Neural Comput & Applic (2013) 23:999–1010
DOI 10.1007/s00521-012-1024-0

ORIGINAL ARTICLE

Arabic sign language continuous sentences recognition using PCNN and graph matching

M. F. Tolba · Ahmed Samir · Magdy Aboul-Ela

Received: 21 February 2012 / Accepted: 21 June 2012 / Published online: 23 August 2012

Abstract Many previous researchers have tried developing sign languages recognition systems in general and Arabic sign language specifically. They succeeded to achieve acceptable results for isolated gestures level, but none of them investigated the recognition of connected sequence of gestures. This paper focuses on how to recognize real-time connected sequence of gestures using graph-matching technique, also how the continuous input gestures are segmented and classified. Graphs are a general and powerful data structure useful for the representation of various objects and concepts. This work is a component of a real-time Arabic Sign Language Recognition system that applied pulse-coupled neural network for static posture recognition in its first phase. This work can be adapted and applied to different sign languages and other recognition problems.

Keywords Sign language · Gesture · Posture · Arabic Sign Language Recognition (ASLR) · Pulse-coupled neural network (PCNN) · Graph matching · Graph isomorphism and sub-graph

1 Introduction

Sign language as a kind of gestures is one of the most natural means of exchanging information for most deaf people. The aim of sign language recognition is to provide an efficient and accurate mechanism to transcribe sign language into a text or speech. Sign language is a visual and manual language made up of signs created with the hands, facial expressions, and body posture and movement. Sign language conveys ideas, information, and emotion with as much range, complexity, and versatility as spoken languages [1]. Signs can be static or dynamic:

A. Static: The static gestures are called “hand postures”. Posture is a specific combination of hand position, orientation, and flexion observed at some time instance. Posture or static gestures are not time-varying signals [2], so they can be completely analyzed using only one or a set of images of the hand taken in a specific time.

B. Dynamic: Dynamic gesture is a sequence of postures connected by motions over a short time span. A gesture can be thought as a sequence of postures [2]. In the video signals, the individual frames define the postures and the video sequence defines the gesture. Arabic Sign Language (ASL) has more than 9,000 gestures and uses 26 static hand postures and 5 dynamic gestures to represent the Arabic alphabet [3]. Attempts at machine sign language recognition have begun to appear in the literature over the past decade. However, these systems concentrated on isolated signs and small dataset. This paper focuses on how a real-time sequence of dynamic gestures (a whole sentence) can be represented and segmented into primitive gestures (words) to be translated. It presents a proposed model using the graph-matching technique and a customized algorithm for connected gestures classification, which is a part of Arabic Sign Language Recognition (ASLR) System. The following points illustrate the reasons for applying graphs:

- Graphs have some interesting invariance properties [4]. For example, if a graph, which is drawn on paper, is

M. F. Tolba · A. Samir (✉)
Scientific Computing Department, Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt
e-mail: ahmed.new80@hotmail.com

M. Aboul-Ela
Sadat Academy for Management Sciences, Cairo, Egypt

translated, rotated, or transformed into mirror image, it is still the same graph in the mathematical sense. These invariance properties as well as the fact that graphs are very well suited to model complex objects in terms of parts and their relations make them.
- Graph representation and graph matching have been successfully applied to a large number of problems in computer vision and pattern recognition [4]. Examples include character recognition, schematic diagram interpretation, shape analysis, image registration, and 3-D object recognition.
- Contrary to different techniques such as Artificial Neural Network, Hidden Marcov Model, etc, graphs models database can be extended offline without system re-training.

Section 2 illustrates some previous sign language recognition system and some state in the art technology used. Section 3 discusses the pre-ASL recognition system that classifies static postures. Section 4 discusses graph-matching problem and how it can be employed for connected dynamic signs representation. In Sect. 5, the traditional traversal algorithm is discussed; Sect. 6 shows how graphs can be constructed for representation of dynamic gestures. A proposed modification of tree traversal is explained in Sect. 7; experimental results are illustrated in Sect. 8.

2 Related work

Previous work has been done in sign language recognition, Arabic and other languages. Bauer and Hienz [5] in 2000 developed a GSL (German Sign Language) recognition system that uses colored cloth gloves in both hands. The system is based on hidden Markov models (HMM) with one model of each sign. A lexicon of 52 signs was collected from one signer both for training and classification. A 94 % recognition percentage was achieved. Tanibata et al. [6]—in 2001—proposed a method of extraction of hand features and recognition of JSL (Japanese Sign Language) words. For tracking the face and hand, they could recognize 64 out of 65 words successfully by 98.4 %. Chen et al. [7] introduced in 2003 a system for recognizing dynamic gestures (word signs) for TSL (Taiwanese Sign Language). They used frequency domain features (Fourier Transform) plus some information from motion analysis for recognizing 20 words. The data set was collected from 20 signers, but the system is person dependent. HMMs were used as the classifier. An average of 92.5 % recognition rate was achieved. In 2004 and 2005, Zieren et al. [8, 9] presented two systems for isolated recognition: the first is for recognizing GSL, on a vocabulary of 152 signs achieving a rate of 97 %, using HMM, but the rate decreases for the group of signs that contain overlaps in either hands or face and hands. A larger ongoing project called SignSpeak is EU-funded and was introduced by Dreuw et al. [10]. It is being built on previous work, using an ordinary 2-D camera and aiming at translating continuous sign language to text, supporting a large vocabulary and recognizing both manual and non-manual features. Compared to other sign languages, not much has been done in the automation of the Arabic sign language, except few individual attempts. Al-Rousan et al. [11] and Aljarrah et al. [12] in 2000 and 2001, respectively, developed two systems for recognizing 30 static gestures of Arabic sign language, using a collection of Adaptive Neuro-Fuzzy Inference System (ANFIS) networks for training and classification depending on spatial domain features. In 2003, Assaleh et al. [13] used colored gloves for collecting varying size data samples for 30 manual alphabet of Arabic sign language. Polynomial classifiers were used as a new approach for classification. In a recent (2005) work in Arabic Sign Language, Mohandes et al. [14] developed a system that recognized 50 signs of words performed by one person having 10 samples per sign. They achieved a 92 % recognition accuracy. In 2010, Tolba et al. [15] used PCNN for recognition of 30 alphabets postures and gained a 93 % recognition accuracy. The system is signer independent and achieved system invariance against rotation, scaling, and color. When Microsoft released “Kinect”, Fig. 1, in November 2010 [16], it was mainly targeted at consumers owning a Microsoft Xbox 360 console, allowing the user to interact with the system using gestures and speech [16, 17]. The device itself features an RGB camera, a depth sensor, and a multi-array microphone and is capable of tracking users’ body movement [18].

This new technology encourages researchers in sign language recognition to customize it in real-time recognition. Natheer [19] at Jordan University of Science and Technology developed a Kinect demo for Arabic Sign Language Recognition that is capable of tracking some user’s fingers. However, the application used very limited datasets. Unfortunately, due to low resolution of Kinect cameras, hands details and shapes in fast time animation in many cases are indistinguishable.

