of gestures, where we reemploy the LCS to determine similarity between gesture trajectories. The MDSLCS classifier achieved a gesture recognition rate of 92.6% when tested using a set of pre-cut free hand digit (0–9) gestures, while hidden Markov models (HMMs) achieved an accuracy of 89.5%. When the MDSLCS was tested against a set of streamed digit gestures, an accuracy of 89.6% was obtained. At present, the HMMs method is considered the state-of-the-art method for classifying motion trajectories. The MDSLCS algorithm had a higher accuracy rate for pre-cut gestures, and is also more suitable for streamed gestures. MDSLCS provides a significant advantage over HMMs by not requiring data re-sampling during run-time and performing well with small training sets.

1. Introduction

Gestures may be the most natural way for humans to communicate with their environment and fellow humans, second only to speech. In recognition of this fact, there is an increased interest, among researchers and industrialists, in the development of digital devices that use hand gesture interfaces as a major mode of interaction. Dynamic hand gestures encode information by their temporal trajectories. They are most commonly found in gesture-based applications using semantic based vocabularies where each gesture encodes a meaning. Gesture classification is a critical component of such gesture recognition systems (GRSs).

In a GRS with dynamic gestures we assume that there exists a tracking device (vision based, accelerometer, magnetic, etc.) that calculates a centroidal position of the hand. A sequence of centroids constitutes a gesture trajectory from which descriptive features are extracted. These features are fed into the classification module, which either recognizes the particular gesture or deems it as non-classified. When a gesture is recognized its representative command is sent to a state machine (which is application dependent). The problem of extracting temporal gesture information (frames in the case of a video stream) from a tracker is exacerbated because such trajectories may be of different lengths with unknown locations within the hand-motion data stream.

Existing GRSs can be classified into three types of systems: encumbered, touch-based, and vision-based. For encumbered systems a user must hold a mobile device or an external controller to make gestures. Touch-based systems can accurately map finger/pen positions and moving directions on the touch-screen to different commands. While encumbered and touch-based systems require users to make contact with devices, vision-based systems (using a camera and computer vision algorithms) allow users to make gestures without contact. However, vision-based systems require sophisticated algorithms for detection, segmentation and tracking of the hand. For a recent review of vision based hand gesture recognition see Rautaray and Agrawal (2012).

Dynamic gesture can take on many forms. We use, without loss of generality, digit gestures for testing the proposed algorithm. The gestures used are the digits 0–9, performed by moving the hand in free air (Fig. 1a). This gesture set is standard and considered a difficult set for recognition. This is not to be confused with handwriting classification, often termed optical character recognition (OCR),
where characters (including numerical digits) are written on a stable flat surface such as paper or a digital tablet, with an easily separable background (Fig. 1b). In OCR applications, determining the trajectory and finding its boundaries are quite simple as the digits upon execution are directly placed in storage. Examples of handwriting recognition can be found in Ciresan et al. (2012); Seewald (2012) and Siddharth et al. (2011). In static hand gesture recognition, a similar effort exists for gesture character recognition, e.g., Kasprzak et al. (2012) recognize static hand poses of the finger alphabet in the Polish sign language.

In vision-based GRS the trajectories are in fact motions in free space making trajectory capture much more difficult. Determining trajectory temporal boundaries is a very difficult problem without simplifying assumptions, e.g., that gesture trajectories are initiated by a sound, a button press, or a long interval without motion. Also, for free hand gestures, positional errors are enhanced by hand tremor, improper segmentation from a noisy background, and tracking errors. This makes free hand gesture classification a significantly harder problem than OCR.

Dynamic time warping (DTW) and Hidden Markov models (HMMs) are the most popular methods for dynamic gesture recognition (Corradini, 2001; Zhu et al., 2000; Lichtenauser et al., 2008; Yang and Sarkar, 2006). DTW is a method for sequence comparison, initially used in various applications such as Deoxyribonucleic acid (DNA) comparison in microbiology, comparison of strings of symbols in signal transmission, and analysis of bird songs and human speech (Sankoff and Kruskall, 1983). One solution to the problem of gestures of varying lengths is to use DTW to warp the tested trajectory to match a predetermined template of an exemplary gesture trajectory. HMMs use statistical trajectory representations, and have been applied successfully to recognize speech (di Martino, 1985), and handwriting (Bahlmann and Burkhardt, 2004). Recently, DTW has been out of favor because HMMs are able to statistically model a set of samples, while DTW is an exemplar-based matching procedure, usually requiring matching with a plurality of prototypes to get comparable performance. Recurrent Neural Networks (RNNs) have also been used for recognition of temporal inputs (Murakami and Taguchi, 1991). The main disadvantage of HMMs and RNNs is the requirement for a large number of test samples and long training times to calibrate the model. The precursor to DTW is the longest common subsequence (LCS) method of alignment. The use of the LCS as a similarity measure between pairs of full or partial trajectories (segments) which reside in different dimensional spaces, has a natural advantage over the usual similarity measures such as Euclidean, Manhattan, Mahalanobis, etc., which require fixed dimensional data strings. A second advantage of the LCS measure, as reported by Vlachos et al. (2002) who have tested this non-metric similarity function, is that the LCS measure is very robust to noise, unlike the Euclidean and DTW distance measures. This is mainly because in determination of the LCS, noisy elements of the trajectories need not be matched, whereas in Euclidean and DTW all elements of a trajectory sequence must be paired, even to outlier points, and to unwanted variations at the beginning and end of the trajectory. Moreover, the LCS computation is more efficient than the DTW computation, which requires a costly Lp Norm to be computed. Morris and Trivedi (2009) substantiate the advantage of the LCS distance measure, by finding the LCS measure consistently superior, in an evaluation of six trajectory distance measures.

