Arabic Dynamic Gesture Recognition Using Classifier Fusion

BASMA HISHAM
ALAA HAMOUDA

Sign language is a visual language that is the primary way used by hearing-impaired people in order to connect and communicate with each other and with their societies. Some studies have been conducted on Arabic sign language (ArSL) recognition systems, but a practically deployable system for real-time use is still a challenge. The main objective of this paper is to develop a novel model that is able to recognize the ArSL using Microsoft's Kinect V2. This paper works on the dynamic gestures that are performed by both hands and body parts, and introduces an effective way of capturing and detecting the hand and skeleton joints from the depth image that is provided by Kinect. The model used two supervised machine learning algorithms, support vector machine (SVM) and *K*-nearest neighbors (KNN), and then applied Dezert–Smarandache theory (DSmT) as a fusion technique in order to combine their results. We compared the results of the proposed model with the Ada-Boosting technique and finally applied two most widely used methods that are used with dynamic gesture recognition, dynamic time warping (DTW) and hidden Markov model (HMM), to compare their results with the previous classifier fusion. Finally, we applied the model on ArSL dataset that is composed of 40 Arabic medical signs to ease the communication between hearing-impaired patients and their doctor. The accuracy of the model is improved when the classifier fusion is applied compared to using each classifier separately. The overall accuracies for SVM, KNN, DSmT fusion, and Ada-Boosting are 79%, 89%, 91.5%, and 90.2%, respectively. Also, DTW and HMM achieved overall accuracies of 82.6% and 79.5%, respectively.

Manuscript received February 6, 2019; released for publication July 8, 2019.

The authors are with the Computer Engineering Department, Al-Azhar University, Cairo, Egypt.

(Corresponding author: Basma Hisham) basmahisham.2015@gmail.com

1557-6418/19/$17.00 © 2019 JAIF

I. INTRODUCTION

Sign language is the most basic way for hearing-impaired people to connect and interact with each other and integrate with their societies. The main problem is that most of the normal people do not understand sign language [1]. Therefore, the need to develop an automatic system that is capable of translating sign languages into understandable words and sentences is becoming very necessary. There are two main approaches to sign language recognition systems: vision-based approach and sensor-based approach. The main advantage of the vision-based system is that there is no need to use complex devices, so it has low cost and does not need pre-setup, but this approach requires extra calculations in the preprocessing stage, image processing, and artificial intelligence to recognize and interpret signs. Also, it suffers from the background problems because it needs subtraction techniques to subtract the signer from the background and it may fail if the background changes. Sensor-based systems provide robust, reliable, and more accurate data, but they are not user-friendly like vision-based systems because they require extra equipment like data gloves. The user is required to wear the gloves in order to collect the data, so this approach is not practical [2]. Microsoft Kinect is a motion sensing input device that is developed by Microsoft. It provides live streams of depth information about the skeleton joints and body motion. This information is essential to construct the three-dimensional (3-D) view of the tracked objects. It used to track standing skeleton with high-depth fidelity, so compared with other depth sensors, Kinect is the best choice in short-range environment [3]. Kinect also has an RGB camera, voice recognition capability, face-tracking capabilities, and access to the raw sensor records. Once the data have been collected from the user, the recognition system, whether it is sensor-based or image-based, must use these data for processing to recognize the signs [4]. Several approaches have been proposed for sign recognition, the most important and effective approach being that using machine learning algorithms. It can handle the complexity and the differentiation of sign language gestures [5]. Also, it can handle the different manners in which the people repeat different signs [6]. Machine learning algorithms such as neural networks, support vector machines (SVMs), *K*-nearest neighbors (KNN), decision tree models, etc. have been focused on the classification stage of recognizing a gesture captured from the signer. The single sign classifier assumes that signs are presegmented, and it recognizes sign by sign rather than continuous sentences. It is supposed to automate the process of splitting a sentence into words, which is called segmentation. Segmentation is one of the major issues of information processing in sign languages. Motion speed during capturing of continuous sentences may be used as a segmenter. It is noticeable that the motion speed is changed while performing the signs, and when the transition from one sign to another occurs, the motion speed is slowed down.

The main aim of this paper is to develop an Arabic sign language (ArSL) recognition system that identifies the Arabic signs captured by Microsoft Kinect based on the data that represent body and hand motion. These data will be excluded from the depth image information obtained from the Kinect sensor. However, Kinect cannot accurately detect the hand movement and also the details of fingers, but we overcame such limitations and introduced an effective and simple method for hand detection. We used two machine learning algorithms, KNN and SVM, and introduced an effective fusion method based on Dezert–Smarandache theory (DSmT) to combine the classifier results and enhance the accuracy. Also, we applied two direct matching algorithms, dynamic time wrapping (DTW) and hidden Markov model (HMM), to compare the results of fusion with other dynamic gesture recognition techniques. The structure of this paper is organized as follows. Section II presents the related work of the sign language recognition. The methodology is presented in Section III. The experimental setup and results are presented in Section IV. Section V contains conclusion and future work.

II. RELATED WORK

Sign language is a combination of words that are represented by using movements of different body parts such as head, shoulders, elbow, wrist, etc. and finally added to the hand signs to create a meaning [7]–[11]. Many researchers aimed to build an automated system in order to translate ArSL to Arabic text or voice. ArSL research works faced several difficulties; for example, it is not defined well and the works in it started in the last decade. However, in ArSL, there are more than 9000 signs and it uses 26 hand postures and 5 dynamic gestures in order to represent the Arabic alphabet. There is a variation in ArSL among the different Arabic countries. Some Arabic countries have their own sign language, such as Tunis, Gulf countries, Egypt, etc. We are concerned with ArSL in Egypt. The organization of ArSL in Egypt has started in 1983, and there are 7 million hearing-impaired persons till the last studies performed by United Nations. This is a large number, so they need to be merged with their societies as any normal person. As explained earlier, there are mainly two approaches for sign recognition: 1) vision-based approach and 2) sensor-based approach. For vision-based approach, there are several ArSL recognition works, such as an Arabic sign recognition model developed by Mohandes *et al.* for Arabic alphabet recognition, using SVM as a classifier with an accuracy of 87% [12]. Ahmed *et al.* also proposed a model for sign language recognition. Several statistical analyses were performed on the data extracted from the collected images to create the feature vector that is input to an SVM. The model was tested on ten letters and the accuracy was 83%. They suggested building a real-time system that is able to work on both dynamic and static signs [13]. Maraqa and Abu-Zaiter developed a static and dynamic ArSL recognition system by applying feedforward and recurrent neural networks on the features extracted from the captured images. They tested the proposed system on 30 gestures and achieved an accuracy of 95% [14]. Assaleh and Al-Rousan developed ArSL recognition for alphabet signs based on polynomial classifiers; they compared the results of the system with the previously published results using Artificial Neural Network Fuzzy Inference System (ANFIS)-based classification on the same dataset and feature extraction method. The comparison showed significant improvement, and the misclassified patterns were reduced by 36% on the training set and by 57% on the test set [15]. Al-Jarrah and Al-Omari developed an automatic system for Arabic alphabet recognition with an accuracy of 97.5% [16]. Also, El-Bendary *et al.* proposed a sign language recognition system for the Arabic alphabets, which dealt with the images of bare hands that allow the user to interact naturally with the system and achieved an accuracy of 91.3% [17]. For a sensor-based approach, Assaleh *et al.* proposed a low-complexity word-based classification for ArSL recognition system using two DG5-VHand data gloves, and the recognition rates were 92.5% and 95.3% for user-independent and user-dependent modes, respectively [18]. Mohandes proposed an ArSL recognition system using CyberGlove in order to track 100 two-handed signs with 20 samples and achieved an accuracy of 99.6% [19]. Sadek *et al.* proposed a hand gesture recognition system using a smart glove that was designed from a set of sensors; the recognition was based on a statistical analysis of the hand shape while performing the 1300 words of the ArSL [20]. Hemayed and Hassanien (2010) presented Arabic alphabets recognition system, the alphabets signs were converted into speech but the system cannot operate in real time, the model was built based on the vision-based, the system take the colored images as input also they used Prewitt edge detector in order to detect the hand shape. KNN (K-Nearest Neighbour) was used in the classification phase [21]. Recently, some researchers used active devices such as Microsoft Kinect and Leap Motion controller (LMC). Almasre and Al-Nuaim proposed hand gesture recognition systems using supervised machine learning that predicts the hand pose using two sensors, Microsoft Kinect and LMC, depending on the depth images. They collected data regarding 28 letters from different signers and the results achieved about 100% detection rate in recognizing 22 letters from 28 Arabic letters [22]. ElBadawy *et al.* proposed a system that integrates a set of different types of sensors to capture all sign features. They used LMC in order to capture hands with finger movements, and also used two digital cameras to capture face features and body movement. They applied their system on 20 dynamic signs and the system achieved an accuracy of

95% [23]. Aliyu *et al.* proposed a Kinect-based system for ArSL recognition; the system was applied to 20 signs, and they used linear discriminant analysis for feature reduction and sign classification. Furthermore, fusion from RGB and depth sensor was carried out at feature and the decision level and it achieved an overall accuracy of 99.8% [24]. Jmaa *et al.* proposed a new approach based on hand gesture analysis for ArSL alphabet recognition by extracting a histogram of oriented gradient features from a hand image and then using them to train SVM models. Their approach involved three steps: 1) hand detection and localization using a Microsoft Kinect camera; 2) hand segmentation; and 3) feature extraction using Arabic alphabet recognition. The results showed an accuracy of about 90% [25]. Mohandes *et al.* developed a new model for ArSL recognition in order to detect and track at least one hand and one finger; two different sensors in two different locations in a room generate 3-D interaction space. They used a classifier integrated with two different sensors, LMC and Microsoft Kinect, and 28 Arabic alphabet signs were performed in the interaction space [26]. Almasre and Al-Nuaim proposed a model to recognize the hand gestures of ArSL words using two depth sensors. They examined 143 signs gestured by 10 users for 5 ArSL words. The sensors captured depth images of the upper human body, from which 235 angles (features) were extracted for each joint and between each pair of bones. The dataset was divided into a training set (109 observations) and a testing set (34 observations). They used an SVM classifier with different parameters in order to proceed with four SVM models, with linear kernel (SVMLD and SVMLT) and radial kernel (SVMRD and SVMRT) functions. The accuracy of the words in the training set for the SVMLD, SVMLT, SVMRD, and SVMRT models was 88.92%, 88.92%, 90.88%, and 90.884%, respectively. The accuracy of the testing set for SVMLD, SVMLT, SVMRD, and SVMRT was 97.059%, 97.059%, 94.118%, and 97.059%, respectively [10]. Several sign language recognition research works were performed based on data fusion. Rashid *et al.* developed a multimodal system in order to combine both gestures and postures for recognizing alphabets and numbers; the fusion was done on the decision level. The gesture recognition system was trained using HMM and was concerned with the dynamic motion. The posture recognition system was trained using SVM and was concerned with the static hand at the same time. They applied Gaussian distribution on the captured 3-D depth information to detect and segment gestures and postures. Then, feature vectors were constructed and extracted from spatial and temporal hand properties. Finally, they used the rule of AND/OR combination to state the decision; the model achieved an accuracy of 98% for alphabet and number gestures [27]. Song *et al.* introduced a model of gesture recognition using Microsoft Kinect. The 3-D position data regarding all body skeleton joints were captured using Kinect, and then the features of interest for each gesture were extracted. They segmented the gestures in real time and finally applied the data fusion approach on the decision level by combining the decision of the trained Gaussian mixture model and HMM. They applied their model on seven common gestures and achieved an accuracy of 94.36% [28]. Kishore and Rajesh Kumar presented an Indian sign language recognition system. They extracted the features from the captured video using two algorithms, Fourier descriptions and principal component analysis, and finally performed fusion on the level of features and applied a Sugeno-type fuzzy inference system. The system was applied on 80 common Indian signs and achieved an accuracy of 96% [29]. Penelle and Debeir proposed a data fusion system using Leap Motion and Microsoft Kinect sensors to improve hand recognition accuracy [30]. ElBadawy *et al.* proposed a hybrid system using LMC and two digital cameras. They used LMC for finger tracking and the digital cameras for body movement tracking with facial emotions. The proposed system was applied by a neural network on 20 Arabic signs with an accuracy of 95% [23]. Marin *et al.* proposed a framework to recognize static American signs. They used LMC for fingers and captured features based on distance, while Kinect was used for capturing features based on body and correlation. The proposed system applied SVM with an accuracy of 91% [31]. Fok *et al.* proposed a data fusion system based on two devices. Kalman filter was used for fusion and HMM was used for sign recognition. They applied the system on ten American digits [32]. Yang *et al.* proposed an optimized framework based on a tree structure classification model using three sensors, sEMG, ACC, and GYRO, to get the best performance as a single sensor, two-sensor fusion, and three-sensor fusion. The final recognition rates were 94.31% and 87.02% for 150 Chinese sign language subwords by two test scenarios [33]. Sadek *et al.* proposed a hand gesture recognition using a smart glove that was designed from a set of sensors; the recognition was based on a statistical analysis of the hand shape while performing the 1300 words of the ArSL [20]. Kumar *et al.* proposed a multisensor fusion framework for sign language recognition using a coupled HMM. They used Microsoft Kinect and LMC [34]. Sun *et al.* proposed a weighted fusion method based on Dempster–Shafer evidence theory (DST). The proposed recognition method depends on Kinect and sEMG signal. The average recognition rate was about 87% [35]. Mohandes *et al.* proposed ArSL recognition for the Arabic alphabets using two LMCs and applied DST. They tested the system using ten cross-validations. The first LMC achieved an accuracy of 93.077% and the second LMC achieved an accuracy of 89.907%. Then, they applied the DST on the feature level and on the decision level. The achieved accuracy reached 97.686% and 97.053%, respectively [26]. The main contribution of our proposed model is applying the data fusion on the decision level by combining the results of the two classifiers, KNN and SVM, using an effective fusion technique (DSmT). This combination enhanced the accuracy of the system and made it robust rather than depending on single classifier. We have to mention that ArSL is not a unified language and varies from one country to another, so we focused on the Egyptian ArSL that is most generally comprehended by Arabs. The works of data fusion in sign language recognition, especially the ArSL, are very rare, so our research introduces a way for improving the existing sign language recognition systems by applying the concept of data fusion techniques that make the system robust and more reliable.