Fig. 1 Using kinect in sign language recognition

3 Static postures recognition

This paper focuses on a single component in a whole ASL recognition system; it is assumed the existence of prior static posture recognition system, Fig. 2. First, the input sequence of gestures will be divided into static postures (frames). These frames are 2-D images containing both meaningful postures and movement transitions frames. A modern image understanding technique (PCNN) was used in converting the 2-D image to 1-D time series that is called “Image signature”. This signature uniquely identifies the image and can be considered as image features. Meanwhile, the classification component applies Multi-Layer Perceptron (MLP) neural network to classify the features to a posture class.

Since the available datasets are labeled, the supervised learning technique can be used in classification. This network has many benefits: it is a general purpose [20] model with various applications; it is capable of modeling non-linear complex functions, good performance in noisy input, and it can be adaptable for environmental changes by changing weights and/or topology.

3.1 PCNN

A pulse-coupled neural network (PCNN) is a model of a biological network, specifically, a model of fragment of cat’s sight network. It is a single-layer network [21, 22] composed of neurons. Each of them is linked to one pixel of the input image. Each neuron contains two input compartments: the feeding and the linking. The feeding receives an external stimulus as well as a local stimulus while the linking only receives a local stimulus [21, 23]. The local stimulus comes from the neurons within feeding radius. This local stimulus is hereafter called the firing information. The external stimulus is the intensity from the corresponding pixel in the picture. The feeding and linking are combined in a second order fashion to create the potential that then decides together with the output whether the neuron should fire or not [24]. There are several differences between the algorithms for the modified PCNN neuron and the exact physiological pulse-coupled neuron. The differences are due to several simplifications made to the calculations, while still keeping the main features of the general theory. Each neuron in the modified PCNN could be described by the following set of equations [22]:

$$L(i) = L(i-1)\cdot e(-\alpha L) + VL\cdot(R * \mathrm{Ysur}(i-1)) \tag{1}$$

$$F(i) = S + F(i-1)\cdot e(-\alpha F) + VF\cdot(R * \mathrm{Ysur}(i-1)) \tag{2}$$

$$U(i) = F(i)\cdot[1+\beta\cdot L(i)] \tag{3}$$

$$\theta(i) = \theta(i-1)e - (\alpha q) + V\theta\mathrm{Yout}(i-1) \tag{4}$$

$$\begin{aligned} &U > \theta(i) \Rightarrow Yout = 1(\text{Firing condition}) \\ &\quad \text{otherwise} \Rightarrow \mathrm{Yout} = 0 \end{aligned} \tag{5}$$

where $L(i)$ is input linking potential, $F(i)$ is input feeding potential, and S represents the intensity of a given image element. $U(i)$ is the activation potential of neuron, $\theta(i)$ is threshold potential of neuron, and (i) is iteration step. Parameters αL, αF, and αq decay coefficients, β is linking coefficient, and parameters VL and VF are coefficients of the linking and threshold potential. Ysur is the firing information that indicates whether the surrounding neurons have fired or not and Yout indicates whether this neuron fires or not. R is the matrix of weight coefficients and $*$ is convolution operator. An example of the modified PCNN neuron architecture is shown in Fig. 3.

PCNN model with some characteristics, such as strong adaptive capturing ignition, has been widely applied to image de-noising, image smoothing, image processing, image segmentation, and image fusion. It is also partly used in shortest path optimization, structural layout optimization, etc.

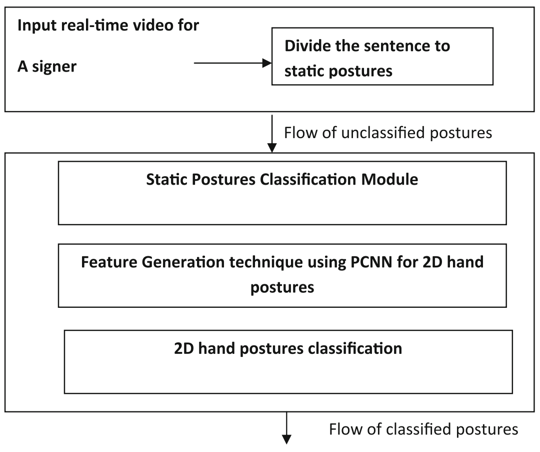

Input real-time video for
A signer
Divide the sentence to static postures
Flow of unclassified postures
Static Postures Classification Module
Feature Generation technique using PCNN for 2D hand postures
2D hand postures classification
Flow of classified postures

Fig. 2 ASL static postures recognition system

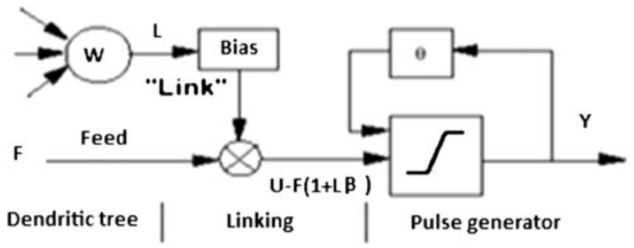


Fig. 3 Model for modified PCNN neuron

3.2 PCNN feature generation

Many feature generation methods have been developed using pulse-coupled neural network (PCNN). The comparative study on them is deeply investigated by Tolba et al. [15], which is out of this paper scope. Meanwhile, the equation proposed by Tolba [15] has been used to generate image signature:

$$g(n) = \frac{\sum_{i=1}^{n} (X(i) \times Y(i) \times CF(i))}{\sum_{ij} S_{ij}} \qquad (6)$$

where $Y(i)$ is output quantity based on step function, $CF(i)$ is the continuity factor, and $X(i)$ is output quantity based on sigmoid function. Figure 4 shows the image signature using (7) of the same hand posture. One image has a uniform light distribution, and another one is scaled and is distorted.

Why PCNN for feature generation

- It has the ability to extract image features without preprocessing steps, such as filtering or segmentation.
- It generates 1-D image signature that can identify the image. This signature is invariant to standard image transformations, such as rotation and scaling.
- The generated signature is easy to compute and adds no performance cost to the classification process.
- The image signature is invariant to background even in an uncontrolled environment.