The identification of gesture temporal boundaries (the gesture start and end points) is termed gesture spotting. There are two approaches to gesture spotting: (i) first spot and extract the gesture from the data stream and then send it to a classifier for recognition, and (ii) solve the spotting and recognition problems simultaneously. In this paper we take the second approach and treat spotting and recognition as a combined process. For this we introduce an advanced adaptive windowing management system that runs in real time. We suggest a classification algorithm based on the LCS algorithm and on identifying a set of most discriminating sub-segments (MDSs) of gesture trajectories. Accordingly, we name our algorithm the MDSLCS. A key concept in the MDSLCS algorithm is that the automatic extraction and recognition of the MDSs allows for a more discriminative classifier, than extracting and recognizing the full gestures themselves. Modeling each gesture as a sequence of MDSs (not necessarily collectively representing the entire gesture) is similar to the idea of phonemes and strokes for speech and handwriting recognition, respectively. The MDSs are spotted using windows, which are smaller than the full gesture itself. Since segments are of different lengths, the usual similarity metrics such as Euclidean, Manhattan, Mahalanobis, Chebychev, etc. cannot be used. Instead, we quantify the similarity between pairs of segments using the LCS algorithm. In Alon et al. (2005), sub-gestures are used for classification but the sub-gestures used are full gestures that appear as parts of larger gestures (for example the digit 1 imbedded in the digit 4). In Malgrieddy (2010) a sub-gesture modeling approach is also used, whereby each gesture is completely represented as a sequence of smaller sub-gestures (a sub-gesture is a group of consecutive points in a gesture trajectory whose union constitutes the full gesture). The classifier is a variant of the HMM using a hidden conditional random field model. Both differ from our method, which employs MDSs as small "patches" that form a cover of the full gesture (which need not be disjoint nor complete).

We use the LCS measure as our main analysis engine, and employ it in three different (yet connected) methodologies: (i) clustering gesture trajectories, (ii) learning MDSs from a training set of gesture trajectories, and (iii) within the MDSLCS classification algorithm for recognizing gestures through detection of MDSs.

**Fig. 1.** Digit classification: (a) typical free-hand gesture scenario, and (b) typical handwriting input for OCR.
• Clustering gesture trajectories – In a preprocessing step, gestures that have large motion variations between subjects are replaced by several forms with lesser variation. We learn these forms by sequential clustering of a training set of gesture samples, where we employ the LCS to measure the similarity between pairs of gesture trajectories.

• Learning sub-segments – The MDSs for each gesture are learned through an automated method using a training set of gesture samples and the LCS similarity measure.

• The MDSLCS algorithm – Within the MDSLCS algorithm, a modified LCS algorithm is used for gesture classification.

As the LCS measure is not a classifier as such, it is modified by using templates of pattern models, and a gesture segment verification module. In Chen et al. (2007) it is claimed that trajectory based LCS (as well as DTW) is sensitive to shifting and scaling in the amplitude dimension. To avoid this we use the derivative of the gesture trajectory (motion vectors). These features are location and size invariant, yet are sensitive to rotation, which is required for gesture sets containing rotated movements (for example, down and up are rotated 180°). An adaptive window is employed where the window parameters (length and relative position) are determined by the successive results of multiple calls to the LCS classifier.

The rest of this paper is organized as follows: Section 2 reviews the LCS measure and literature related to the suggested LCS based algorithms. Section 3 presents the methodology for the MDSLCS algorithm, starting with its architecture, clustering of gesture trajectories, automated extraction of MDSs, and the MDSLCS gesture classification. The section concludes with an analysis of the computational complexity of the MDSLCS algorithm. Results of testing the MDSLCS classifier on a set of free space drawn digit gestures are described in Section 4. Conclusions are provided in Section 5.

2. LCS and related literature

In this section we present three main methodologies, (i) clustering gesture trajectories, (ii) learning MDSs from a set of gesture trajectory samples, and (iii) a MDSLCS algorithm for gesture recognition through the detection of MDSs. The central aspect connecting these three methods is the use of the LCS as a similarity measure. We start by providing a brief review of the LCS measure and LCS related research that is most closely related to ours.

2.1. The LCS measure

The LCS measure (others may use the acronym LCSS) was developed for matching character sequences in DNA strings and text documents. 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, because instead of a complete mapping between all points, a point without a good matching can be ignored.

The LCS problem is one of finding a common substring of two candidate character strings. A dynamic programming approach is used to reduce the time and space complexity of the problem. It uses a 2-dimensional array to store the size of the longest common subsequence. The classic dynamic programming solution to the LCS problem was proposed by Wagner and Fischer (1974). A comparison of various versions appears in Bergroth et al. (2000).

The standard method of computing the length of a LCS is a “bottom-up” dynamic programming approach based on the recurrence expressions shown in Eq. (1). The inputs are two string segments of length m and n. An array L is filled recursively with values of L(ij)

\[
L(i,j) = \begin{cases} 
0 & \text{if } i = 0 \text{ or } j = 0 \\
L(i-1,j-1) + 1 & \text{if } i,j > 0 \text{ and } a_i = b_j \\
\max\{L(i-1,j), L(i,j-1)\} & \text{otherwise}
\end{cases}
\]

for i = 0,1,...,m, j = 0,1,...,n. Each value (ij) represents the LCS of the pair of substrings, of length i and j.