III. METHODOLOGY

In this section, we will discuss the structure of our proposed system for ArSL recognition using Microsoft Kinect. We describe the various phases of our system from capturing the gestures using Kinect till the gesture recognition. The structure of the proposed model is shown in Fig. 1. The first step in the proposed model is the data acquisition phase that occurs when the Kinect depth camera starts capturing the skeleton of standing signer in front of the Kinect camera and infers his/her skeleton positions; the system receives joint information such as type and coordinates, bone orientation, and motion velocity as a stream of frames. The preprocessing phase includes extracting the features of interest for both signer skeleton joints and signer hands. Normalization is applied on the collected frames to overcome mainly two problems: first, the variation of user position; and second, the variation of users' sizes. Feature integration is used for fusing the hand features with the skeleton features in order to form the final feature vector. Two classifiers, SVM and KNN, are applied and each one works separately on the feature vector; these classifiers work as two sources of information and the results of each classifier can be considered as the basic belief assignment (BBA). The late fusion is applied using DSmT to combine the BBA of both SVM and KNN; this includes applying the BBA fusion, applying proportional conflict redistribution 5 (PCR5) rule, calculating the pignistic probability, and finally recognizing the performed sign according to the pignistic probability. We compared the results of the DSmT with the Ad-Boosting algorithm and also applied the direct matching techniques (DTW and HMM) instead of the fusion model as an alternative method for recognition in order to compare between the fusion model and other dynamic gesture recognition techniques.

A. Data Acquisition

In this step, we used Microsoft's Kinect Version 2.0 to track the skeleton joints of the standing signer. Kinect provides the information regarding color, depth, and

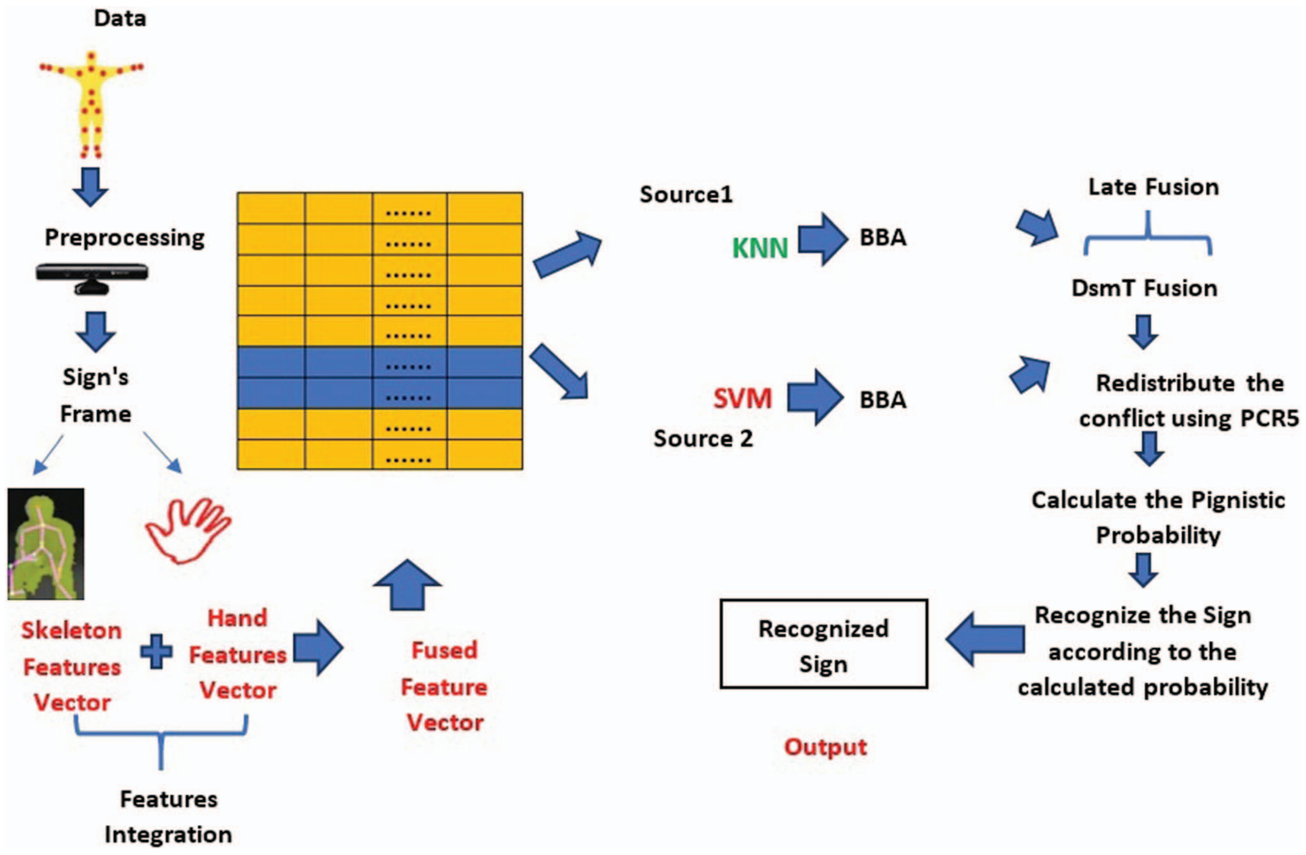


Fig. 1. Proposed model structure.

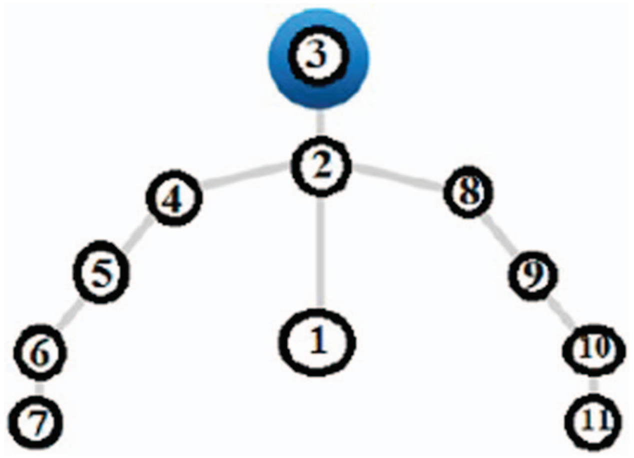


Fig. 2. Points of the upper human body joint: 1—spine, 2—shoulder center, 3—head, 4—right shoulder, 5—right elbow, 6—right wrist, 7—right hand, 8—left shoulder, 9—left elbow, 10—left wrist, and 11—left hand.

joint coordinates using its open-source SDK. The depth information is captured frame by frame. So, when Kinect depth camera starts, we capture the coordinates of 20 skeleton joints with a rate of 30 frames per second. In our system, we are interested in the upper human joint points as shown in Fig. 2.

B. Preprocessing

In this step, we are concerned with the feature extraction, preparation, and normalization for both skeleton and hands of the signer.

1) Feature Extraction The feature extraction step has a very important role in distinguishing between the captured signs. The features are extracted from the sequences of depth information. The extracted features from Kinect frames are divided into two parts: 1) skeleton joint features and 2) hand features.

a) Skelton joint features Kinect has the ability to infer the positions of the detected objects, after studying the selected signs carefully. We found that only ten joints of the skeleton are required to represent and describe the sign. These joints are hand (left and right), shoulder (left and right), elbow (left and right), wrist (left and right), spine mid, and head center. All signs are represented and performed with the upper part of the body and the lower part remains static while performing the sign. The captured frames are required to be normalized in order to overcome the variation in signer's position and signer's size.

Position normalization The signer can be in any position while performing the sign as shown in Fig. 3 and this variation can make a conflict to the model, so we performed the position normalization. The captured coordinates (X, Y, Z) for any joint are scaled by subtracting them from the spine-mid coordinates.

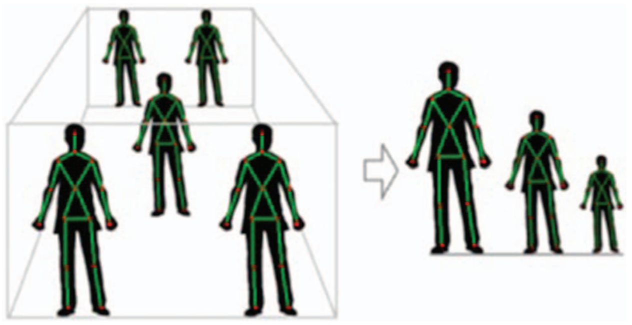


Fig. 3. Position normalization.

The coordinates of the selected joints will be converted from Cartesian coordinates X, Y, and Z into spherical coordinates that are represented by (ϕ, θ, r) as shown in Fig. 4.

The computation of the spherical coordinates is illustrated in the following equations:

$$\sum_{i=1}^{n} r(i) = \sqrt{(J(i)_x - \mathrm{S\_M}_x)^2 + \left(J(i)_y - \mathrm{S\_M}_y\right)^2 + (J(i)_z - \mathrm{S\_M}_z)^2} \quad (1)$$

$$\sum_{i=1}^{n} \theta(i) = a \tan 2 \times \left(\sqrt{(J(i)_x - \mathrm{S\_M}_x)^2 + \left(J(i)_y - \mathrm{S\_M}_y\right)^2}, J(i)_z - \mathrm{S\_M}_z\right) \quad (2)$$

$$\sum_{i=1}^{n} \phi(i) = a \tan 2 \left(\left(J(i)_y - \mathrm{S\_M}_y\right), (J(i)_x - \mathrm{S\_M}_x)\right) \quad (3)$$

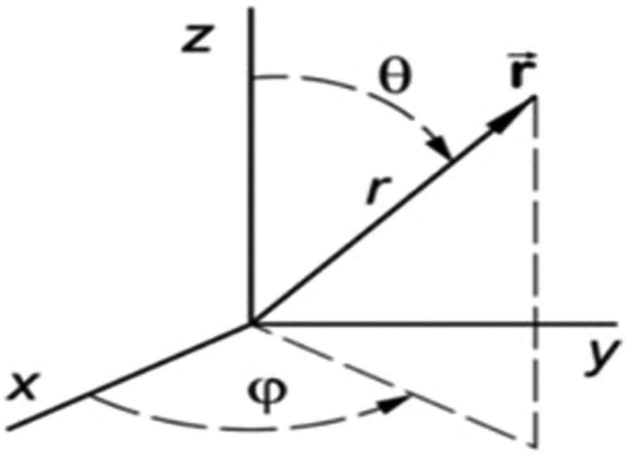


Fig. 4. Spherical coordinates.

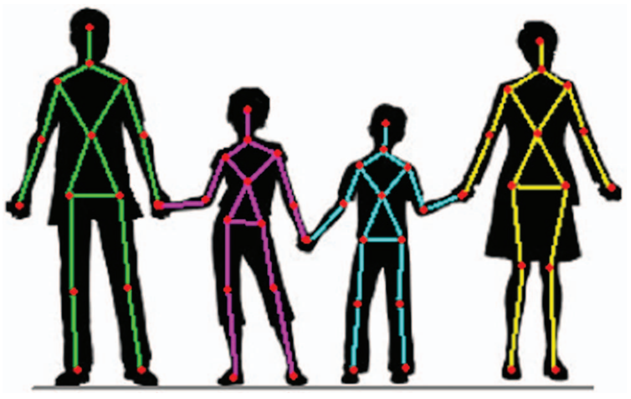


Fig. 5. Size normalization.