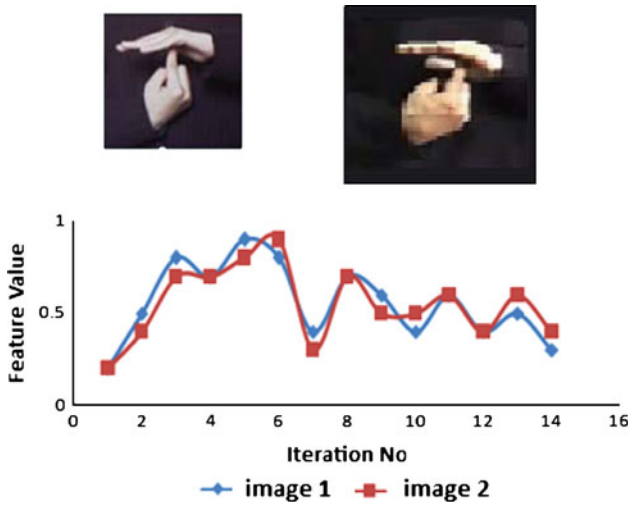


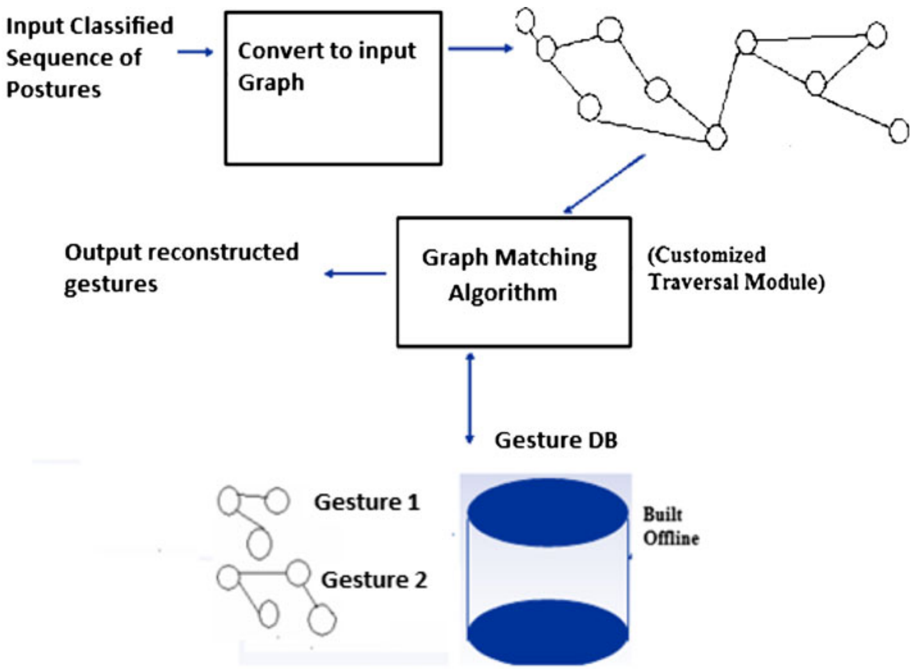
Fig. 4 Images and signatures for one has a uniform light distribution and another is a scaled distorted one

4 Proposed subgraph isomorphism by means of decision trees

The main idea of this paper is to customize the graph-matching problem and algorithms as a proposed solution for connected gestures classification. The gestures that represent alphabets or words are stored in database as models graphs; each graph consists of a group of vertices and edges, and these graphs are attributed directed graphs. The connected flow of input gestures is represented by the input graph. Figure 5 illustrates the main diagram for connected gestures recognition.

Let: $G = (V, E, \mu, \nu, L_v, L_e)$ be a model graph and M its corresponding $n \times n$—adjacency matrix. Furthermore, let $\boldsymbol{A}(\boldsymbol{G})$ denotes the set of all permuted adjacency, matrices of

Fig. 5 The main steps to recognize the connected gestures


Input Classified Sequence of Postures
Convert to input Graph
Graph Matching Algorithm
(Customized Traversal Module)
Output reconstructed gestures
Gesture DB
Gesture 1
Gesture 2
Built Offline

G, $A(G) = \{M_P | M_P = PMP^T$ where P is a $n \times n$ permutation matrix$\}$. The total number of permuted adjacency matrices is $|A(G)| = n!$ [25] as there are $n!$ different permutation matrices of dimension n. For a model graph G with corresponding $n \times n$ adjacency matrix $\boldsymbol{M}$ and an input graph $\boldsymbol{G_1}$ with an $m \times m$ adjacency matrix (M1) and $m \leq n$, determine whether there exists a matrix $M_p \in A(G)$ such that $M_1 = S_{m,m}(M_p)$. If such a matrix M_p exists, the permutation matrix $\boldsymbol{P}$ corresponding to M_p describes a sub-graph isomorphism from to $\boldsymbol{G}$, that is $M_1 = S_{m,m}(M_p) = S_{m,m}(PMP^T)$. If $\boldsymbol{G_1}$ and $\boldsymbol{G}$ are of equal size, the permutation matrix $\boldsymbol{P}$ represents a graph isomorphism between G1 and $\boldsymbol{G}$, that is, $M_1 = PMP^T$. One proposes to organize the set $\boldsymbol{A}(\boldsymbol{G})$ in a decision tree that each matrix in $\boldsymbol{A}(\boldsymbol{G})$ is classified by the tree. The features that will be used for the classification process are the individual elements in the adjacency matrices. One introduces a new notation for an $n \times n$ adjacency matrix $M = (m_{ij})$. One says that the matrix consists of an array of so-called ***row-column*** elements ***ai***, where each ***ai*** is a vector of the form [26]:

$$a_i = (m_{1i}, m_{2i}, \ldots m_{ii}, m_{i(i-1)}, \ldots m_{i1})$$

The matrix can then be written as: $M = (a_1, a_2, \ldots a_n)$; $i = 1, 2, \ldots, n$. Figure 6 illustrates the structure of an adjacency matrix $\boldsymbol{M}$ with regard to its row-column elements.

In Fig. 7, [27] a graph, g1, and its corresponding decision tree are shown. The nodes of the decision tree are represented by shaded circles. Each directed branch from one node to another has associated with it a row-column element. At the top of the figure, the set A(g1) of permuted adjacency matrices of g1 is listed.

If there are several model graphs in a database then the most trivial solution would be to build a decision tree individually for each model graph. However, it is possible