At the end of the algorithm the length of a LCS of the two input string segments is read off from L(m,n) in the array. The complexity of the algorithm is O(mn) (Corman et al., 2001). The actual segment LCS is found by a “backtracking” approach starting from L(m,n).

One can calculate the percent of commonality or similarity between the two input sequences using Eq. (2).

\[
S(i,j) = L(m,n) / \min(m,n)
\]

2.2. Related LSC papers

Trajectory clustering is a popular topic and has been addressed by many researchers. For a survey see Liao (2005), and the introduction in Piciarelli and Foresti (2006). We found three projects that use or discuss the LCS measure for clustering trajectories. The method of Buzan et al. (2004) unlike ours, uses a modified version of an agglomerative-dendrogram based hierarchical clustering algorithm. Vlachos et al. (2002) use the LCS measure to index trajectories for similarity retrieval. Yanagisawa and Satoh (2006) discuss trajectory clustering using LCS.

As for the extraction of segments, primitives, or sub-gestures from gesture trajectories, we have not found any work using the LCS.

We have not found any work that uses LCS directly as a classifier. However, a number of researchers have integrated the LCS measure in various ways within the architecture of their recognition systems. Oikonomopoulos and Pantic (2007), in a paper on human body recognition, use the LCS measure to compare sets of trajectories of different lengths. The LCS measure is not used directly as a classifier, but is used to define new kernels for a Relevance Vector Machine (RVM) classification scheme (A RVM is a probabilistic sparse kernel model identical in functional form to the Support Vector Machine (SVM)). Tanaka (2009) uses optimal flow and Principle Component Analysis (PCA) to determine a feature vector. These are compared using a LCS similarity measure. The method was tested on a small gesture vocabulary (GV) of four gestures with accuracies of 64–80% for individual gestures. Kuzmanic and Zanchi (2007), perform gesture classification with a k-NN algorithm using DTW and LCS as similarity measures. This is a static hand gesture recognition system without the need for gesture spotting. Hsu et al. (2010) have proposed an accelerometer based system, for 3D handwriting recognition using the LCS to compute the length between a sample and a keyword which is input to a SVM for the classification task. In Stern et al. (2010) a LCS classifier is used to classify directional gestures for an IPTV system. This classifier is a forerunner to the MDSLCS algorithm but only in its use of the LCS measure in a classifier.

3. Methodology

3.1. Overview and system architecture

The architecture of the proposed gesture recognition system is presented in Fig. 2. The architecture is comprised of off-line learning algorithms and an on-line recognition algorithm.

Off-line clustering and learning of MDSs is based on a set of gesture training samples. The samples are represented by temporal motion directions, parameterized by tangent angles. Without loss of generality we also refer to these as patterns. A sequential
clustering procedure is used to learn several exemplar patterns, i.e., gesture forms for each gesture. MDSs are determined for each of the resulting gesture forms. The aim is to find for each gesture, a set of segments that are most discriminative with respect to the segments of all other gestures. This set, termed the MDS set, is a cover of the gesture form. The cover is not required to be complete or disjoint. The problem is to learn the best collection of MDSs for all gestures. These MDS patterns are referred to as templates and stored for use by the on-line MDSLCS classification algorithm (Fig. 2b).

On-line classification requires a window management strategy for spotting and recognizing MDSs in the data stream. The approach is twofold; first detect a primary MDS, and note its associated gesture form, and second search for adjacent MDSs (referred to as secondary MDSs) that belong to the same gesture form. Recognition of a MDS uses the LCS algorithm which compares a candidate segment to each exemplary MDS in a stored template. Upon recognition of the primary MDS, a verification procedure is used to determine if there is enough evidence to substantiate the recognition of a gesture form. The procedure uses the recognition of the primary MDS to anchor a location in the input data sequence where a candidate gesture is believed to be present. Next, it searches for the secondary MDSs associated with the candidate gesture. Each of the secondary MDSs are known a priori to be located before or after the primary MDS, and if all are found, they verify the recognition of the gesture. Otherwise, the system attempts to retrieve a new primary MDS.

3.2. Clustering gesture trajectories

Many gestures have large motion variability because of differences within and between subjects. Such gestures can be represented by more than one exemplary pattern (gesture form). For each gesture, a plurality of forms is learned from a data set of sample gesture trajectories using the LCS similarity measure. A multi form (MF) algorithm is used to determine, for each gesture g, one or more clusters with exemplary patterns that are subsequently used for pattern matching in the MDSLCS gesture classification algorithm. The algorithm is based on a sequential clustering procedure (Thedoridis and Koutroumbas, 1999), which assigns samples to clusters. The similarity, \( S(i,j) \), between two patterns i and j as determined by the classical LCS measure (described in Section 2) uses the Min operator in the denominator (Eq. (2)). However, for our purposes we use a Max operator (Eq. (3)) to compute a LCS similarity measure, \( S(i,j) \).

\[
S(i,j) = \frac{L(m,n)}{\text{Max}(m,n)}
\]  

Both Eqs. (2) and (3) provide similarity measures in the range 0...1 (1 represents a perfect match, 0 implies i and j have no overlap). Eq. (3) is stronger in capturing the effect of pattern pairs of different lengths. Consider trying to detect the word "cat" in a document also containing catapult. For \( i = \text{CAT}, j = \text{CAT-AULT} \), using the classical LCS measure (Eq. (2)) yields \( S(i,j) = 3 \) whereas using the Max version (Eq. (3)) yields \( S(i,j) = 3/8 = .38 \) indicating less than a perfect match. As a side-note, the LCS similarity function is often referred to as a distance function, since the distance \( D(i,j) \) can be defined based on \( S(i,j) \), \( D(i,j) = 1 - S(i,j) \).