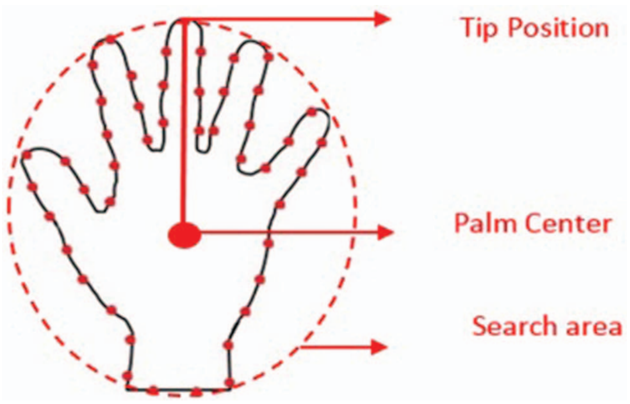


Fig. 7. Hand detection.

where
n is the number of joints from J, r is the radial distance,
S_M_x is the x coordinate of spine-mid joint,
S_M_y is the y coordinate of spine-mid joint, and
S_M_z is the z coordinate of spine-mid joint.

Size normalization To overcome the problem arising from the variation of user's size as shown in Fig. 5, we normalized all the distances that result from the position normalization step by a factor; in our model, we chose this factor as (r_{H,S_M}), which is the distance between the head and spine mid as given in the following equation:

$$\sum_{i=1}^{n} r_{\text{norm}}(i) = \frac{r(i)}{r_{H,S\_M}} \tag{4}$$

where
n is the number of joints from J,
r_{norm} is the normalized radial distance of the joint, and
r_{H,S_M} is the radial distance from head center to spine mid.

Finally, we selected another subset feature added to the spherical coordinates of the selected joints in order to enhance the recognition process as the difference in distance between hand (left and right) and shoulder (left and right). The total number of Kinect features (f_S) is about 32 in spherical coordinates. These features are denoted by ($f_1, f_2, f_3, \ldots, f_{32}$), where the feature vector consists of two sets:

1) $\{r, \emptyset\}$ of right, left {hand, wrist, elbow, shoulder} position.

$\text{Hand}_{\text{Left }r}$, $\text{Hand}_{\text{Right }r}$, $\text{Wrist}_{\text{Left }r}$, $\text{Wrist}_{\text{Right }r}$, $\text{Elbow}_{\text{Left }r}$, $\text{Elbow}_{\text{Right }r}$, Shoulder_Left_r, Shoulder_Right_r

$\text{Hand}_{\text{Left }\phi}$, $\text{Hand}_{\text{Right }\phi}$, $\text{Wrist}_{\text{Left }\phi}$, $\text{Wrist}_{\text{Right }\phi}$, $\text{Elbow}_{\text{Left }\phi}$, $\text{Elbow}_{\text{Right }\phi}$, $\text{Shoulder\_Left}_\phi$, $\text{Shoulder\_Right}_\phi$

2) $\{r\}$ of separation between right and left {hand, wrist, elbow, shoulder} as shown in Fig. 6.

b) Hand features Adding the hand features to the skeleton features will give a complete view and accurate description of the performed sign. The extraction of the hand features is based on the algorithm in [36]. The methodology of hand feature extraction starts by detecting hand joints of the tracked human body; the detected coordinates (x, y, z) for the hand represent the palm center. The next step is to specify where the search area of the hand lies; this 3-D area can be limited by the captured hand and tip position as shown in Fig. 7. After specifying the search area, all depth values that do not belong to the hand area can be excluded. The fingers can be detected by applying the algorithm of the convex hull on the search area; the edges of the convex hull above the wrist represent the fingertips as shown in Fig. 8.

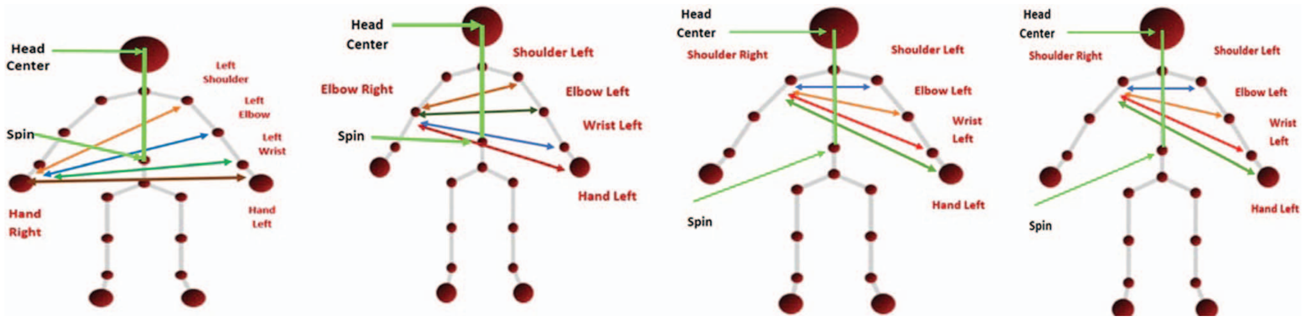


Fig. 6. Features of joint separation. (a) Hand separation. (b) Elbow separation. (c) Elbow separation. (d) Shoulder separation.

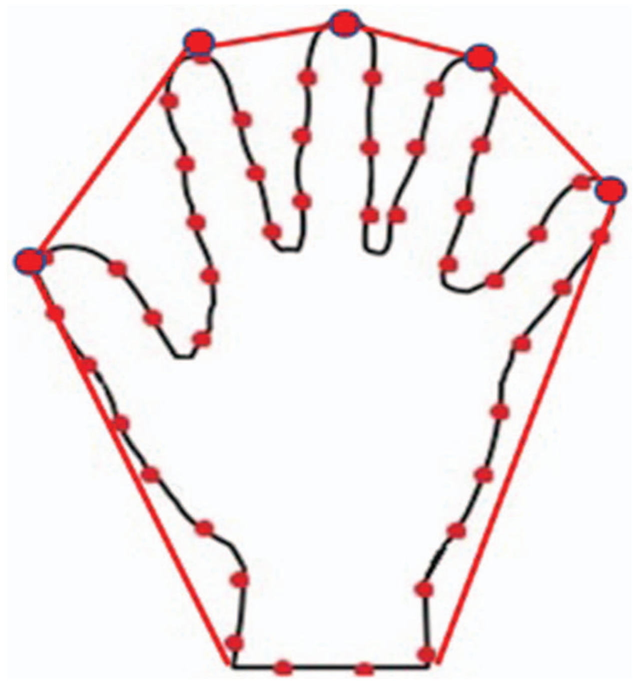


Fig. 8. Convex hull detection.

Finally, the total number of hand features (f_H) is about 30 in Cartesian coordinates. These features are denoted by ($f_1, f_2, f_3, \ldots, f_{30}$) where the feature vector consists of fingertip direction that is composed of 3-D data $\langle x_i, y_i, z_i \rangle$. After fingertip positions are detected, the fingers' direction vectors can be easily calculated by subtracting the tip position of each finger from the palm center P_C (p_x, p_y, p_z). The vectors that are pointing from the palm center to fingers can be calculated using the following equation:

$$V_{\text{Direction}} = \left(\left(f_x - p_x \right), \left(f_y - p_y \right), \left(f_z - p_z \right) \right). \quad (5)$$

c) Feature integration Feature integration is the process of integrating the feature vectors of both skeleton joint features (f_S) and hand features (f_H) in order to produce the fused vector $f_c = \{f_S, f_H\}$. The resultant fused feature vector has a dimension of 62. It should be mentioned that the data sequence is synchronized perfectly because they are coming from the same device.

C. Classification

Once the gesture features have been extracted, the descriptor of gestures that the system must classify will be formed. The goal of our system is to recognize the gestures, so after extracting the features, we applied two classifiers: KNN with $K = 1$ and SVM with RBF kernel function (gamma = 0.48 and cost = 0.5). We chose these classifiers after applying different classifiers on the test set. They gave us the best accuracy, are widely used in many pattern recognition applications such as the handwritten digit recognition [37], and are efficient in dealing with multiclass nonlinear classification problems. These two classifiers work as two sources of information. It is better to use two classifiers rather than using one classifier in order to improve the overall accuracy. The combination of information from different sources is critical, especially when developing a system that depends on conflicting, imprecise, and uncertain data. In the proposed model, each classifier takes the sequence of frames that formed the single presegmented gesture and classifies each frame separately to predict the class that frame belongs to. However, there are similarities between some gestures so that, for example, if frames of gestures enter into the classifier, the output may be classified by 70% of frames as Sign_ID = "1", 10% of frames as Sign_ID = 4, and 20% of frames as Sign_ID = 8. These values were considered as BBA that will enter the fusion phase.

To define BBA, let X be the universe that represents all possible states of a system under consideration. In the evidence theory, the BBA assigns belief mass to each element of the power set formed from the underlying universe X. We can consider the function $m : 2^x \rightarrow [0, 1]$ as a BBA, when two conditions occurred:

1) The mass of the empty set is 0 (i.e., $m(\emptyset) = 0$).
2) The masses of the remaining members of the power set add up to a total of 1 ($\sum_{A \subseteq 2^X} m(A) = 1$).

D. Late Fusion

The late fusion occurs by combining the results of the two classifiers (SVM and KNN) and applying the rules of DSmT. The fusion of these classifiers was done on the measurement level, which is more confident. The evidence (results of the classifier) is considered as BBA.

1) Dezert–Smarandache Theory DSmT is an effective fusion method. It can deal with the uncertainty and the data coming from highly conflicting sources. It allows the combination of information that is coming from different independent sources; this information is represented in terms of belief function. Dezert–Smarandache rules combine the conflict evidence accurately, so it is very successful in problems of object recognition [38].

DSmT is a theory of plausible and paradoxical reasoning; it overcame the limitations of DST [39]. We can summarize the comparison between DST and DSmT as follows.

Let $\vartheta = \{\theta_1, \ldots, \theta_n\}$ is a finite set of hypotheses.

- The DST considers a discrete and finite frame of discernment θ based on a set of exhaustive and exclusive elementary elements θ.
- The bodies of evidence are assumed independent and provide their own belief function on the power set θ but with same interpretation for θ [39].

DSmT has two types of models: 1) a free model that combines the evidence without taking the integrity constraint into consideration and 2) a hybrid model that

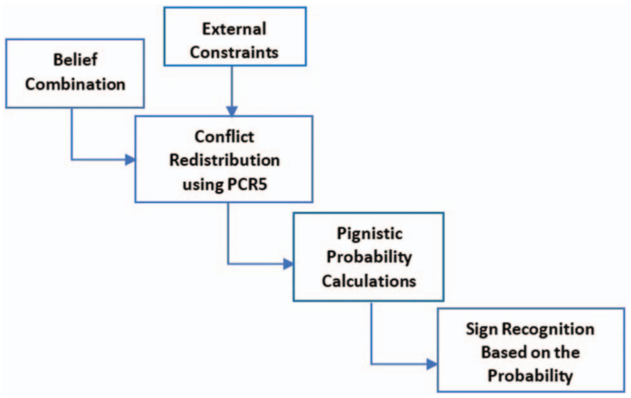

Belief Combination
External Constraints
Conflict Redistribution using PCR5
Pignistic Probability Calculations
Sign Recognition Based on the Probability

Fig. 9. Fusion framework.

includes all operators such as union and intersection and the constraints that are required to build the class Φ, so it is used in real applications.

Based on that model, the hyper-power set is given by $\vartheta = \{\theta_1, \ldots, \theta_n\}$ as a finite set (called frame) of n exhaustive elements. The free Dedekind's lattice denoted hyper-power set D^θ is defined as

1) $\emptyset, \theta_1, \ldots, \theta_n \in D^\theta$.
2) If $A,\ B \in D^\theta$, then $A \cap B$ and $A \cup B$ belong to D^θ.
3) No other elements belong to D^θ, except those obtained by using rules 1 or 2 [38].

2) Basic Belief Assignment For any finite discrete frame ϑ, we define a belief assignment as a mapping $m(\cdot)$: $G^\theta \rightarrow$ [0, 1] associated with a given body of evidence, B, that satisfies the following conditions:

$$m(\emptyset) = 0 \quad \text{and} \quad \sum_{A \in G^\theta} m(A) = 1 \tag{6}$$

where G^θ is a hyper-power set of Θ, which is $= \{\emptyset, \theta_1, \theta_2, \theta_1 \cap \theta_2, \theta_1 \cup \theta_2\}$.

In (6), $m(A)$ is the generalization of BBA/mass, where the belief function is defined as

$$\mathrm{Bel}(A) \triangleq \sum_{\substack{B \subseteq A \\ B \in G^\theta}} m(B). \tag{7}$$

In DSmT, there is a two-level process: credal (for combination of evidences) and pignistic (for decision making); i.e., when we need to take a decision, we should depend on a probability function. The classical pignistic probability transformation is defined as [38]

$$\mathrm{BetP}\{A\} = \sum_{X \in 2^\theta} \frac{|X \cap A|}{|X|} m(X) \tag{8}$$

where $|x|$ denotes the cardinality of x (with convention $|\emptyset|/|\emptyset| = 1$, when defining $\mathrm{BetP}\{\emptyset\}$). Decisions are achieved by computing the expected utilities of the acts using the subjective/pignistic $\mathrm{BetP}\{\cdot\}$ as the probability function needed to compute expectations. It is easy to show that $\mathrm{BetP}\{\cdot\}$ is a proper probability function [38].