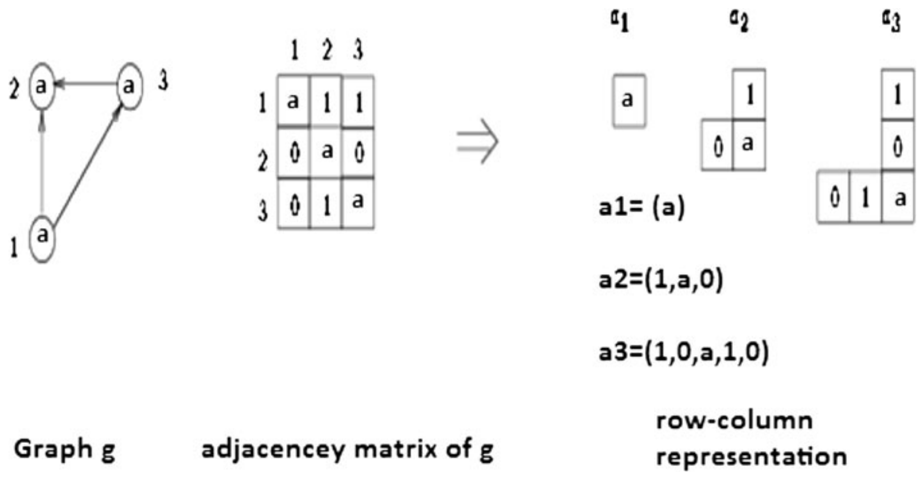


Fig. 6 The row-column representation of the adjacency matrix

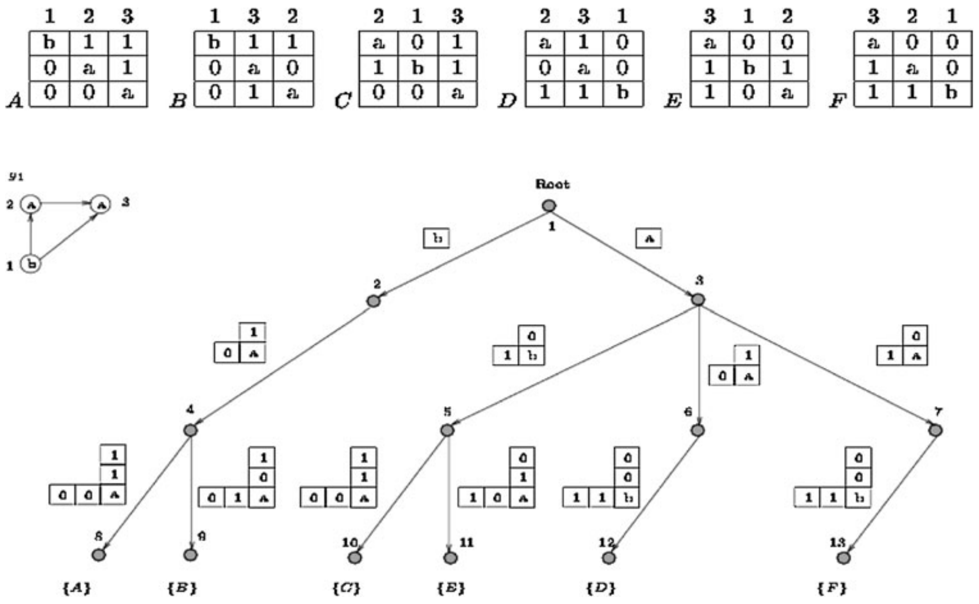


Fig. 7 The decision tree

to represent several graphs by the same decision tree. In Fig. 8, the decision tree for the graph g1 of Fig. 2 and another graph, g2, is displayed. In order to classify each of the adjacency matrices in A(g2), only two nodes have to be added to the decision tree that corresponds to the graph g1. As there are 3 automorphisms of g2, each of the nodes 13 and 15 in Fig. 8 represents 3 adjacency matrices.

5 Decision tree traversal

The decision tree structure that was previously described can now be used to get a very efficient graph and sub-graph isomorphism algorithm. Messmer and Bunke [28] described a decision tree approach to graph and sub-graph isomorphism detection. Figure 9 illustrates the (decision_tree) procedure [28]; this procedure tries to find out whether there exists a matrix $M \in A(G_i)$ such that $M_1 = S_{m,m}(M)$ by classifying $\boldsymbol{M_1}$ according to its row-column elements. Let $G_1, G_2 \ldots G_l$ be a set of model graphs represented by a decision tree and an unknown input graph. One assumes that the input graph is represented by its adjacency matrix $M_1 = (a_1, a_2, \ldots a_m)$ given in row-column format.

The algorithm starts at the root node of the decision tree and first classifies according to its first element a_1. If this step is successful, the classification is continued on the next level. In general, if the process is on level K and N is the current node of the decision tree, then the successor node of N that represents perfectly the process follows the branch from N to the successor node that represents the row-column element $a_{k,n}$. If no such element can be found in the dictionary then cannot be classified by the decision tree and it follows that G_i is not isomorphic to any sub-graph of the model graphs G_I. In step (2-d) of the algorithm, it is checked whether the current node has an outgoing redirecting branch. If this is the case then one follows this redirecting branch. Accordingly, the matrix of the input graph must be permuted by applying the permutation matrix R that is attached to the redirecting branch. Note, however, that if N is on level K then R is $(k \times k)$ permutation matrix because it was created at compilation time for a sub-graph of size k. An $(m \times m)$ adjacency matrix of the input graph with $m \geq k$. In order to apply R to M_i, it is necessary to extend R to an $(m \times m)$ matrix (R') by copying rows and column from $(m \times m)$ identity matrix.

Termination conditions:

1. The algorithm terminates in step (2-b) when it is detected for the first time that there is no sub-graph isomorphism from the input graph to any of the model graphs
2. In step (3) when the last row-column element a_m of M_i has been processed and some node N has been reached.

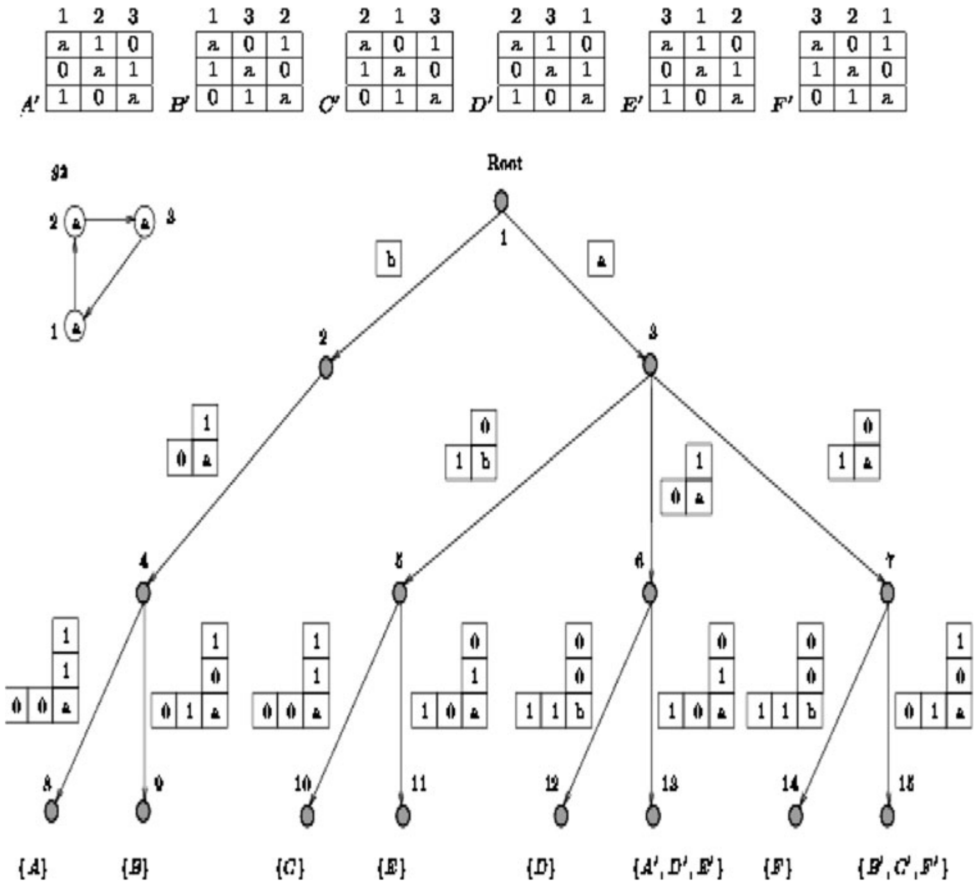
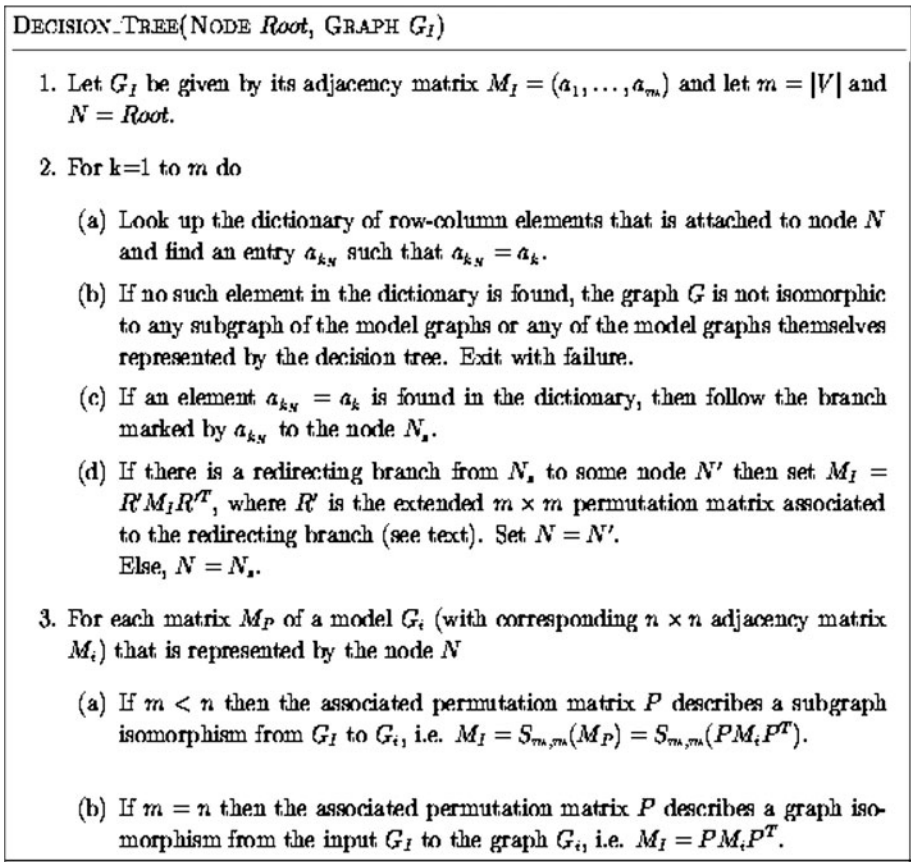