**Algorithm MF: Multi-form clustering** – The input to the MF algorithm is a set of training samples (gesture trajectories) for each gesture \( g \) (\( g = 1,...,G \)). Each sample is a motion trajectory, represented by a sequence of tangent angles of the inter-point motion vectors, \( x = \{x_1, x_2, ..., x_n\} \) where \( x_j \) is the jth tangent angle. Let \( Q^k = \{x^k_u : u = 1,...,s^k\} \) be the set of samples of gesture g, where \( x^k_u \) is the uth sample of gesture g. All samples are resampled such that their length, m, is equal. Since the number of clusters is unknown a priori, the clusters are formed dynamically by using a distance threshold \( \epsilon \). Let \( n^k \) be the number of samples currently as-

---

**Fig. 2.** The architecture of the MDSLCS gesture recognition system. (a) off-line clustering and learning of MDSs, and (b) on-line MDSLCS classification algorithm.
signed to cluster \( k \) such that all within cluster sample pairs \( x_i^k \) and \( x_j^k \) are “highly” similar, i.e., \( S(i,j) > \tau \). Let the representative member of cluster \( k \) be \( x^k \), where \( i^k \) is the sample number that yields the maximum average similarity as determined by Eq. (4).

\[
i^k = \arg \max_{i \in [1, \ldots, n^k]} \mu(i), \quad \mu(i) = \frac{\sum_{j \in [1 \ldots n^k]} S(i,j)}{n^k - 1}
\]

where, \( \mu(i) \) is the average similarity of sample \( x_i^k \) to all other samples in cluster \( k \).

For a new sample \( x_i \), if \( S(a, i) > \tau \) for all \( k > \tau \) then \( x_i \) is assigned to cluster \( k \) as \( \arg \max \{ S(a, i) : \forall k \} \). Otherwise, if \( \mu(i) < \tau \) (low similarity) then a new cluster is formed and sample \( x_i \) is assigned to it. Once all samples have been assigned, small clusters are considered as outliers and removed. In addition clusters close to each other are merged.

At the end of the MF algorithm, after the training samples for all gestures have been processed, the clusters for all gestures are re-indexed to \( k = 1, \ldots, K \), where each cluster \( k \) is composed of a set of samples \( X^k = \{ x_{u}^k : u = 1, \ldots, n^k \} \). In what follows we use the term cluster \( k \) when referring to a set of samples (of a gesture), and form \( k \) when referring to a representative member of cluster \( k \). We retain the gesture – cluster/form information, by storing an indicator variable \( h(k) \) for each \( k \), where \( h(k) = g \) indicates that the \( k \)th cluster and its representative sample belong to gesture \( g \). Since the gestures are split into multiple forms, the set of forms associated with a gesture \( g \) is shown in Eq. (5).

\[
F_g = \{ k : h(k) = g \}, \quad \forall g = 1, \ldots, G
\]

3.3. Automated extraction of the most discriminating segments (learning MDSs)

The task is to learn the MDSs set for every form, that is, the set of segments that best represents the form and best differentiates it from all other gestures. A MDS set, associated with a gesture-form \( k \), is denoted as MDS\( ^k(b) \), where \( b = 1 \) and \( b > 1 \) represent the indices of the primary and secondary MDSs, respectively. Finding the MDSs for each gesture-form in the gesture vocabulary is conducted off-line. The process uses a training set and employs the LCS algorithm to compare the similarities between segments. The MDSs are stored as templates for later use by the MDSLCS classifier.

First we construct a set of segments for all training samples. Then for each gesture-form, \( k \), we find the primary MDS, MDS\( ^k(1) \); that segment that is most dissimilar from all segments of all other gesture samples (notice samples that belong to different forms of the same gesture are taken into account in this comparison). Finally, for each MDS\( ^k(1) \), we find two or more secondary MDSs. The steps of the procedure are; (1) create segments, (2) find the similarity between sets of segments, (3) find primary MDSs and (4) find secondary MDSs.

3.3.1. Create segments

Given a sample \( x_i = \{x_1, x_2, \ldots, x_m\} \), define a segment of \( x_i \), as \( \{x_{j-1}, x_j, \ldots, x_{j+\beta}\} \). Such a segment may be denoted by the tuple \( (j, \beta) \), where \( j \) represents the start position of the segment and \( n \) its length. For a sample of length \( m \), the number of possible segments (including segments of length 1) is \( m(m-1)/2 + m \).

Let \( \text{seg}_i^k \) define the \( i \)th segment extracted from \( x_i^k \). To avoid overly small and large segments we provide \( L, U \) as the lower and upper limits, respectively, on the length of any extracted segment. These limits are selected according to a percentage of the average length of a gesture. When using these limits let \( n^k \) represent the number of generated segments. The following algorithm, denoted as Algorithm SG (segment generator), generates the set of sample segments \( \text{seg}_i^k = \{ \text{seg}_i^k(i) : i = 1, \ldots, n^k \} \) for all samples \( x_i^k \) in cluster \( k \).