TABLE I
BBA: Stage 1

	"1"	"2"	"3"	"4"	"5"	"6"	"7"	"8"	"9"	"10"
BBA (S1/KNN)	0.8	0	0	0.2	0	0	0	0	0	0
BBA (S2/SVM)	0.75	0.25	0	0	0	0	0	0	0	0

3) Fusion Framework As introduced earlier, we used the DSmT for the beliefs of each evidence, and then applied the combination rule. We summarize the fusion framework in Fig. 9.

The belief calculation is computed using (6) and (7) and then the conflict is redistributed using PCR5 rule, which is the mathematical form to redistribute the conflicting mass to nonempty sets; the conflict mass should be distributed to the elements that are involved in the partial conflict with respect to their mass, considering the canonical form of the partial conflict as in (9). Finally, the pignistic probability is calculated using (8) in order to decide the performed sign according to the highest probability.

$$m_{\mathrm{PCR5}}(X) = m_{12}(X) + \sum_{\substack{Y \in G^\theta\{X\} \\ X \cap Y = \emptyset}} \left[\frac{m_1(X)^2 m_2(Y)}{m_1(X) + m_2(Y)} + \frac{m_2(X)^2 m_1(Y)}{m_2(X) + m_1(Y)} \right]. \tag{9}$$

The dataset contains 40 signs, so it is divided into four parts in order to simplify the calculation. Sign_ID = "1" is chosen as a common sign between the divided datasets in order to relate them to each other. When the test sign enters into the system, it will pass four stages of fusion with each divided dataset. The goal is to calculate the ranked pignistic probability in order to recognize the performed sign.

The following calculation represents "tested sign with Sign_ID = 1" when entered into the fusion framework. As mentioned earlier, the first step of the model is applying the classification using the two classifiers (SVM and KNN) as two sources of information. Table I presents the results of the two classifiers, which are considered as BBA. The first stage of fusion is done with the first dataset that contains the signs with ID = 1,2, …, 10.

The second step is applying the classical DSM combination rule. Table II presents the fusion results of the

TABLE II
Fusion Output: Stage 1

	"1"	"2"	"4"	"1 ∩ 2"	"1 ∩ 4"	"2 ∩ 4"
m_{DSm}	0.6	0	0	0.2	0.15	0.05

TABLE III
PCR5 Output: Stage 1

	"1"	"2"	"4"	"1 ∩ 2"	"1 ∩ 4"
m_{PCR5}	0.6	0.025	0.025	0.2	0.15

beliefs after applying the fusion rules.

$$m_{12}(A) \triangleq \sum_{\substack{X_1, X_2, \ldots, X_k \int D^\theta \\ (X_1 \cap X_2 \cap \cdots \cap X_k) = A}} \prod_{i=1}^{k} m_i(X_i) \tag{10}$$

$$(1) = m_1(1) \cdot m_2(1) = 0.8 \times 0.75 = 0.6$$

$$(2) = m_1(2) \cdot m_2(2) = 0 \times 0.25 = 0$$

$$(4) = m_1(4) \cdot m_2(4) = 0.2 \times 0 = 0$$

$$(1 \cap 2) = m_1(1) \cdot m_2(2) + m_2(1) \cdot m_1(2) = 0.2$$

$$(1 \cap 4) = m_1(1) \cdot m_2(4) + m_2(1) \cdot m_1(4) = 0.15$$

$$(2 \cap 4) = m_1(2) \cdot m_2(4) + m_2(2) \cdot m_1(4) = 0.05.$$

Consequently, redistribute the conflict factor using PCR5 rule.

Redistribute: "2 ∩ 4 = Φ".

So, we will distribute this conflict proportionally.

$$m_{12}(2) = 0.025$$

$$m_{12}(4) = 0.025.$$

Table III presents the values of the beliefs after applying PCR5 rule.

The pignistic probability can be obtained from the above beliefs using (8). Table IV presents the pignistic probability.

$$\text{CM}(1) = 3,\ \text{CM}(2) = 2,\ \text{CM}(4) = 2,$$
$$\text{CM}(1 \cap 2) = 1,\ \text{and}\ \text{CM}(1 \cap 4) = 1$$

$$P(1) = \frac{1}{2} \times m_{12}(2) + \frac{1}{3} \times m_{12}(1) + \frac{1}{2} \times m_{12}(4) = 0.225$$

$$P(2) = \frac{1}{2} \times m_{12}(2) + \frac{1}{3} \times m_{12}(1) = 0.2125$$

$$P(4) = \frac{1}{2} \times m_{12}(4) + \frac{1}{3} \times m_{12}(1) = 0.2125$$

$$P(1 \cap 2) = \frac{1}{1} \times m_{12}(1 \cap 2) = 0.2$$

$$P(1 \cap 4) = \frac{1}{1} \times m_{12}(1 \cap 2) = 0.15.$$

TABLE IV
Pignistic Probability Output: Stage 1

	"1"	"2"	"4"	"1 ∩ 2"	"1 ∩ 4"
Probability	0.225	0.2125	0.2125	0.2	0.15

TABLE V
BBA: Stage 2

	"1"	"11"	"12"	"13"	"14"	"15"	"16"	"17"	"18"	"19"	"20"
BBA (S1/KNN)	0.64	0	0	0	0.2	0.16	0	0	0	0	0
BBA (S2/SVM)	0.7	0	0	0	0.2	0.1	0	0	0	0	0

Again, reprocess the sign in the second stage with the second dataset, where ID = 11, 12, …, 20. Table V presents the BBA for the second group.

Apply the classical DSM combination rule as in (10). Table VI presents the fusion results of the second group.

$$(1) = m_1(1) \cdot m_2(1) = 0.64 \times 0.7 = 0.469$$

$$(14) = m_1(14) \cdot m_2(14) = 0.2 \times 0.2 = 0.04$$

$$(15) = m_1(15) \cdot m_2(15) = 0.16 \times 0.1 = 0.016$$

$$(1 \cap 14) = m_1(1) \cdot m_2(14) + m_2(1) \cdot m_1(14) = 0.268$$

$$(1 \cap 15) = m_1(1) \cdot m_2(15) + m_2(1) \cdot m_1(15) = 0.176$$

$$(14 \cap 15) = m_1(2) \cdot m_2(4) + m_2(2) \cdot m_1(4) = 0.052.$$

Consequently, redistribute the conflict factor using PCR5 rule. Table VII presents the beliefs of the second group after redistributing the conflict using PCR5 rule.

Redistribute: Φ = "14 ∩ 15".

So, we will distribute this conflict proportionally.

$$m_{12}(14) = 0.04 + (0.72 \times 0.052) = 0.07744$$

$$m_{12}(15) = 0.016 + (0.28 \times 0.052) = 0.03056.$$

The pignistic probability can be obtained from the above beliefs using (8). Table VIII presents the pignistic probability.

$$\text{CM}(1) = 3,\ \text{CM}(2) = 2,\ \text{CM}(4) = 2,$$
$$\text{CM}(1 \cap 2) = 1,\ \text{and}\ \text{CM}(1 \cap 4) = 1$$

$$P(1) = \frac{1}{2} \times m_{12}(14) + \frac{1}{3} \times m_{12}(1) + \frac{1}{2} \times m_{12}(15) = 0.203$$

$$P(14) = \frac{1}{2} \times m_{12}(14) + \frac{1}{3} \times m_{12}(1) = 0.18853$$

$$P(15) = \frac{1}{2} \times m_{12}(15) + \frac{1}{3} \times m_{12}(1) = 0.16447$$

$$P(1 \cap 14) = \frac{1}{1} \times m_{12}(1 \cap 14) = 0.268$$

$$P(1 \cap 15) = \frac{1}{1} \times m_{12}(1 \cap 15) = 0.176.$$

TABLE VI
Fusion Output: Stage 2

	"1"	"14"	"15"	"1 ∩ 14"	"1 ∩ 15"	"14 ∩ 15"
m_{DSm}	0.448	0.04	0.016	0.268	0.176	0.052

TABLE VII
PCR5 Output: Stage 2

	"1"	"14"	"15"	"1 ∩ 14"	"1 ∩ 15"
m_{PCR5}	0.448	0.07744	0.03056	0.268	0.176

Again, reprocess the sign in the third stage with the third dataset, where ID = 21, 22, …, 30. Table IX presents the BBA of the third group.

Apply the classical DSM combination rule as in (10). Table X presents the fusion results of the third group.

$$(1) = m_1(1) \cdot m_2(1) = 0.8 \times 0.7 = 0.56$$

$$(24) = m_1(24) \cdot m_2(24) = 0.2 \times 0.1 = 0.02$$

$$(26) = m_1\ (26) \cdot m_2(26) = 0 \times 0.2 = 0$$

$$(1 \cap 24) = m_1(1) \cdot m_2(24) + m_2(1) \cdot m_1(24) = 0.22$$

$$(1 \cap 26) = m_1(1) \cdot m_2(26) + m_2\ (1) \cdot m_1(26) = 0.16$$

$$(24 \cap 26) = m_1(2) \cdot m_2(4) + m_2(2) \cdot m_1(4) = 0.04.$$

Consequently, redistribute the conflict factor using PCR5 rule. Table XI presents the beliefs of the third group after redistributing the conflict using PCR5 rule.

Redistribute: = Φ"24 ∩ 26".

So, we will distribute this conflict proportionally.

$$m_{12}(24) = 0.02 + (0.04) = 0.06.$$

The pignistic probability can be obtained from the above beliefs using (8). Table XII presents the pignistic probability.

$$\text{CM}(1) = 3,\ \text{CM}(2) = 2,\ \text{CM}(4) = 2,$$

$$\text{CM}(1 \cap 2) = 1,\ \text{and CM}(1 \cap 4) = 1$$

$$P(1) = \frac{1}{2} \times m_{12}(24) + \frac{1}{3} \times m_{12}(1)$$

$$+\frac{1}{2} \times m_{12}(26) = 0.2167$$

$$P(24) = \frac{1}{2} \times m_{12}(24) + \frac{1}{3} \times m_{12}(1) = 0.217$$

$$P(26) = \frac{1}{2} \times m_{12}(15) + \frac{1}{3} \times m_{12}(1) = 0.1867$$

$$P(1 \cap 24) = \frac{1}{1} \times m_{12}(1 \cap 14) = 0.22$$

$$P(1 \cap 26) = \frac{1}{1} \times m_{12}(1 \cap 15) = 0.16.$$

TABLE VIII
Pignistic Probability Output: Stage 2

	"1"	"14"	"15"	"1 ∩ 14"	"1 ∩ 15"
DSm	0.203	0.18853	0.16447	0.268	0.176

TABLE IX
BBA: Stage 3

	"1"	"21"	"22"	"32"	"24"	"25"	"26"	"27"	"28"	"29"	"30"
BBA (S1/KNN)	0.8	0	0	0	0.2	0	0	0	0	0	0
BBA (S2/SVM)	0.7	0	0	0	0.1	0	0.2	0	0	0	0

Again, reprocess the sign in the fourth stage with the fourth dataset, where ID = 31, 32, …, 40. Table XIII presents the BBA of the fourth group.

Apply the classical DSM combination rule as in (10). Table XIV presents the fusion results of the fourth group.

$$(1) = m_1(1) \cdot m_2(1) = 0.9 \times 0.95 = 0.855$$

$$(35) = m_1(35) \cdot m_2(35) = 0.05 \times 0.03 = 0.0015$$

$$(39) = m_1(39) \cdot m_2(39) = 0.05 \times 0.2 = 0.001$$

$$(1 \cap 39) = m_1(1) \cdot m_2(39) + m_2(1) \cdot m_1(39) = 0.0655$$

$$(1 \cap 35) = m_1(1) \cdot m_2(35) + m_2(1) \cdot m_1(35) = 0.0745$$

$$(39 \cap 35) = m_1(39) \cdot m_2(35)$$

$$+m_2(39) \cdot m_1(35) = 0.0025.$$

Consequently, redistribute the conflict factor using PCR5 rule. Table XV presents the beliefs of the fourth group after redistributing the conflict using PCR5 rule.

Redistribute: Φ = "35 ∩ 39".