Fig. 8 Decision tree for multiple graphs



```
DECISION_TREE(NODE Root, GRAPH $G_I$)

1. Let $G_I$ be given by its adjacency matrix $M_I = (a_1, \dots, a_m)$ and let $m = |V|$ and
   $N = Root$.
2. For k=1 to $m$ do
   (a) Look up the dictionary of row-column elements that is attached to node $N$
       and find an entry $a_{k_N}$ such that $a_{k_N} = a_k$.
   (b) If no such element in the dictionary is found, the graph $G$ is not isomorphic
       to any subgraph of the model graphs or any of the model graphs themselves
       represented by the decision tree. Exit with failure.
   (c) If an element $a_{k_N} = a_k$ is found in the dictionary, then follow the branch
       marked by $a_{k_N}$ to the node $N_s$.
   (d) If there is a redirecting branch from $N_s$ to some node $N'$ then set $M_I =$
       $R' M_I R'^T$, where $R'$ is the extended $m \times m$ permutation matrix associated
       to the redirecting branch (see text). Set $N = N'$.
       Else, $N = N_s$.
3. For each matrix $M_P$ of a model $G_i$ (with corresponding $n \times n$ adjacency matrix
   $M_i$) that is represented by the node $N$
   (a) If $m < n$ then the associated permutation matrix $P$ describes a subgraph
       isomorphism from $G_I$ to $G_i$, i.e. $M_I = S_{m,m}(M_P) = S_{m,m}(P M_i P^T)$.
   (b) If $m = n$ then the associated permutation matrix $P$ describes a graph iso-
       morphism from the input $G_I$ to the graph $G_i$, i.e. $M_I = P M_i P^T$.
```

Fig. 9 Decision tree traversal procedure [28]

In this case, the matrix is identical to all matrices M_i of the model G_i that are represented in N.

If N is not a leaf node then the set of permutation matrices that are stored in N represents all sub-graph isomorphism from the graph G_I to G_i. If, on the other hand, N is a leaf node and G_I, G_i are of equal size then the set of permutation matrices in N represents all graph isomorphism between G_I and G_i.

5.1 Important notes

- It is easy to see that the new algorithm for graph isomorphism traverses the decision tree without the need for backtracking and therefore has a time complexity that is only polynomial in the number of vertices of the input graph.
- The algorithm is clearly independent of the number of model graphs that are represented in the decision tree.

One now faces two main challenges:

- Finding a technique that constructs the graphs and discards the transitional movements inherent to the finger spelling or dynamic gesture.
- Secondly, finding a customization methodology to the Tree_Traversal procedure to be applicable to the dynamic gesture re-construction problem.

6 Graph construction

The gestures that represent alphabets or words are stored in database as models graphs. Each graph consists of a group of vertices and edges. These graphs are attributed and directed graphs. The connected flow of input gestures is represented by the input graph. The main challenge is how to discard the transitional movements inherent to the finger spelling or dynamic gesture.

First approach: This approach mainly ignores this problem. It gives each image (posture) a distinct vertex in the constructed graph. In this approach, unknown posture class appears, this class represents the undetermined postures.

Second approach: In this approach, one discards the transitional movements before posture classification. One defines the frame difference energy function between two consecutive frames as:

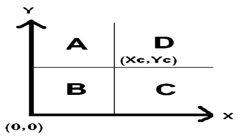


Fig. 10 The main 4 regions of the frame

$$E_d(f_i, f_{i-1}) = \sum_{l=1}^{n!} \sum_{c=1}^{nc} f_i(l, c) - f_{i-1}(l, c))^2 \tag{7}$$

$f_i(l, c)$ is the pixel value for the ith frame on the position (line, column). Given the condition $E_d(f_i, f_{i-1}) \geq E_{th1}$ where $Eth1$ is a fixed threshold value. If the energy of frame difference violates the condition, then it is selected as a transitional frame, so it is discarded.

Third approach: this approach mainly depends on filtering the hand postures after hand classification module. The filtering criterion is based on:

1. *The output class of the posture*. This feature represents the output class from the previously applied hand classification module. Besides, the unknown class exists.
2. *The relative position of the hand*. This feature represents the relative hand position. The frame is presegmented to four regions as illustrated in Fig. 10. These regions are labeled as {A, B, C, and D}. Any static posture lies into one of these regions.

The graph consists of a set of vertices. The input posture sequence is filtered according to the previously mentioned criteria. The given posture is considered a new vertex in the following cases (Fig. 11):

1. The new posture (Pnew) and the previous posture (Pold) have a different posture classes.
2. The new posture (Pnew) and the previous posture (Pold) are in different regions.

The three methods have been applied to the dataset; the third technique shows better time results. The results and comparison of the three methods are shown in detail in the experimental results section (Fig. 12).