**Algorithm SG: segment generator**

for each cluster \( k \) do

load \( Q^k = \{ x_{u}^k : u = 1, \ldots, n^k \} \), (set of samples (size \( n^k \)) of cluster \( k \))

for each \( x_u^k \) do

\( x \leftarrow x_u^k = \{ x_1, x_2, \ldots, x_m \} \)

\( L \sim m/3, U \sim m/2 \) (calculate range of segment length limits, \( \text{start} \) segment extraction)

\( i = 1, j = 1, N = j + L \) (segment no, position no, segment length)

for each \( (j = 1, \ldots, m - L) \) do (segment length filter, \( \text{for segment set} (n = j + L, \ldots, \text{Min}(j + U, m)) \) do)

\( \text{seg}_i^k(i) = \{ x_j, x_{j+1}, \ldots, x_{j+n^k} \} \) (the segment)

\( i = i + 1 \)

end for

\( n^k \leftarrow i \)

end for

end for

3.3.2. Calculate the similarity between sets of segment samples

Using \( \text{seg}_i^k \) from algorithm SG, define \( Q^i(i) = \{ x_{u}^k : u = 1, \ldots, n^k \} \) for all the \( i \)th segments in cluster \( k \), as shown by Eq. (6). Using Eq. (3) one may find \( S_{a,b}^i(i,j) \) the similarity between the pair of segments \( \text{seg}_a^i \) and \( \text{seg}_b^i \). Similarly, let \( S(i,j) \) represent the mean similarity between the pair of segment sets, \( Q^i(i), \ Q^i(j) \), calculated according to Eq. (7).

\[
Q^i(i) = \{ \text{seg}_i^k(i) : u = 1, \ldots, n^k \}
\]

\[
S(i,j) = \frac{1}{2} \sum_{u=Q^i(i); v=Q^j(j)} S_{a,b}^i(i,j)
\]

where \( z \) is the total number of sample pairs \( n^p \cdot n^q \).

3.3.3. Find primary MDS (inter gesture – form analysis)

The primary MDS (determined for each form) is the segment that best represents it, i.e., is most dissimilar from all other segments of all other forms of the other gestures. This is determined by a three-step procedure. Step 1 – for the segments \( Q^i(i) \) calculate an overall average similarity, \( S(i) \) (Eq. (9) below), as the cumulative average similarity of the segments in \( Q^i(i) \), to all other segments of all forms not associated with gesture \( g \) where \( h(k) = g \). Step 2 – from all the segment sets \( Q^i(i) \), \( i = 1, \ldots, n^k \), find the segment set \( Q^i(i^*) \) that is most dissimilar. Step 3 – for the most dissimilar set find the exemplar segment and denote it as MDS\( ^k(1) \). Denote \( S(k,q) \) (of size \( n^k \cdot n^q \)) as the segment similarity matrix for the pair of forms \( (k,q) \), with common matrix element \( S_{a,b}^i(i,j) \). For each \( k \), construct the “super matrix” \( S(k) \) of size \( n^k \cdot \sum_{q=1}^{n^q} n^q \) comprised of an ordered sequence of matrices \( S(k,q) \) as shown in Eq. (8).

\[
S(k) = \{ S(k,1), \ldots, S(k,q), \ldots, S(k,K) \} : k = 1, \ldots, n^k
\]
Algorithm PMDS: Find MDS(1) for all forms

for \( k = 1, \ldots, K \) do (forms \( k \))

Load \( S(k) \) (Eq. (8))

Do Steps 1–3

Step 1. For each row \( i \) of \( S(k) \) (which represents \( Q^k(i) \) (Eq. (6)), the set of segments \( i \) of cluster \( k \). Calculate an overall average similarity \( s^k(i) \), to all segments in all forms, except those associated with gesture \( g \) determined by \( h(k) = g \) as shown in Eq. (9).

\[
\tilde{s}^k(i) = \frac{1}{V} \sum_{j: v = \tilde{v}^N} s^{\tilde{k}}(i,j) \quad \forall i \in N^k
\]  

(9)

where, \( V = \bigcup_{i=1}^{N^1} \bigcup_{j}^{N^k} \bigcup_{g}^{N^g} \) (set of all segment indices, except those relating to forms associated with gesture \( g \))

\( N^g = \{i = 1, \ldots, n^g\} \) (set of segment indices of form \( g \))

\( F_g = \{k: h(k) = g\} \) (set of all forms associated with gesture \( g \) (Eq. (5))

Step 2. Find \( Q^k(i^*) \), \( i^* = \arg \min_{i} (s^{k}(i) / v \in n^k) \)

Step 3. Find MDS\((1)\)

MDS\((1)\) is the exemplar segment from all the sample segments in cluster \( Q^k(i^*) \). This is found using Eq. (4) for segment cluster \( k \).

end for

3.3.4. Find secondary MDSs (intra gesture – form analysis)

The secondary MDSs are determined such that they best represent the variability within the form, i.e., the segments that are most dissimilar from each other and from the MDSs previously generated for the form, and are temporally distinct from each other. For simplicity we will describe the algorithm for generating the pair of secondary MDSs, MDS\((2)\) and MDS\((3)\). The extension to additional pairs is straightforward and obtained by repeating the algorithm for each pair. The number of required secondary MDS pairs is determined empirically. The algorithm has three steps: Step 1 – segment set extraction, Step 2 – MDS pair selection, and Step 3 – calculation of exemplar segments.

Step 1: Remove all segment sets \( Q^k(i) \) that have \( s^{\tilde{k}}(i) \) that overlaps MDS\((1)\) (for \( i > 3 \): overlap with all already selected MDSs). For very short samples this process may eliminate all the segments. In such a case the first and last segment sets are reintroduced.