So, we will distribute this conflict proportionally.

$$m_{12}(39) = 0.001 + (0.001) = 0.002$$

$$m_{12}(35) = 0.0015 + (0.0015) = 0.003.$$

The pignistic probability can be obtained from the above beliefs using (8). Table XVI presents the pignistic probability.

$$\text{CM}(1) = 3,\ \text{CM}(2) = 2,\ \text{CM}(4) = 2,$$

$$\text{CM}(1 \cap 2) = 1,\ \text{and CM}(1 \cap 4) = 1$$

$$P(1) = \frac{1}{2} \times m_{12}(39) + \frac{1}{3} \times m_{12}(1)$$

$$+\frac{1}{2} \times m_{12}(35) = 0.2875$$

$$P(35) = \frac{1}{2} \times m_{12}(35) + \frac{1}{3} \times m_{12}(1) = 0.2865$$

$$P(39) = \frac{1}{2} \times m_{12}(15) + \frac{1}{3} \times m_{12}(1) = 0.286$$

TABLE X
Fusion Output: Stage 3

	"1"	"24"	"26"	"1 ∩ 24"	"1 ∩ 26"	"24 ∩ 26"
m_{DSm}	0.56	0.02	0	0.22	0.16	0.04

TABLE XI
PCR5 Output: Stage 3

	"1"	"24"	"26"	"1 ∩ 24"	"1 ∩ 26"
m_{PCR5}	0.56	0.06	0	0.22	0.16

$$P(1\cap 39)=\frac{1}{1}\times m_{12}(1\cap 14)=0.0655$$

$$P(1\cap 35)=\frac{1}{1}\times m_{12}(1\cap 15)=0.0745.$$

Table XVII shows the results of pignistic probabilities of the four stages by combining Tables IV, VIII, XII, and XVI in which the Sign_ID is chosen by taking the higher probability value. So, from Table XVII the performed sign is Sign_ID = 1 with the highest probability of 0.2875.

E. Ada-Boosting and Majority Voting

It is one of the fusion techniques that was first introduced by Freund and Schapire in 1996. The ensemble classifier is constructed from multiple weak classifiers. The single classifier can act poorly, but the results of multiple classifiers are expected to be more accurate. We applied the Ada-Boosting method on the two classifiers (SVM and KNN) and because we deal with dynamic gestures (i.e., the captured sign cannot be represented by single frame), so after applying the Ada-Boosting on the received frames, the result is finally selected by majority voting. The main idea of Ada-Boosting is to give higher importance to the more accurate classifiers in the sequence, so it starts by giving equal weights to each observation in dataset. If the prediction of the first classifier is incorrect, then it gives the highest weight to the observation with incorrect prediction. Fig. 10 presents the Ada-Boosting of two classifiers (SVM and KNN).

The algorithm of Ada-Boosting classifier is as follows.

TABLE XIII
BBA: Stage 4

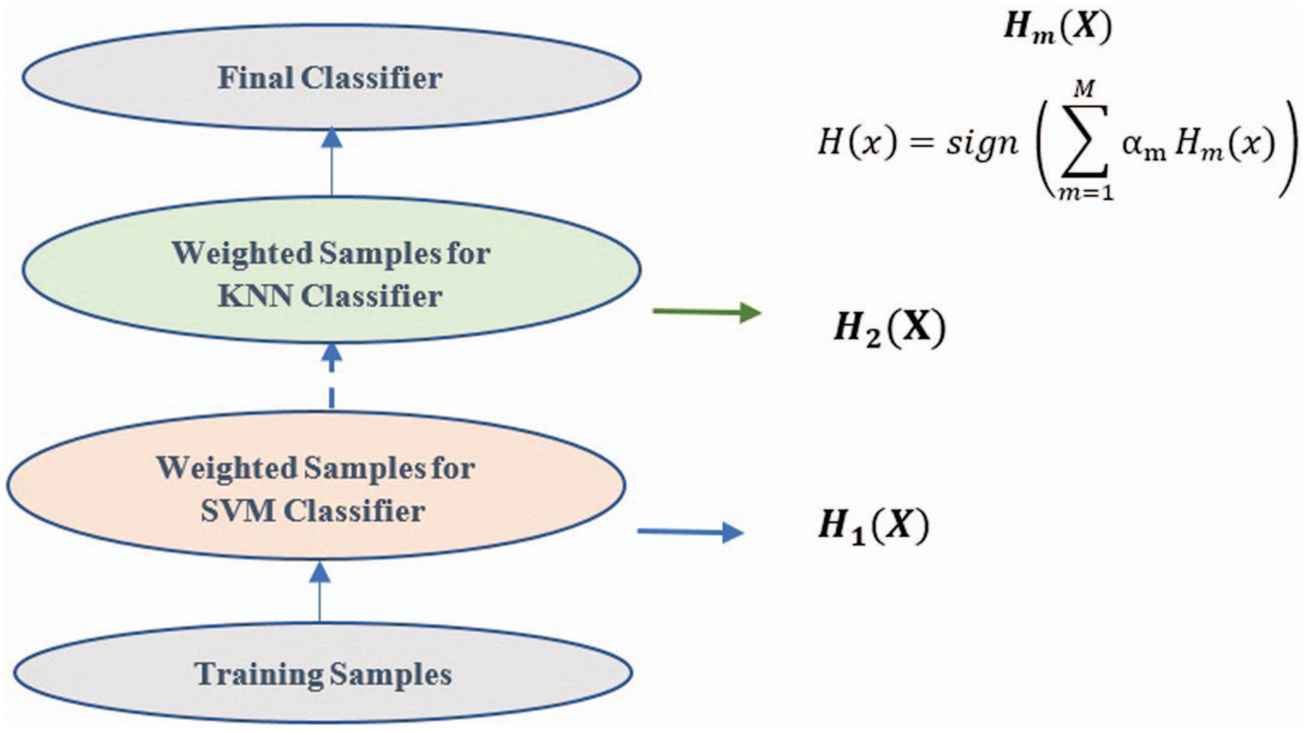
	"1"	"31"	"32"	"33"	"34"	"35"	"36"	"37"	"38"	"39"	"40"
BBA (S1/KNN)	0.9	0	0	0	0	0.05	0	0	0	0.05	0
BBA (S2/SVM)	0.95	0	0	0	0	0.03	0	0	0	0.02	0

TABLE XII
Pignistic Probability Output: Stage 3

	"1"	"24"	"26"	"1 ∩ 24"	"1 ∩ 26"
DSm	0.2167	0.2167	0.1867	0.22	0.16

TABLE XIV
Fusion Output: Stage 4

	"1"	"39"	"35"	"1 ∩ 39"	"1 ∩ 35"	"35 ∩ 39"
m_{DSm}	0.855	0.001	0.0015	0.0655	0.0745	0.0025



$H_m(X)$
$H(x) = sign\left(\sum_{m=1}^{M} \alpha_m H_m(x)\right)$
Final Classifier
Weighted Samples for KNN Classifier
$H_2(X)$
Weighted Samples for SVM Classifier
$H_1(X)$
Training Samples

Fig. 10. Ada-Boosting structure.

TABLE XV
PCR5 Output: Stage 4

	"1"	"39"	"35"	"1 ∩ 39"	"1 ∩ 35"
m_{PCR5}	0.855	0.002	0.003	0.0655	0.0745

Table XVI
Pignistic Probability Output: Stage 4

	"1"	"39"	"35"	"1 ∩ 39"	"1 ∩ 35"
DSm	0.2875	0.286	0.2865	0.0655	0.0745

Input: Let $\{(x_1, y_y), \ldots, (x_n, y_n)\}$ be a training set, where n is the number of patterns.

Output: The ensembled classifier $H(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m H_m(x)\right)$, where m is the number of classifiers (SVM, KNN).

a) Initialize the weights w_i, $w_i = 1/n$, where $i \in \{1, \ldots, n\}$.

From $m = 1$ to M

b) Train the weak classifier H_m with the new weighted training data.
c) Calculate the error rate err_m of the classifier.
d) Calculate the classifier contribution $\alpha_m = 0.5 \times \log[(1 - \mathrm{err}_m)/\mathrm{err}_m]$.
e) Update the classifiers weights $w_i \leftarrow w_i \exp(-\alpha_m I(y_i \neq H_m(x_i)))$, where w_i is the weight of each input sequence and I is an indicator function:

$$I = \begin{cases} -1, & \text{prediction incorrect, then increase } w_i \\ 1, & \text{prediction correct, then decrease } w_i \end{cases}.$$

f) Output the ensemble classifier

$$H(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m H_m(x)\right).$$

F. Dynamic Time Wrapping

DTW is a pattern recognition algorithm that is widely used with dynamic gestures. It applies a direct matching technique because it tries to match the tested gesture with the most similar stored sign in the training set irrespective of the sign's length depending on measuring the distance between the two series. It tries to find the optimal alignment between two time series sequences that are varying in their speed or their time and also have different lengths. For our model, the system receives the sign (test sign) as a set of frames and DTW compares these sequences of frames with stored signs' sequences in the training set; the sequences that are compared must be wrapped in the time dimension to compute the DTW similarity coefficient. The similarity depends on the calculated distance for each sign and then the test sign is matched with the sign that has least distance. Fig. 11 presents the process of alignment for two time-independent sequences. Given two time series $X = (x_1, x_2, x_4, \ldots, x_n)$ with length $n \in N$ and $Y = (y_1, y_2, y_4, \ldots, y_n)$ with length $m \in N$ and let $\mathcal{F}$ be a feature space where $X_n, Y_m \in \mathcal{F}$. DTW will analyze the sequences in order to find the similarities between them and finally find optimal alignment $O(nm)$ [40]. To compare x and y sequences, we need to find the local cost matrix that represents the cost distribution between each two elements in the two sequences as given in the following equation:

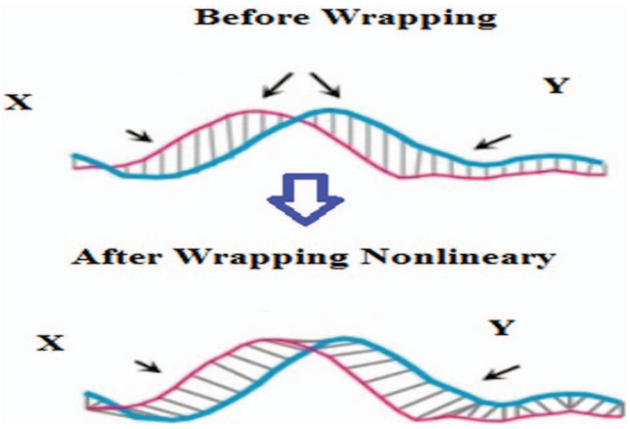


Fig. 11. DTW alignment process.

$$C : \mathcal{F} \times \mathcal{F}. \tag{11}$$

The value of C represents the similarity between the stored sign (x) and the test sign (y); if they are similar, this value must be small, else it must be large, to generate the local cost matrix with a dimension of $(n \times m)$ as shown in Fig. 12. The cost of any position at the local cost matrix $M(i, j)$ can be determined as follows:

$$\begin{aligned} M(n, m) &= d(n, m) + \min\{M(n-1, m-1), \\ &M(n-1, m), M(n, m-1)\}. \end{aligned} \tag{12}$$

This equation has two parts: the first part is the Euclidean distance $d(i, j)$ between the feature vectors

TABLE XVII
Pignistic Probability Output

Sign index	"1"	"2"	"4"	"1 ∩ 2"	"1 ∩ 4"	"14"	"15"	"1 ∩ 14"	"1 ∩ 15"
BetP{·}	0.2875	0.2125	0.2125	0.2	0.15	0.18853	0.16447	0.268	0.176
Sign index	"24"	"26"	1 ∩ 24""	1 ∩ 26""	"39"	"35"	"1 ∩ 39"	"1 ∩ 35"	–
BetP{·}	0.2167	0.1867	0.22	0.16	0.286	0.2865	0.0655	0.0745	–

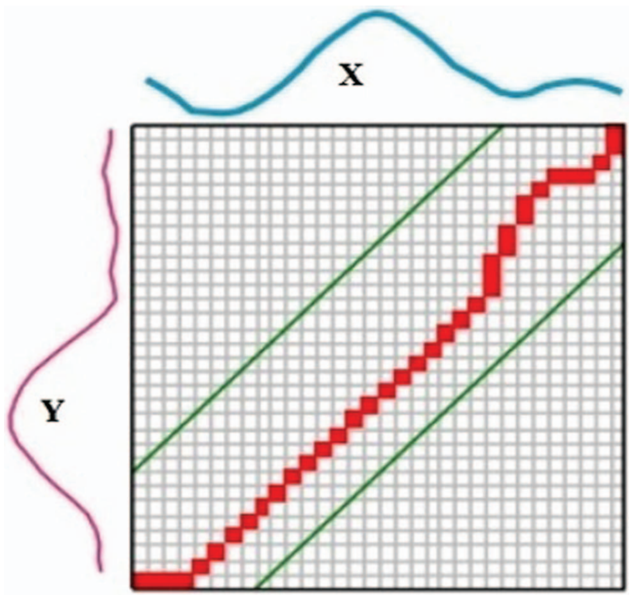


Fig. 12. Local cost matrix.

of the sequences X and Y, and the second part is the minimum cost of the adjacent elements of the cost matrix up to that point [41].

After getting the local cost matrix, we must find the wrapping path through it by applying the following equation to get the wrapping list:

$$\mathrm{wp}_{n,m} = \min\left(c_{n-1,m-1}, c_{n-1,m}, c_{n,m-1}\right) \tag{13}$$

where C is the cost value of each element in the cost matrix.