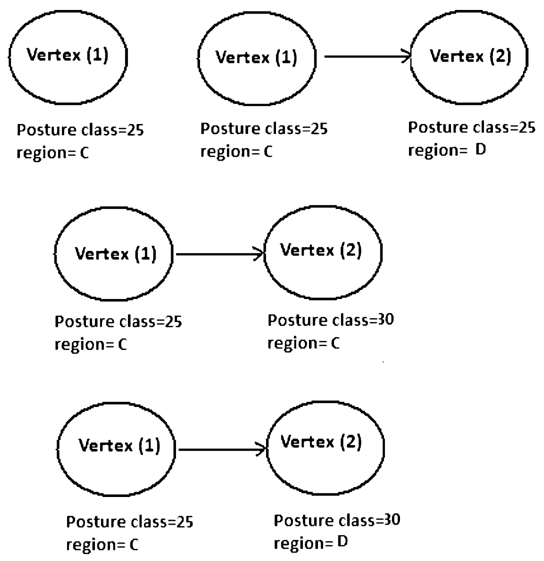


Fig. 12 The corresponding graphs in order

7 Proposed modification of traversal algorithm

The algorithm proposed by Messmer and Bunke [28] has been customized to suit the given application, Fig. 13. To accomplish this, one has two points to cover:

1. The algorithm assumes that the input graph is smaller than or equal to any model graph in the database. In other words, the procedure looks for sub-graph isomorphism from input graph to models graphs. While in the current application, the input graph represents an input sentence and the model graphs are isolated gestures. In other words, one needs to discover the contrary, sub-graph isomorphism from models graphs to input graph.
2. The algorithm terminates after finding one sub-graph isomorphism. Meanwhile, the application needs to discover all sub-graphs isomorphism with all models graphs.

The customization idea depends on changing the termination condition of the algorithm, in such a way that, it stops when the input sentence is recognized. The modification should cover three different cases:

- The input sentence is equivalent to one sentence (graph isomorphism).
- The input sentence corresponds to a sequence of gestures. (multiple sub-graphs isomorphism).
- No gestures were recognized.

In step (2-b) while an inner loop is added, this loop finds out whether there exists one sub-graph isomorphism. In step

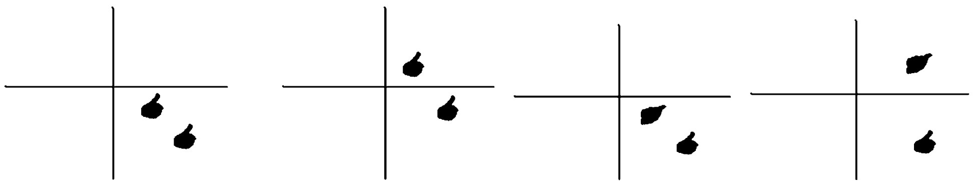
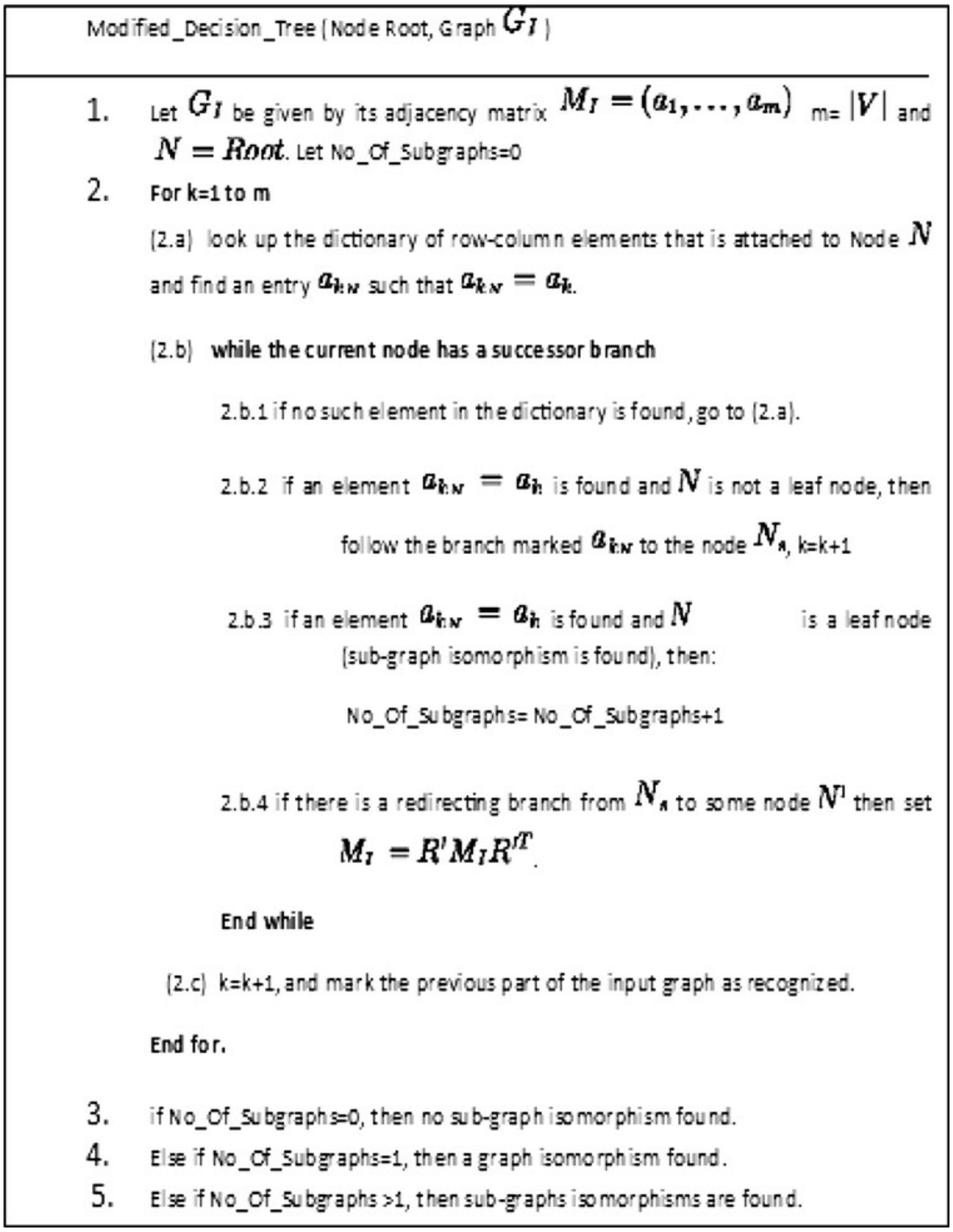


Fig. 11 Four examples for two hand postures with different two hand posture classes and different regions



```
Modified_Decision_Tree (Node Root, Graph $G_I$)

1. Let $G_I$ be given by its adjacency matrix $M_I = (a_1, \dots, a_m)$ m= $|V|$ and
   $N = Root$. Let No_Of_Subgraphs=0
2. For k=1 to m
   (2.a) look up the dictionary of row-column elements that is attached to Node $N$
   and find an entry $a_{kN}$ such that $a_{kN} = a_k$.
   (2.b) while the current node has a successor branch
       2.b.1 if no such element in the dictionary is found, go to (2.a).
       2.b.2 if an element $a_{kN} = a_k$ is found and $N$ is not a leaf node, then
             follow the branch marked $a_{kN}$ to the node $N_s$, k=k+1
       2.b.3 if an element $a_{kN} = a_k$ is found and $N$ is a leaf node
             (sub-graph isomorphism is found), then:
             No_Of_Subgraphs= No_Of_Subgraphs+1
       2.b.4 if there is a redirecting branch from $N_s$ to some node $N'$ then set
             $M_I = R' M_I R'^T$.
       End while
   (2.c) k=k+1, and mark the previous part of the input graph as recognized.
   End for.
3. if No_Of_Subgraphs=0, then no sub-graph isomorphism found.
4. Else if No_Of_Subgraphs=1, then a graph isomorphism found.
5. Else if No_Of_Subgraphs >1, then sub-graphs isomorphisms are found.
```