Step 2: Determine two secondary MDS sets \( Q^k(i^*) \) and \( Q^k(j^*) \). Pairs of segment sets that are dissimilar from each other are sought. The average intra-similarity \( s^{\tilde{k}}(i, j) \) between each pair of segment sets \( Q^k(i) \) is calculated using Eq. (7). For all pairs with an intra-similarity lower than a threshold, \( \tau_1 \), \( s^{\tilde{k}}(i, j) < \tau_1 \), a mark is calculated quantifying their dissimilarity from MDS\((1)\) using Eq. (10) (for \( i > 3 \) the dissimilarity from all previously selected MDSs is quantified using an adapted equation (10)). Pairs for which the mark is lower than a threshold \( \tau_2 \) are tagged. From the tagged pairs, the pair for which the segment sets are temporally furthest distant from each other, based on average segment start position, is selected.

\[
Mark^k(i,j) = s^{\tilde{k}}(i, \text{MDS}(1)) + s^{\tilde{k}}(j, \text{MDS}(1)) < \tau_2
\]  

(10)

Step 3: The exemplar segments of the selected pair of segment sets \( Q^k(i^*) \) and \( Q^k(j^*) \) are calculated using Eq. (4).

Algorithm SMDS: find a pair of secondary MDSs for each form \( k \)

Create a reduced collection of segment sets \( Q^k(i) \)

Remove all segment sets that overlap MDS\((1)\) (when selecting MDS\((i), i > 3 \) the overlap is tested to all previously selected MDSs).

Verify there are at least two segment sets.

\( QD = [] \); \( n^k \) -- the number of remaining segments sets.

for \( i = 1, \ldots, n^k \) do

for \( j = i + 1, \ldots, n^k \) do

if \( s^{\tilde{k}}(i,j) < \tau_1 \) then

calculate mark according to Eq. (10)

if mark < \( \tau_2 \)

\( QD = [QD; \{i, j, i-j \text{ average segment start position, mark}\}] \)

end if

end if

end for

end for

From \( QD \) select \( ij \) for which \( i-j \) average segment start position is the largest

Calculate the exemplar segments of segments sets \( i \) and \( j \)

Set the two calculated segments as the secondary MDSs

If more than two secondary MDSLCSs are desired than the algorithm is re-run

3.4. The MDSLCS gesture classification algorithm

The MDSLCS algorithm recognizes MDSs using the modified LCS algorithm. The algorithm also uses an adaptive window size for identifying the MDSs of each gesture-form (see Section 3.4.1). The window size, overlap, and maximum allowed shift are a priori associated with each of the MDSs. The algorithm starts by looking for the primary MDS of each form. When such a segment is recognized the algorithm then looks for secondary MDSs associated with the same form. According to the specific form the secondary MDSs may be located before or after the primary MDS. If they are found the gesture is recognized, otherwise the algorithm re-initiates the search for a primary MDS by shifting the main window. A small example is given in Fig. 3 to illustrate the procedure. Before presenting the flow diagram of the MDSLCS algorithm, descriptions of the window management and LCS classifier are given. Note, that in the real-time classification stage samples are not resampled to a constant common length.

3.4.1. Window management

The main moving window \( W \), is shifted by increments of \( \Delta \) over the data stream. When looking for the MDS\((1)\), sub-windows \( W^k(1) \) are shifted within the main window \( W \) (Fig. 4a). This is repeated for

Fig. 3. Partition of digit “1” into three ordered MDSs. (a) MDS\((1)(1)\), (b) MDS\((1)(2)\), and (c) MDS\((1)(3)\). When all MDS\((i)\) are identified digit “1” is recognized. Notice MDS\((2)\) is located before MDS\((1)\) while MDS\((3)\) is located after it. The algorithm will look for them accordingly.
each form $k$. Since extracted segment test patterns are constrained by the window size, we set the window size according to 95% of the expected lengths of $MDS^k(1)$. If a MDS is recognized we set its form to $k_0$ and the corresponding MDS to $MDS^{k_0}(1)$. Here, within the same $W$, the position of segment $MDS^k(1)$ is fixed and the sub-windows $W^k(b)$, $b = 2, \ldots, B^k$ are shifted over the remaining space (maximum allowable shift position) of the data stream in some pre-defined areas of $W$ (Fig. 4b). Again, the secondary window sizes are set according to the lengths of $MDS^k(b)$.

### 3.4.2. LCS classifier

The patterns extracted from each of the windows described above are presented to the LCS algorithm. Let $X$ be a test pattern, extracted from the data stream to be classified, and $T^k$ the template (every form has an associated template). Let $X$ be aligned to each template $T^k$ for ($k = 1, \ldots, K$), in the template database $T$, using the LCS algorithm. Let $LCS(X,T^k)$ be the longest common subsequence between $X$ and $T^k$, of lengths $n$ and $m$, respectively. Then the fraction remaining ($FR^k$) of the test pattern $X$ after finding the LCS is calculated according to Eq. (11). This provides a good similarity measure between the test pattern and a template, as the larger the $FR$ the better the match.

$$FR^k = L(m,n)/n, \text{ where } L(m,n) \text{ is the length of the LCS}(X,T^k)$$

### 3.4.3. Overall description and flow diagram of the MDSLCS algorithm

Recognition of a MDS uses the LCS algorithm which compares $X$, extracted from the data stream, to each MDS stored as a template. The templates are equal to the primary and secondary MDSs learned using algorithms PDMS and SDMS respectively. These are denoted as $T^k(1)$ and $T^k(b)$ in Fig. 5.