Finally, apply the following distance equation on the wrapping list in order to calculate the DTW distance:

$$\mathrm{DTW}_d = \frac{1}{p}\sum_{i=1}^{p} w_i \tag{14}$$

where w is the value of each element in the wrapping path.

Fig. 13 is an illustrative example for applying DTW in our model. Let us have a dataset of three stored signs, and a new sign is performed. The DTW algorithm computes the distance between the captured sign and each sign in the training set. Finally, the algorithm matches the sign with the stored sign that corresponds to the least distance.

G. Hidden Markov Model

HMM is a statistical model and time-domain process. It represents the statistical behavior for the observed sequence using a set of hidden states called "hidden network." The model can transition from one state to another with probability assignment [41]. The expression "hidden" comes from the fact that the Markov model constructs a sequence of hidden states from the observed sequence. HMM was successful and achieved a good accuracy with the applications of speech recognition and it is noted that there are similarities between the nature of speech and dynamic gestures [42].

$Q = q_1, q_2, q_3, \ldots, q_n$, a set of n states.

$\pi = \pi_1, \pi_2, \pi_3, \ldots, \pi_n$, the probability distribution over the states.

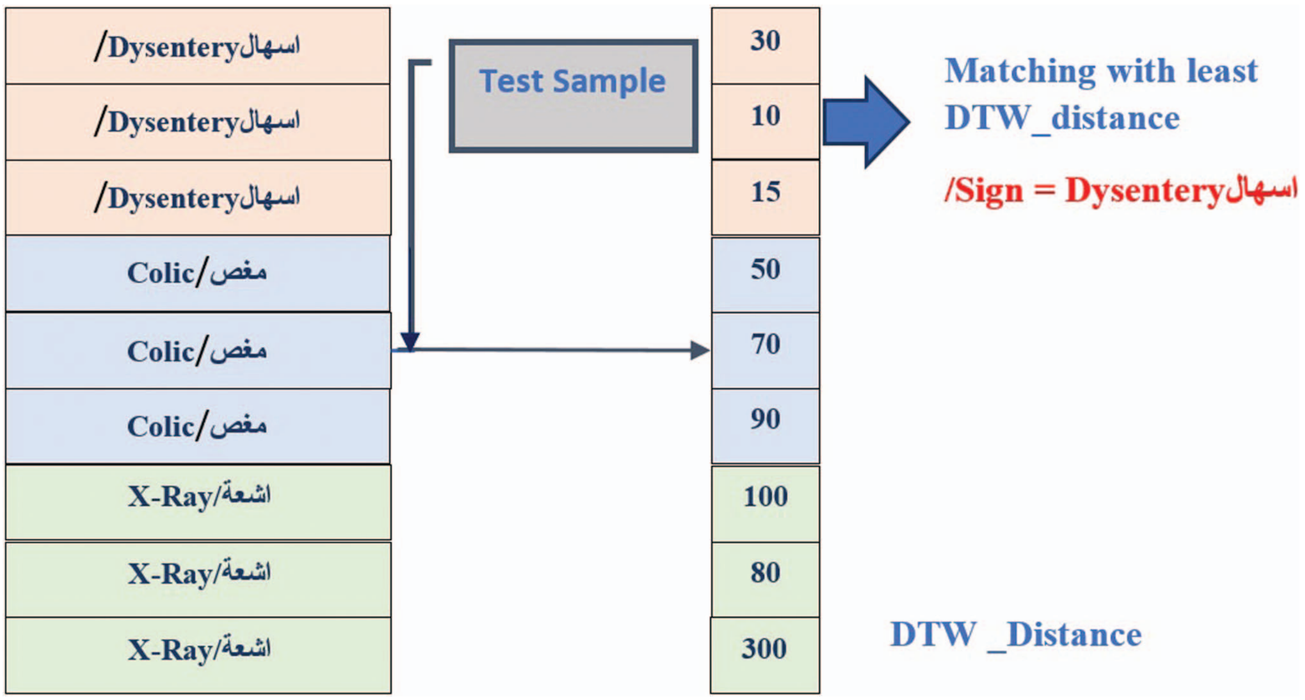


Fig. 13. DTW classification example.

$A = a_{01}, a_{02}, \ldots, a_{n1}, \ldots, a_{nn}$, the matrix A of transition probability that contains the transition probability for the transition from one state to another.

$B = b_j(O_k)$, the observation probability from state j and the observing sequence O_k.

$O = o_1, o_2, \ldots, o_T$, a sequence of T observations.

q_0, q_F, start state and end (final) state.

There are two axioms in the HMM: 1) from the law of probability, the sum of all values on the directed arcs from a given state to other must equal 1 as in (15); and 2) the sum of all π probabilities must equal 1 as in (16).

$$\text{Axiom \#1:} \quad \sum_{i=1}^{n} a_{ij} = 1 \tag{15}$$

$$\text{Axiom \#2:} \quad \sum_{i=1}^{n} \pi_i \,. \tag{16}$$

The Markov model assumed two important assumptions: 1) the probability of each state depends only on the previous state in the state sequence as in (17); and 2) the probability of any observation o_i depends only on the state q_i that produced the observation and not on any other states or any other observations as in (18).

Markov assumption #1:

$$P(q_i|q_1, \ldots, q_{i-1}) = P(q_i|q_{i-1}) \tag{17}$$

Markov assumption #2:

$$P(o_i|q_1, \ldots, q_i, \ldots, q_T, o_1, \ldots, o_i, \ldots, o_T) = P(o_i|q_i). \tag{18}$$

For our model, there are two phases:

1) *Training phase*: In this phase, we fed the model with all gesture sequences and their feature vector to build the model for each sequence and then re-estimate the probability distribution using the Baum–Welch algorithm. Also, K-means clustering is used to clusters all the 3-D sequence's points in the training set into n clusters. This will reduce the data of the stored gestures to a set of discrete states and symbols. Now each point in the training set is converted to a specific symbol that is tightly related to the clustered n states. Fig. 14 presents building of the HMM states for one gesture "Injection/حقن" as an example.
2) *Testing phase*: In this phase, we used the Viterbi decoding algorithm to match the test sign sequence with the stored sign that has highest likelihood L, which is computed using the following equation:

$$L(S_1, \ldots, S_n\,|O_1, \ldots, O_n\,) = \prod_{i=1}^{n} P\,(S_i|O_i) \cdot \prod_{i=1}^{n} P\,(S_i|O_{i-1})\,. \tag{19}$$

Fig. 15 presents the workflow of the system when using the HMM in dynamic gesture recognition.

IV. EXPERIMENTAL RESULTS

The experimental results have two aspects:

1) Recognition accuracy.
2) Latency (execution time).

First, we applied the proposed model using Microsoft Kinect V2, which consists of an IR emitter, an RGB

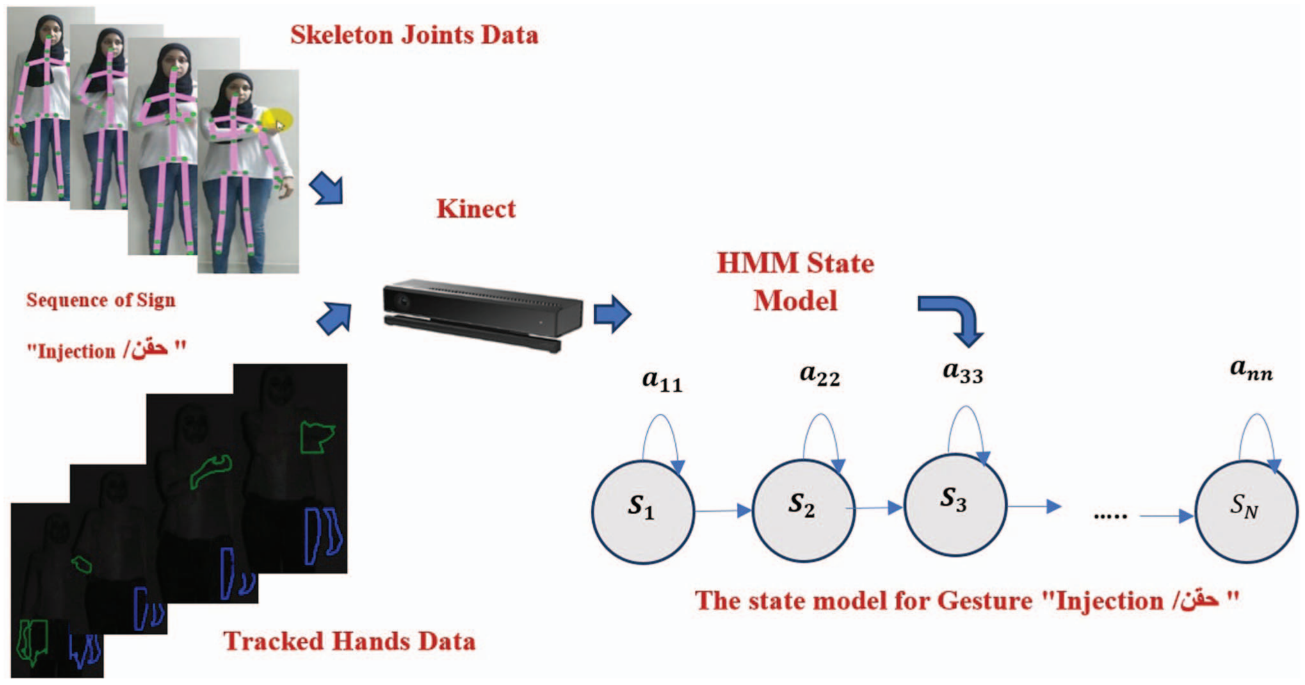


Fig. 14. HMM state model for gesture.

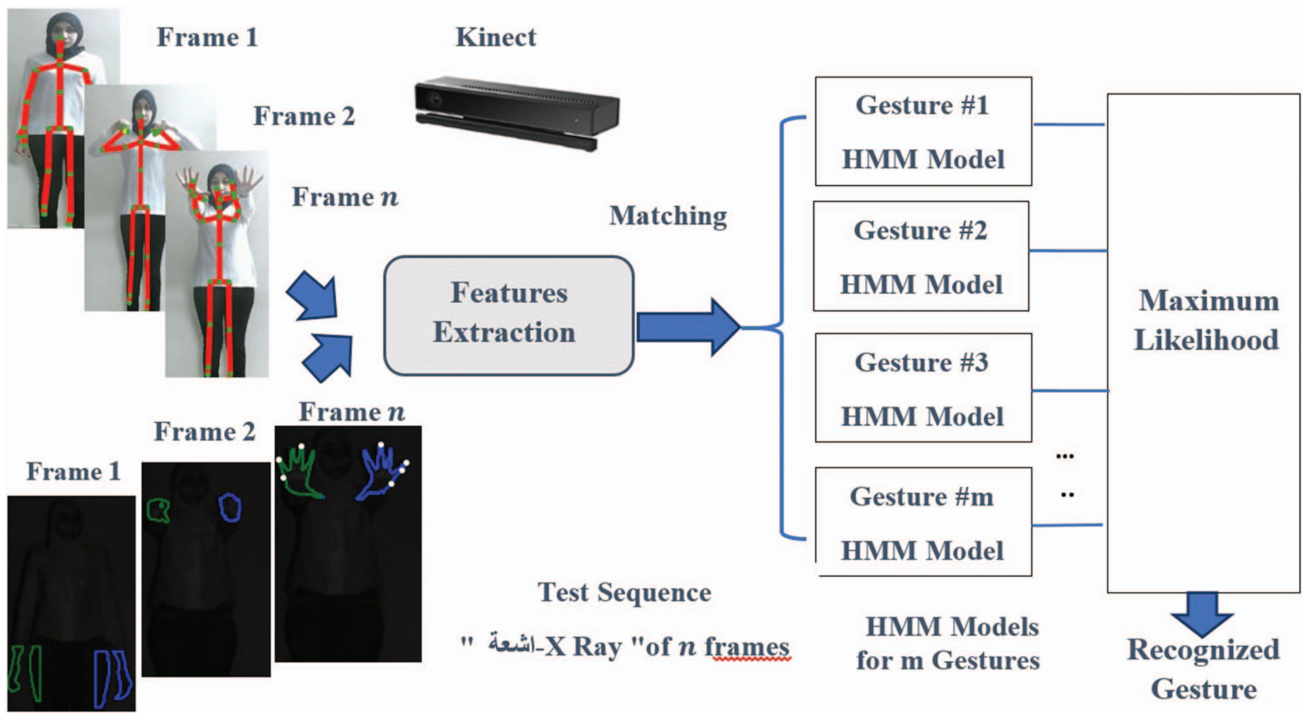


Fig. 15. HMM model for dynamic gesture recognition.

camera, an IR depth sensor, and a microphone array [42]. It is used to acquire signs and obtain the depth streams with a rate of 30 frames per second. We connected Kinect with a laptop that has a 64-bit architecture, Windows 8 operating system, 8 GB of physical memory, and Intel Core i7-5500U and 2.40 GHz with x-64-based processor. The proposed model is developed using Microsoft C# program and Microsoft Kinect SDK library.