Fig. 13 The Modified_decision_tree procedure

(2-b.3), if a leaf node is reached, then a sub-graph isomorphism exists, increasing the No_Of_Subgpaphs by 1. In step (2-c), go to the next input graph node and mark the previous input graph part as recognized. After the end of the external for loop (end of input graph), check for the value of No_Of_Subgpaphs. Such that, if No_Of_Subgpaphs still takes 0, then either graph or sub-graph isomorphism are found. Else if No_Of_Subgpaphs takes 1 then a graph isomorphism was found, it means that the input sentence is equivalent to one gesture. Else if No_Of_Subgpaphs takes a value more than 1, then sub-graphs isomorphism is found, it means that the input sentence corresponds to a sequence of gestures.

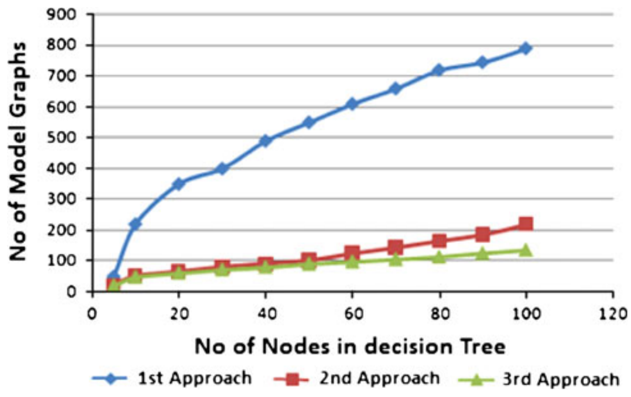


Fig. 14 The tree size against model graphs database size

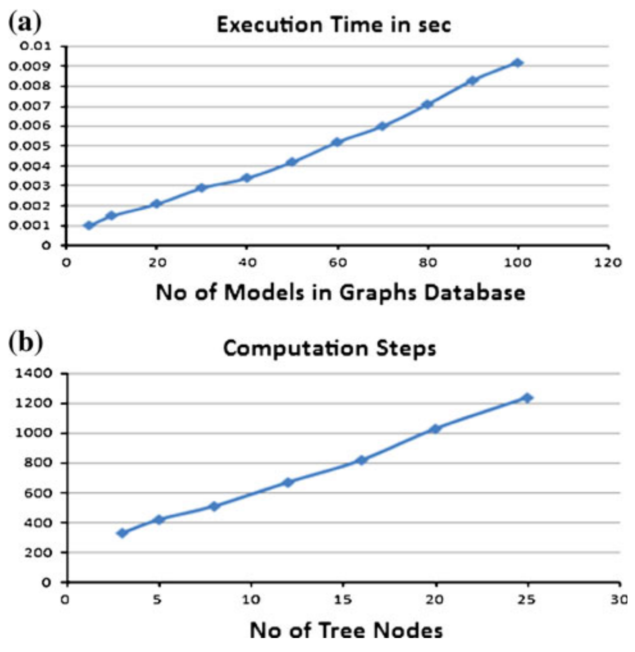


Fig. 15 **a** Execution time in sec against the database size. **b** Computational steps against number of tree nodes

8 Results

In this section, an experiment contains 30 connected sentences of total 100 words. First, a study is conducted to measure the effect of graph construction approaches on the decision tree size in terms of nodes count using the three approaches (graph construction section). Figure 14 illustrates the size of the offline built decision tree against the size of the graph models database size.

It is clear that applying the third approach leads to a minimum tree size relative to the other approaches. The 1st approach gives the maximum tree size because it ignores the transitional movements; it causes that each frame represents a distinct vertex in the gesture graph. The equation used in the second approach decreases the tree size because it omits transition frames. Meanwhile, it does not accomplish the minimum size because of its relative nature of the threshold values. The performance is then measured by: computing the execution time in seconds and counting the number of basic computation steps that are performed while searching for all graph and sub-graph isomorphism. A basic computation step is defined as the comparison of one model graph vertex and its incident edges to one input graph vertex and its incident edges. Figure 15 illustrates the system performance measures using execution time against model graphs database size. Moreover, the computational steps needed against the decision tree nodes count.

The performance measurements emphasizes the polynomial nature of the proposed customized sub-graph isomorphism algorithm; it concludes that as much as the gestures graphs database size increases, the running time will be still bounded by a polynomial. As mentioned before, 30 connected sentences are tested using a graph database containing 100 dynamic gestures (words). The recognition accuracy of the proposed system is measured using the number of 3 words-sentences that have been successfully translated and the number of 4 words-sentences that have been successfully recognized. Table 1 illustrates the recognition accuracy of both 3 and 4 words sentences.

Table 1 The recognition accuracy

Graph database size	Totally successfully translated	Sentences with single error		Sentences with double error		Sentences totally unclassified
		Misclassified	Unclassified	Misclassified	Unclassified	
Sentences composed of 3 words						
10	28	1	–	1	–	–
20	28	1	–	1	–	–
40	25	2	1	1	–	1
60	24	3	1	1	–	1
80	21	5	1	2	1	1
100	21	5	1	2	1	1
Sentences composed of 4 words						
10	26	2	–	2	–	–
20	25	2	1	2	–	–
40	22	2	2	2	1	1
60	20	3	2	2	2	1
80	20	3	2	2	2	1
100	19	3	2	2	2	2

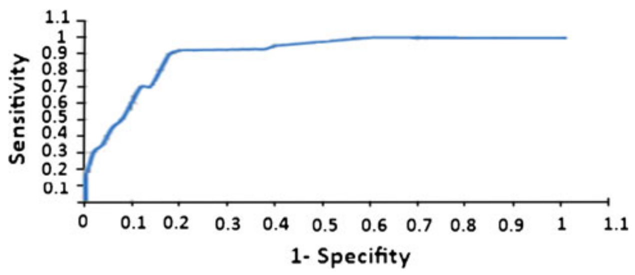


Fig. 16 ROC curve for the third proposed graph construction methods

Table 1 illustrates that the recognition accuracy of 30 sentences does not lay down 19 complete successfully translated sentences. Besides, it reveals the reality that as much as the database size increases, the recognition accuracy decreases. This decrease is a result of adding models graphs that are very similar, which causes misclassification of the input sentences. The comparison of the ROC (receiver operating characteristics) curves obtained by the third graph construction methods is given by Fig. 16.

Table 2 illustrates a comparative study between the proposed system and the previous researches in sign language recognition.