First a main window, $W$, is scanned by $K$ sub-windows $W^k(1)$ (Fig. 4a) in an attempt to recognize any one of the primary MDSs($1), k = 1, \ldots, K$. The candidate segments in each sub-window $W^k(1)$ are compared to the template $T^k(1)$ using the LCS classifier to obtain the fraction reduction, $FR^k$ (Eq. (11)). If max $FR^k$ occurs for $k^*$ and is above a threshold, $MDS^{k^*}(1)$ is deemed recognized, and $k^*$ is used in the search for secondary MDSs. Sub-windows $W^k(b)$, $b = 2, \ldots, B^k$ are scanned over designated portions of $W$ sequentially (Fig. 4b). Let $X^k(b)$ and $T^k(b)$ represent a segment test pattern extracted from the window $W^k(b)$, and template for segment of form $k^*$, respectively. $FR^k(b)$ is the fraction reduction obtained from the LCS classifier. If $FR^k(b)$ is above a threshold then $MDS^{k^*}(b)$ is recognized. If all secondary patterns $MDS^{k^*}(b)$ ($b = 2, \ldots, B^k$) are recognized, then the full gesture $g$, as determined by the segment-form identifier $h(k^*)$, is considered as recognized. If at any stage an MDS pattern is not recognized, the main window $W$ is shifted by $\Delta$ and the system reverts to looking for a primary MDS.

### 3.4.4. Computational complexity of the MDSLCS algorithm

Determination of the computational complexity of the MDSLCS algorithm is based on the flowchart of Fig. 5. First we find the complexity of determining if a primary MDS is present in the input data stream. For each window in the data stream, a sub-window $X$ is extracted as a candidate MDS, and compared to each stored MDS template pattern. The comparison is performed by the LCS algorithm to determine $LCS(X,T^k)$, the longest common subsequence between $X$ and $T^k$, of lengths $n$ and $m$ and having complexity $O(nm)$. Repeating this operation $K$ times, one for each primary MDS in the database (recall there exists one primary MDS for each form), results in complexity $O(Kmn)$ after which a single operation according to Eq. (11) determines the $FR^k$. Finding max $FR^k$ takes time $O(log K)$. Hence, determination of a primary MDS in the input window takes time $O(log K) + O(Kmn)$, where $m$ is the sub-window size, $n$ is the longest MDS template $T^k(1)$ over all forms $k$, and $K$ is the number of gesture forms.

The complexity analysis of determining the secondary MDS follows the lower part of Fig. 5. New sub-windows $W^k(b)$, $b = 2, \ldots, B^k$ are scanned over designated portions of $W$. Let $X^k(b)$ and $T^k(b)$ represent a segment test pattern (extracted from the window $W^k(b)$), and the $b$th secondary MDS template of form $k^*$, respectively. Since the search for a secondary MDS is done for only one form $k$, and it can be any form $k$, the LCS algorithm is called a maximum of $Max \{B^k : k = 1, \ldots, K\} = B$ times to compare $X^k(b)$ and $T^k(b)$ and takes time the $O(Bmn)$, where $m$ is the sub-window size, $n$ is the longest MDS template and $B$ is maximum possible number of secondary MDS for any form $k$. Here, without lose of generality, we assume $n$ to be the maximum length of any primary and secondary MDS template. Thus, the total time complexity of the online MDSLCS classification algorithm is $O(log K) + O((K + B)mn)$. We note for our test digits $B$ was equal to 2, so that the time depends mostly on $m$, $n$, and $K$.

### 4. Testing the MDSLCS algorithm for digit gesture recognition

Palm’s graffiti digits (Fig. 6) were chosen for testing the MDSLCS algorithm. The digits have consistent starting points and can be drawn in free air as a single continuous path. Twelve subjects (4 male and 8 female, aged 23–35, right and left handed) were recorded using a PrimeSense 3D camera. The frame rate was set to thirty frames per second. Subjects were asked to draw the digits in the air while facing the camera. Speed and size were not constrained. Each subject executed 5 repetitions of each of 10 gesture digits. In total there were 61–64 samples of each digit (11 samples were discarded due to technical recording issues). A 5X2 cross validation experiment was performed for both pre-cut and streamed gesture recognition. In all experiments the MDSs were determined using 

pre-cut training samples (30 training samples per digit in each experiment).

For the automated clustering algorithm a distance threshold \( \tau = 0.75 \) was set as the limit below which a new sample was assigned to a new cluster. The number of forms for each gesture was restricted to a maximum of 2. Small “outlier clusters” (below the size of 5) were removed. Examples of different forms of the same gesture for digits ‘3’ and ‘8’ are shown in Fig. 7, where two examples of each digit (form A and form B) illustrate differences that caused the creation of different clusters.

Five samples from each form were used for finding the primary and secondary MDSs described above in Sections 3.3.3 and 3.3.4, respectively. Examples of MDSs for the two forms of digits ‘3’ and ‘8’ are shown in Table 1 (the true form shapes of the digits are not displayed but are stylized for ease of understanding). The variations between the forms are reflected in the variations in the MDSs.