A. Arabic Sign Dataset

We chose 40 different gestures in the medical field. They are listed in Table XVIII. We collected the data from three different volunteers in different positions and with different sizes.

B. Proposed Model Accuracy

For each sign, we collected 30 samples from three different signers and divided them into 20 for the training set and 10 for the testing set. The total samples for all signs were 1200 (800 for the training set and 400 for the testing set). The collected signs were dynamic; i.e., the sign was performed by moving body joints such as wrist, elbow, shoulder, and hands. Each sign's stream contained on average 120–200 frames, so the total number of frames was around 40,000 for the training set and 32,500 for the testing set. The feature vector was formed from the skeleton joint features and hand features that were combined to form 62 features. We applied two classifiers (SVM and KNN) in the classification phase. They were applied on the separated frames, and the accuracies were 79% and 66%, respectively. Because the selected signs

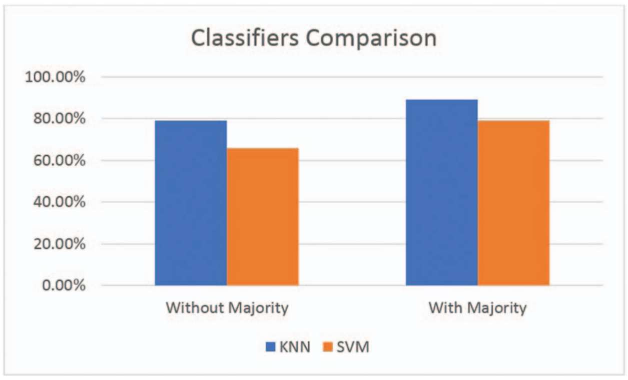


Fig. 16. Classifiers' accuracy comparison.

are dynamic in nature, we can apply the majority voting on the classified frames for each sign in order to get the accuracy of recognizing each sign. The KNN and SVM were improved after majority voting to 89% and 79%, respectively. After applying the DSmT fusion of the two classifier results, the accuracy reached 91.5%. Also, the accuracy of Ada-Boosting reached 90.2%. For the dynamic pattern recognition approach, the accuracies of DTW and HMM reached 82.6% and 79.5%, respectively. Fig. 16 presents the classifier accuracy before and after applying the majority voting for each classifier without applying the fusion. Fig. 17 presents the comparison between the accuracy of each classifier individually and after fusing their results.

It is noticed in Fig. 16 that the DSmT fusion of classifiers improves the model recognition accuracy compared to the individual classifiers and the Ada-Boosting technique. The misclassified signs using SVM reached 21% and using KNN reached 11%, while there were no

TABLE XVIII
Medical Dataset

Index	Arabic Sign	Meaning in English	Index	Arabic Sign	Meaning in English
1	اسهال	Dysentery	21	نزيف	Bleeding
2	اشعة	X-Ray	22	وفاة	Death
3	استقبال	Reception	23	طبيب عظام	Orthopedic doctor
4	رئتان	Two lungs	24	علاج طبيعى	Physical therapy
5	كبد	liver	25	حقن	Injection
6	كلى	kidneys	26	زغللة	Blurred vision
7	معدة	stomach	27	سرطان	Cancer
8	امساك	Constipation	28	جهاز قياس ضغط	Pressure measuring device
9	تحليل	analysis	29	صداع	A headache
10	تطعيم	Vaccination	30	صمم	Deafness
11	شلل	Paralysis	31	طبيب اطفال	Pediatrician
12	طبيب توليد	Obstetrician	32	طبيب انف واذن	Doctor of nose and ear
13	تقيوء	Vomiting	33	طبيب باطنة	Internist
14	تورم	Swelling	34	طبيب عام	General Doctor
15	جرح	Wound	35	كسر عظام	Broken bones
16	حامل	Pregnant	36	فيتامينات	Vitamins
17	حرارة	Fever	37	فشل كلوى	Kidney failure
18	اوردة	Veins	38	قرحة	ulcer
19	حساسية	Allergic	39	قولون	The colon
20	مغص	Colic	40	مختبر	laboratory

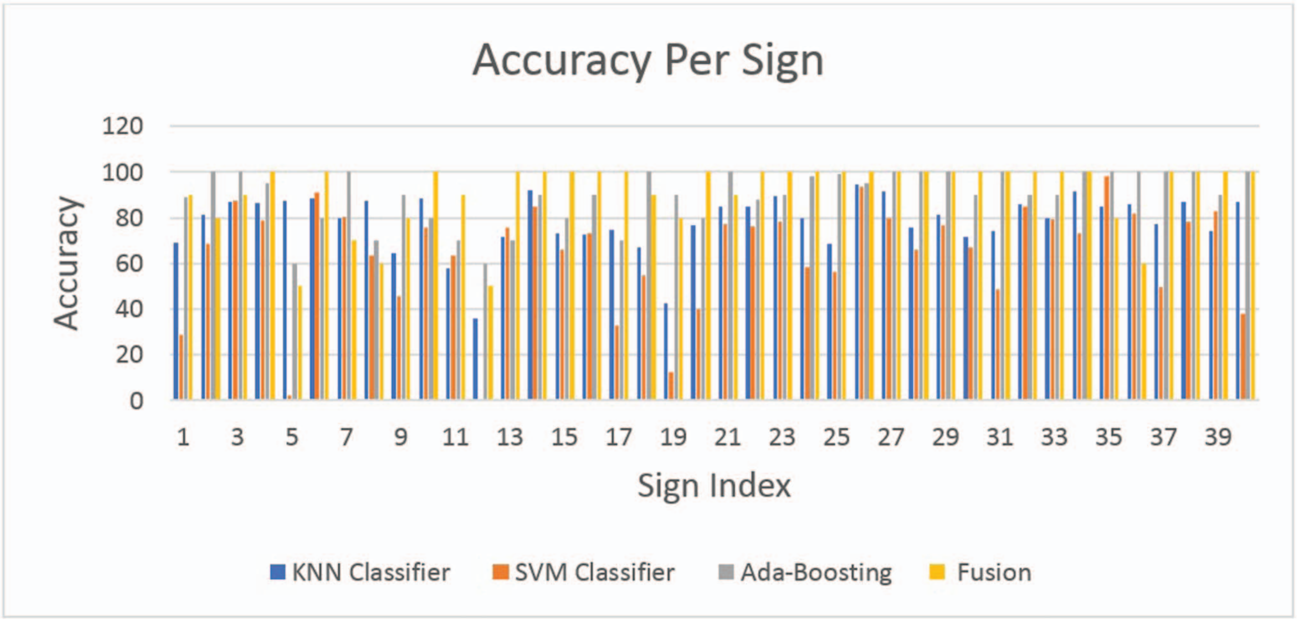


Fig. 17. Accuracy per sign comparison for classifiers versus Ada-Boosting and fusion model.

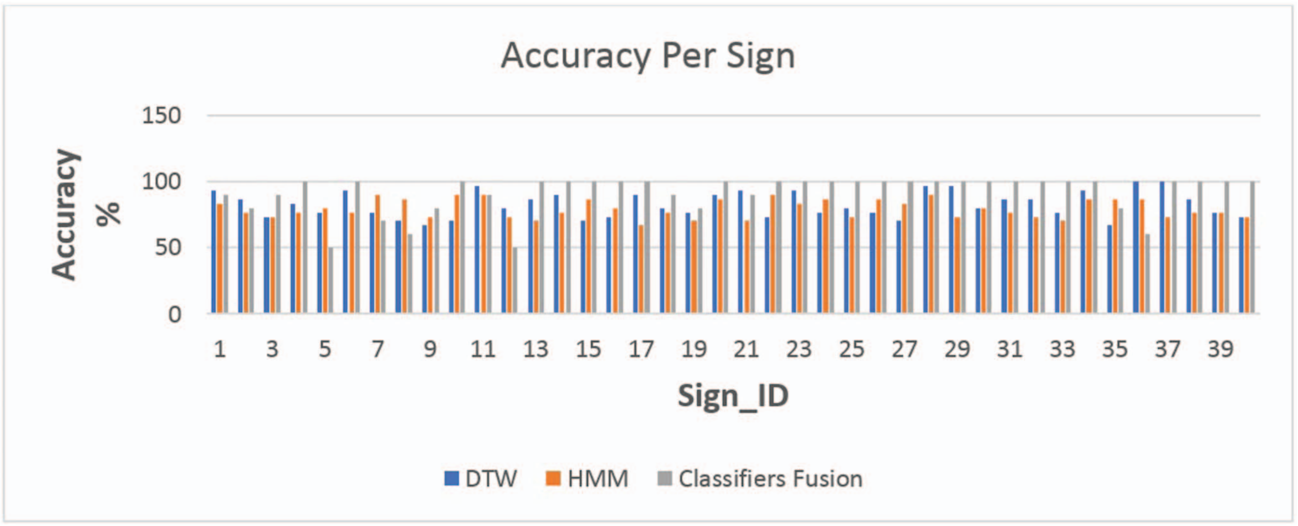


Fig. 18. Accuracy per sign comparison for DTW, HMM, and fusion model.

TABLE XIX
SVM Misclassification

Misclassified signs	Misclassification percentage
1	72%
5	98%
12	100%
17	67%
19	88%
31	51%
40	62%

TABLE XXI
Computation Latency

Process	Time (s)
Sign capturing	6
Preprocessing (data normalization)	2
KNN classifier	8
SVM classifier	5
Late fusion + DSmT fusion	7
Total	28

misclassified signs after using DSmT. Table XIX presents the misclassified signs using only SVM, and Table XX presents the misclassified signs using only KNN.

For DTW and HMM, Fig. 18 presents the comparison between the achieved accuracy per sign using DTW, HMM, and classifier fusion.

From Fig. 17, the DSmT model is more accurate than both DTW and HMM and achieved higher recognition accuracy over the 40 signs.

The system performance is a very important metric, especially when the system works in real time, so the computation latency was computed. We took into consideration that the main processes in the system are performed sequentially and also the frames of Kinect are captured at the rate of 30 frames per second. Table XXI lists the time in seconds as the average time for each process, which was calculated over 30 experiments while performing the selected signs from the dataset. Finally, the total time will be the result of aggregating the times of all processes.

TABLE XX
KNN Misclassification

Misclassified signs	Misclassification percentage
12	64%
19	58%

Also, from the experiments we found that DTW is faster than HMM. Over 30 experiments, DTW takes 5 s on average for recognition and HMM takes 7 s on average. Fig. 19 presents the processing times for both DTW and HMM for X-ray/“اشعة” over ten samples.

V. CONCLUSION AND FUTURE WORK

In this paper, we introduced an automatic system for Arabic sign recognition using Microsoft Kinect V2. The proposed model was applied and tested on 40 Arabic signs that are related to the medical field. Each sign is captured and represented as a depth stream. This stream was analyzed and normalized to overcome the variation of signer’s position and size, and then the features of both skeleton (32 features) and hand (30 features) were extracted and integrated in one feature vector with 62 features. The data with these features were used to train the two classifiers (KNN and SVM). Finally, DSmT was used to combine the results of these two classifiers. Three different signers performed the signs in order to build the required dataset. The number of collected samples was 1200 (800 for the training set and 400 for the testing set). The accuracy of the classifiers was 89% for KNN and 79% for SVM. Classifiers’ accuracy was compared with the fusion results, which reached 91.5%. Finally, we compared the fusion model with the Ada-Boosting technique, which achieved an accuracy of 90.2%, and other

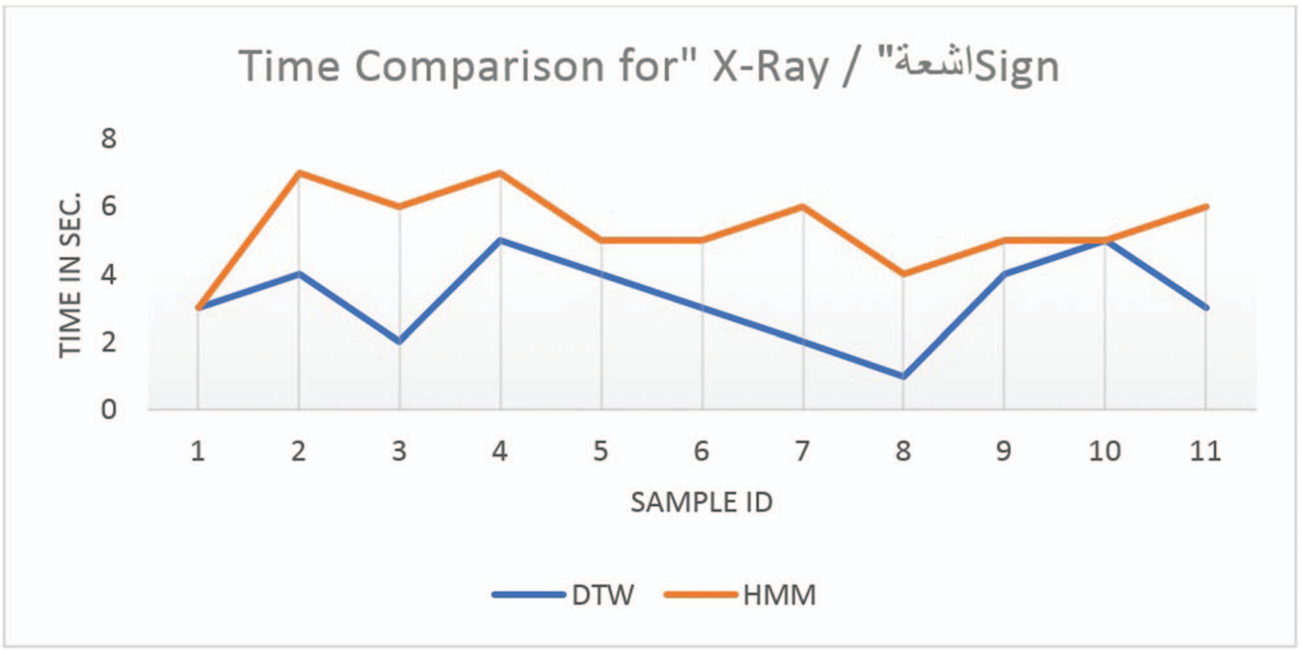


Fig. 19. DTW and HMM processing time comparison.