9 Conclusion

This paper proposes and implements ASLR system of it focuses on recognizing the continuous gestures using graph-matching technique. Gestures have been divided into elementary elements, static postures. Gesture recognition is performed by "graph-matching" algorithm. The gestures, which represent alphabets or words, are stored in database as models graphs. Each graph consists of a group of vertices and edges. The algorithm used for graph and sub-graph isomorphism detection is based on the decision tree paradigm. In the computational complexity analysis, it is shown that the new algorithm has a worst-case run time complexity that is only quadratic in the size of the graphs that are to be compared. The recognition rate does not lay down 70 % for 100 gestures composing

Table 2 A comparative study with existing systems

Author	Sign language	Dataset size	Recognition model	Recognition accuracy	Constraints
Bauer and Hienz [5]	German	52 signs From a single signer	Hidden Marcov models	94 %	Recognition accuracy is guaranteed for one signer The dataset is not extendible, when we need add a new sign, it needs re-training to the whole system
Chen et al. [7]	Taiwanese	20 words from 20 signer	Frequency domain features (Fourier transform) plus some information from motion analysis HMMs was used as the classifier	92.5 %	Recognition accuracy is guaranteed for one signer. Isolated signs only
Aljarrah et al. [12]	Arabic	30 static gestures	Collection of adaptive neuro-fuzzy inference system (ANFIS) networks for training and classification depending on spatial domain	96 %	Static signs only
Mohandes et al. [14]	Arabic	50 signs of words performed by one person having 10 samples per sign	Histograms analysis and MLP as classifier	They achieved 92 % recognition accuracy	Static postures only
Tolba et al. [15]	Arabic	30 alphabets postures	PCNN followed by MLP	Gained 93 % recognition accuracy	The system is signer-independent and achieved system invariance against rotation, scaling and color Found difficulties in identifying some postures. This is caused by the fact that a single view for the hand does not distinguish between two different postures
Proposed model	Arabic	158 static postures and 50 dynamic gestures	Hybrid PCNN and graph-matching approach	Gained 80 % for continuous sentences	The system is insensitive to signer position, view angle and background effects

30 continuous sentences. This work can be extended by applying natural languages techniques as a post-processing component for overcoming any possible classification errors.

References

1. Doner B (1993) Hand shape identification and tracking for sign language interpretation. Looking at People Workshop, Chambery
2. Bauer B, Hienz H (2000) Relevant features for video-based continuous sign language recognition. In: Proceedings of the fourth IEEE international conference on automatic face and gesture recognition, pp 64–75
3. Chao M-W, Lin C-H, Chang C–C, Lee T-Y (2011) A graph-based shape matching scheme for 3D articulated objects. Comput Animat Virtual Worlds 22(2–3):295–305
4. Gupta M (2011) Design pattern mining using greedy algorithm for multi-labelled graphs. Int J Inf Commun Technol 3(4):314–323
5. Bauer B, Hienz H (2000) Relevant features for video-based continuous sign language recognition. In: Proceedings of the fourth IEEE international conference on automatic face and gesture recognition, pp 64–75
6. Tanibata N, Shimada N, Shirai Y (2001) Extraction of hand features for recognition of sign language words. In: Proceedings of the 15th international conference on vision interface. Calgary, Canada
7. Chen F-S, Fu C-M, Huang C-L (2003) Hand gesture recognition using a real-time tracking method and hidden Markov models. Image Vis Comput 21:745–758
8. Zieren J, Kraiss K-F (2004) Non-intrusive sign language recognition for human-computer interaction. In: 9th IFAC/IFIP/IFORS/IEA symposium analysis, design, and evaluation of human-machine systems, Atlanta, GA, pp 221–228
9. Zieren J. Kraiss, K-F (2005) Robust person- independent visual sign language recognition, proceedings of pattern recognition and image analysis, second Iberian conference, Estoril, Portugal
10. Dreuw P, Forster J, Gweth Y, Stein D, Ney H, Martínez G, Vergés-Llahí J, Crasborn O, Ormel E, Du W, Hoyoux T, Piater J, Moya JM, Wheatley M (2010) SignSpeak understanding, recognition, and translation of sign languages. In: 4th Workshop on the representation and processing of sign languages: corpora and sign language technologies, language resources and evaluation conference (LREC)
11. Al-Rousan M, Aljarrah O, Hussain M (2001) Automatic recognition of arabic sign language finger spelling. Int J Comput Their Appl 8(2):80–88 (Issue 1076-5204)
12. Jarrah O, Halawani A (2001) Recognition of gestures in Arabic Sign Language using neuro-fuzzy systems. Artif Intell, pp 117–138
13. Assaleh K, Al-Rousan M (2005) Recognition of Arabic sign language alphabet using polynomial classifiers. EURASIP J Appl Signal Process Soc 13:2136–2145
14. Mohandes M, Deriche M (2005) Image based Arabic sign language recognition. Signal processing and its applications, 2005. In: Proceedings of the eighth international symposium, vol 1, pp 86–89
15. Tolba MF, Abdellwahab MS, Abulle-Ela M, Samir A (2010) Image signature improving by PCNN for Arabic sign language recognition. Can J Artif Intell Mach Learn Pattern Recogn 1(1):1–6
16. Official Microsoft Xbox website, introduction of Kinect, http://www.xbox.com/en-US/kinect
17. Countdown to Kinect: 17 Controller-Free Games Launch in November. Microsoft News Center, https://www.microsoft.com/presspass/press/2010/oct10/10-18mskinectuspr.mspx
18. Kinect Fact Sheet, Microsoft News Center, June 2010, http://www.microsoft.com/presspass/presskits/xbox/docs/KinectFS.docx
19. Integrating Speech and Hearing Challenge Individuals. YouTube channel of Dr. Natheer Khasawneh, http://www.youtube.com/user/knatheer#p/a/u/1/vVL398dUU5Q
20. Topouzelis K, Karathanassi V, Pavlakis P, Rokos (2003) A neural network approach to oil spill detection using SAR data. In: 54th International astronautical congress, Bremen, Germany
21. Ma YD, Li L, Wang YF (2006) The principles of pulse coupled neural networks and its application. Science Press, Beijing
22. Eckhorn R, Reitboeck HJ, Arndt M, Dicke P (1990) Feature linking via synchronization among distributed assemblies: simulations of results from cat visual cortex. Neural Comput 2(3):293–307
23. Forgá R (2008) Feature generation improving by optimized PCNN, applied machine intelligence and informatics. SAMI 2008. In: 6th International symposium, pp 203–207, Issue, 21–22, Jan. 2008
24. Kinser JM, Nguyen C (2000) Pulse image processing using centripetal autowaves. Proc SPIE 4052:278–284
25. Qiu H, Hancock ER (2003) Graph partition for matching. In: Proceedings of 4th international workshop on graph-based representations in pattern recognition. Springer LNCS2726, Berlin, pp 178—189
26. Delalandre M, Trupin E, Ogier JM (2004) Local structural analysis: a primer. In: Proceedings of workshop on graphics recognition. Springer, LNCS3088, Berlin, pp 220–231
27. Qiu H, Hancock ER (2003) Graph partition for matching. In: Proceedings of 4th international workshop on graph-based representations in pattern recognition. Springer LNCS2726, Berlin, pp 178–189
28. Messmer BT, Bunke H (1999) A decision tree approach to graph and subgraph isomorphism detection. Pattern Recogn 32(12):1979–1998