For the pre-cut (pre-isolated) digit dataset an average accuracy of 92.6% was obtained using the MDSLCS algorithm (Table 2). The digits ‘5’ and ‘7’ had the lowest correct recognition rates, due to unclassified samples. Additionally, ‘5’ was misclassified as ‘3’. For the streamed digit dataset (non-cut gestures) an average recognition accuracy of 89.6% was obtained (Table 3). The digit pairs (‘2’, ‘3’) and (‘6’, ‘0’) were misclassified. The digits ‘1’ and ‘7’ had the highest number of unclassified samples. However, for both the cut and non-cut gestures only 5% of the samples were unclassified. At present the HMMs method is considered the standard,
state-of-the-art method for classifying motion trajectories. As such it is considered as the basis for an accuracy comparison of the MDSLCS algorithm. It has been reported in Frolova et al. (2012) that HMMs achieved an accuracy of 89.5% for the pre-cut, free hand digit gestures using a training set of approximately 50 samples per digit. For streamed, non-cut gestures its performance is expected to be poorer. The MDSLCS algorithm achieved a higher accuracy rate (92.6) for pre-cut gestures. The MDSLCS is more suitable than HMMs for streamed gestures, as it does not require a separate gesture spotting stage and handles streamed gesture inputs using adaptive windowing. In addition, the MDSLCS algorithm provides a significant advantage over the HMMs, as unlike HMMs, during real-time operation the MDSLCS algorithm does not require that all input data be converted to the same length. In addition, the MDSLCS algorithm requires a smaller training set. The Palm’s Graffiti Digits have fixed gesture trajectory starting points. To allow for other starting points, it is possible to identify them and treat them as separate gesture forms. To learn user’s typical starting points for a given gesture one may use the multiform clustering algorithm described in Section 3.2.

5. Conclusions

Automatic extraction and recognition of small representative segments (MDSs), allows for a more discriminative classifier than extracting and recognizing the full gestures themselves. We have developed a MDSLCS algorithm that recognizes MDSs using a modified LCS algorithm. A new automated procedure that learns these MDSs is proposed. The procedure extracts candidate segments of varying length and position from each gesture trajectory based on a training set of sample gestures. The LCS measure is used instead of the usual Euclidean distance, as our samples are of different lengths and therefore reside in different dimensional spaces. What makes the MDSLCS approach attractive is that it is easier to detect a small sequence rather than a large one (especially since a large sequence may contain much noise), and as the small segments are more discriminative between them than are full gesture trajectory sequences. The Palm’s graffiti digits (0–9) were chosen for testing the MDSLCS algorithm. The digits have consistent starting points and can be drawn in free air as a single continuous path. A recognition accuracy of 92.6% was obtained for pre-cut gestures and 89.6 for streamed gestures. At present the HMM is considered the standard state of art method for classifying motion trajectories. The HMM, when tested with pre-cut gestures, obtained an accuracy of 89.5%, and its performance is expected to be much poorer for the streamed gestures. It can then be said that the MDSLCS algorithm can outperform the HMM classifier for both pre-cut and streaming gestures. The MDSLCS is suitable for streamed gestures due to its adaptive windowing procedure. Additionally, the MDSLCS algorithm requires a small training set, making it more suitable for commercial gesture systems where users can select their own gestures. We also provide the computational complexity of the MDSLCS algorithm as $O(\log K) + O((K+B)m)$, where $m$ is the

<table>
<thead>
<tr>
<th>Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDSs for digits ‘3’ and ‘8’.</td>
</tr>
<tr>
<td>3 form A (Fig. 7a)</td>
</tr>
<tr>
<td>2211187777666666</td>
</tr>
<tr>
<td>11111888887</td>
</tr>
<tr>
<td>6665555554</td>
</tr>
<tr>
<td>3 form B (Fig. 7b)</td>
</tr>
<tr>
<td>8877777766555</td>
</tr>
<tr>
<td>118888888</td>
</tr>
<tr>
<td>5555555554</td>
</tr>
<tr>
<td>8 form A (Fig. 7c)</td>
</tr>
<tr>
<td>67788881111888</td>
</tr>
<tr>
<td>666554444</td>
</tr>
<tr>
<td>222222333</td>
</tr>
<tr>
<td>8 form B (Fig. 7d)</td>
</tr>
<tr>
<td>88111111888777</td>
</tr>
<tr>
<td>555666677</td>
</tr>
<tr>
<td>444333333</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confusion matrix for pre-isolated digit dataset [%]. Rows correspond to the ground truth labels; columns correspond to the estimated class labels. The ‘N/D’ (‘not a digit’) column shows non-recognized gestures. Recognition accuracy 92.6%.</td>
</tr>
<tr>
<td>0 &amp; 1 &amp; 2 &amp; 3 &amp; 4 &amp; 5 &amp; 6 &amp; 7 &amp; 8 &amp; 9 &amp; N/D</td>
</tr>
<tr>
<td>0 &amp; 99.3 &amp; 0.7 &amp;</td>
</tr>
<tr>
<td>1 &amp; 91.5 &amp;</td>
</tr>
<tr>
<td>2 &amp; 89.1 &amp; 2.8 &amp;</td>
</tr>
<tr>
<td>3 &amp; 0.6 &amp; 93.3 &amp; 0.5 &amp; 4.6 &amp;</td>
</tr>
<tr>
<td>4 &amp; 100 &amp;</td>
</tr>
<tr>
<td>5 &amp; 4.8 &amp; 85.3 &amp; 9.9 &amp;</td>
</tr>
<tr>
<td>6 &amp; 5 &amp; 90.2 &amp; 4.8 &amp;</td>
</tr>
<tr>
<td>7 &amp;</td>
</tr>
<tr>
<td>8 &amp;</td>
</tr>
<tr>
<td>9 &amp; 8.9 &amp;</td>
</tr>
<tr>
<td>1988</td>
</tr>
</tbody>
</table>
streaming video subwindow size, \( n \) is the longest MDS template, \( K \) is the number of gesture forms, and \( B \) is the maximum possible number of secondary MDS for any gesture form.

Acknowledgments

This work was supported in part by Deutsche Telekom AG. The authors would like to thank Tom Godo and Darya Frolova for their assistance during the development process.

References