two algorithms that were widely used in dynamic gesture recognition. These algorithms were DTW and HMM, and they achieved an accuracy of 82.6% and 79.5%, respectively, so our model was more accurate than them. The suggested future work consists of increasing the overall accuracy of the system, improving the model in order to recognize the full sentences, and reducing the computation latency in real time. Also, we suggested the use of deep neural networks to improve the accuracy. Deep learning-based systems can learn efficiently from raw images or video sequences, so it is beneficial to process multimodality data such as the RGB-D data, skeleton, finger points, etc. that can provide rich and wide information of signers' movements. Finally, with the revolution in Internet of Things (IoT), we suggest using IoT devices like wireless wearable devices that will help in monitoring hearing-impaired people to interpret their signs and know their needs without delays.

ACKNOWLEDGMENT

This research was supported and revised by my supervisors Dr. Alaa Hamouda and Prof. Ali Rashed. The authors thank the reviewers for their effort in reviewing and for their insights and comments.

REFERENCES

[1] R. Hoopers *et al.* "Analyzing variation in sign languages: Theoretical and methodological issues," in *Signed Languages:* Discoveries From International Research, M. Metzger, S. Taub, A. M. Baer, and V. Dively Eds., Washington, DC: Gallaudet University Press, 2001, pp. 135–162.

[2] M. Mohandes, S. Aliyu, and M. Deriche "Prototype Arabic sign language recognition using multi-sensor data fusion of two Leap Motion controllers," in *Proc. 12th Int. Multi-Conf. Syst., Signals Devices*, Piscataway, NJ: IEEE, 2015.

[3] T. Dutta "Evaluation of the Kinect^{TM} sensor for 3-D kinematic measurement in the workplace," *Appl. Ergonom.*, vol. 43, no. 4, pp. 645–649, 2012.

[4] Z. Cai, J. Han, L. Liu, and L. Shao "RGB-D datasets using Microsoft Kinect or similar sensors: A survey," *Multimedia Tools Appl.*, vol. 76, no. 3, pp. 4313–4355, 2017.

[5] C.-H. Chuan, E. Regina, and C. Guardino "American sign language recognition using Leap Motion sensor," in *Proc. 13th Int. Conf. Mach. Learn. Appl.*, Piscataway, NJ: IEEE, 2014.

[6] A. Clark and D. Moodley "A system for a hand gesture-manipulated virtual reality environment," in *Proc. Annu. Conf. South Afr. Inst. Comput. Scientists Inf. Technologists*, New York: ACM, 2016.

[7] D. N. Metaxas, B. Liu, F. Yang, P. Yang, N. Michael, and C. Neidle "Recognition of nonmanual markers in American sign language (ASL) using non-parametric adaptive 2D–3D face tracking," in *Proc. Int. Conf. Lang. Resour. Eval.*, 2012.

[8] M. A. Abdel-Fattah "Arabic sign language: A perspective," *J. Deaf Stud. Deaf Educ.*, vol. 10, no. 2, pp. 212–221, 2005.

[9] I. El Naqa, R. Li, and M. J. Murphy *Machine Learning in Radiation Oncology: Theory and Applications*. Cham, Switzerland: Springer, 2015, pp. 57–70.

[10] M. A. Almasre and H. Al-Nuaim "Comparison of four SVM classifiers used with depth sensors to recognize Arabic sign language words," *Computers*, vol. 6, no. 2, p. 20, 2017.

[11] Y. Bengio, A. Courville, and G. E. Hinton
"Representation learning: A review and new perspectives,"
IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[12] M. Mohandes, M. Deriche, and J. Liu
"Image-based and sensor-based approaches to Arabic sign language recognition,"
IEEE Trans. Human–Mach. Syst., vol. 44, no. 4, pp. 551–557, Aug. 2014.

[13] H. Ahmed, S. O. Gilani, M. Jamil, Y. Ayaz, and S. I. A. Shah
"Monocular vision-based signer-independent Pakistani sign language recognition system using supervised learning,"
Indian J. Sci. Technol., vol. 9, no. 25, pp. 1–16, 2016.

[14] M. Maraqa and R. Abu-Zaiter
"Recognition of Arabic sign language (ArSL) using recurrent neural networks,"
in *Proc. 1st Int. Conf. Appl. Digit. Inf. Web Technol.*, Piscataway, NJ: IEEE, 2008.

[15] K. Assaleh and M. Al-Rousan
"Recognition of Arabic sign language alphabet using polynomial classifiers,"
EURASIP J. Adv. Signal Process., vol. 2005, no. 13, p. 507614, 2005.

[16] O. Al-Jarrah and F. A. Al-Omari
"Improving gesture recognition in the Arabic sign language using texture analysis,"
Appl. Artif. Intell., vol. 21, no. 1, pp. 11–33, 2007.

[17] N. El-Bendary, H. M. Zawbaa, M. S. Daoud, A. E. Hassanien, and K. Nakamatsu
"ArSLAT: Arabic sign language alphabets translator,"
in *Proc. Int. Conf. Comput. Inf. Syst. Ind. Manage. Appl.*, Piscataway, NJ: IEEE, 2010.

[18] K. Assaleh, T. Shanableh, and M. Zourob
"Low complexity classification system for glove-based Arabic sign language recognition,"
in *Proc. 19th Int. Conf. Neural Inf. Process.*, 2012, pp. 262–268.

[19] M. A. Mohandes
"Recognition of two-handed Arabic signs using the CyberGlove,"
Arabian J. Sci. Eng., vol. 38, no. 3, pp. 669–677, 2013.

[20] M. I. Sadek, M. N. Mikhael, and H. A. Mansour
"A new approach for designing a smart glove for Arabic sign language recognition system based on the statistical analysis of the sign language,"
in *Proc. 34th Nat. Radio Sci. Conf.*, Piscataway, NJ: IEEE, 2017.

[21] E. E. Hemayed and A. S. Hassanien
"Edge-based recognizer for Arabic sign language alphabet (ArS2V-Arabic sign to voice),"
in *Proc. Int. Comput. Eng. Conf.*, Piscataway, NJ: IEEE, 2010

[22] M. A. Almasre and H. A. Al-Nuaim
"Using the Hausdorff algorithm to enhance Kinect's recognition of Arabic sign language gestures,"
Int. J. Comput. Sci. Secur., vol. 7, no. 1, p. 2, 2017.

[23] M. ElBadawy, A. S. Elons, H. Sheded, and M. F. Tolba
"A proposed hybrid sensor architecture for Arabic sign language recognition,"
in *Intelligent Systems' 2014*. Cham, Switzerland: Springer, 2015, pp. 721–730.

[24] S. Aliyu, M. Mohandes, M. Deriche, and S. Badran
"Arabic sign language recognition using the Microsoft Kinect,"
in *Proc. 13th Int. Multi-Conf. Syst., Signals Devices*, Piscataway, NJ: IEEE, 2016.

[25] A. B. Jmaa, W. Mahdi, Y. B. Jemaa, and A. B. Hamadou
"Arabic sign language recognition based on HOG descriptor,"
in *Proc. 8th Int. Conf. Graph. Image Process.*, vol. 10225. Bellingham, WA: International Society for Optics and Photonics, 2017.

[26] M. Mohandes, M. A. Deriche, and S. O. Aliyu
"Arabic sign language recognition using multi-sensor data fusion,"
U.S. Patent 9 672 418, Jun. 6, 2017.

[27] O. Rashid, A. Al-Hamadi, and B. Michaelis
"A framework for the integration of gesture and posture recognition using HMM and SVM,"
in *Proc. IEEE Int. Conf. Intell. Comput. Intell. Syst.*, vol. 4. Piscataway, NJ: IEEE, 2009.

[28] Y. Song, Y. Gu, P. Wang, Y. Liu, and A. Li
"A Kinect based gesture recognition algorithm using GMM and HMM,"
in *Proc. 6th Int. Conf. Biomed. Eng. Inform.*, Piscataway, NJ: IEEE, 2013.

[29] P. V. V. Kishore and P. Rajesh Kumar
"A video based Indian sign language recognition system (INSLR) using wavelet transform and fuzzy logic,"
Int. J. Eng. Technol., vol. 4, no. 5, p. 537, 2012.

[30] B. Penelle and O. Debeir
"Multi-sensor data fusion for hand tracking using Kinect and Leap Motion,"
in *Proc. Virtual Reality Int. Conf.*, New York: ACM, 2014.

[31] G. Marin, F. Dominio, and P. Zanuttigh
"Hand gesture recognition with jointly calibrated Leap Motion and depth sensor,"
Multimedia Tools Appl., vol. 75, no. 22, pp. 14991–15015, 2016.

[32] K. Y. Fok, C. T. Cheng, and N. Ganganath
"Live demonstration: A HMM-based real-time sign language recognition system with multiple depth sensors,"
in *Proc. IEEE Int. Symp. Circuits Syst.*, Piscataway, NJ: IEEE, 2015.

[33] X. Yang, X. Chen, X. Cao, S. Wei, and X. Zhang
"Chinese sign language recognition based on an optimized tree-structure framework,"
IEEE J. Biomed. Health Inform., vol. 21, no. 4, pp. 994–1004, 2017.

[34] P. Kumar, H. Gauba, P. P. Roy, and D. P. Dogra
"Coupled HMM-based multi-sensor data fusion for sign language recognition,"
Pattern Recognit. Lett., vol. 86, pp. 1–8, 2017.

[35] Y. Sun *et al.*
"Gesture recognition based on Kinect and sEMG signal fusion,"
Mobile Netw. Appl., vol. 23, no. 4, pp. 797–805, 2018.

[36] M. Cheriet, N. Kharma, C. L. Liu, and C. Suen
Character Recognition Systems: A Guide for Students and Practitioners. Hoboken, NJ: Wiley, 2007.

[37] A. Lekova, D. Ryan, and R. Davidrajuh
"Fingers and gesture recognition with Kinect v2 sensor,"
Inf. Technol. Control, vol. 14, no. 3, pp. 24–30, 2016.

[38] F. Smarandache and J. Dezert
Advances and Applications of DSmT for Information Fusion, Collected Works, vol. 3. Rehoboth, DE: American Research Press, 2009, 760 pp.

[39] H. Lifang, G. Xin, and H. You
"Efficient combination rule of Dezert–Smarandache theory,"
J. Syst. Eng. Electron., vol. 19, no. 6, pp. 1139–1144, 2008.

[40] M. A. Bautista *et al.*
"Probability-based dynamic time warping for gesture recognition on RGB-D data,"
in *Advances in Depth Image Analysis and Applications*. Berlin, Germany: Springer, 2013, pp. 126–135.

[41] S. Mitra and T. Acharya
"Gesture recognition: A survey,"
IEEE Trans. Syst., Man, Cybern., Part C: Appl. Rev., vol. 37, no. 3, pp. 311–324, 2007.

[42] E. Gani and A. Kika
"Albanian sign language (AlbSL) number recognition from both hand's gestures acquired by Kinect sensors,"
Int. J. Adv. Comput. Sci. Appl., vol. 7, no. 7, pp. 777–780, 2016.

Basma Hisham received the B.Sc. and M.Sc. degrees from the Faculty of Engineering, Al-Azhar University, Cairo, Egypt, in 2012 and 2015, respectively. He is currently a Teacher Assistant with the Faculty of Engineering, Al-Azhar University, and previously was a Research Assistant with the National Telecommunication Institute.

Alaa Hamouda received the Ph.D. degree in computer engineering from Al-Azhar University, Cairo, Egypt. He is currently an Associate Professor with the Faculty of Engineering, Al-Azhar University. His research interests include data mining, text summarization, opinion mining, multiagent systems, Capability Maturity Model Integration (CMMI), swarm intelligence, and sensory network.