A Gesture Recogintion Architecture for Arabic Sign Language Communication System

TAMÁS AUJESZKY, New York University Abu Dhabi
MOHAMAD EID, New York University Abu Dhabi

Sign language is the most natural and expressive way for the hearing impaired to communicate. With technological advances in multimedia systems and applications, technology-mediated sign language communication systems have long attracted researchers to enhance the communication capabilities for the speech and hearing impaired, promising improved social opportunities and integration. This paper introduces a framework for Arabic sign language communication using Microsoft Kinect device. The merit of the proposed framework is twofold: first, the framework supports an affordable and easily deployable real-time communication system using Arabic sign language, and secondly, it provides a real-time feedback about the signer performance via real-time avatar animation. A prototype application is developed to demonstrate the merits of the proposed framework. Experimental results show that the proposed Arabic sign language method enjoys a sign detection rate of 96%. Furthermore, the average task completion time to complete an Arabic sign was about 2.2 seconds. This implies that the proposed method can be used to create a real-time Arabic sign language communication system. Finally, participants of the study highlighted that the proposed system is user-friendly and easy to use, and can be used at low cost to recognize and display Arabic signs.

Categories and Subject Descriptors: **I.2.1 [Artificial Intelligence]:** Applications and Expert Systems.

General Terms: Natural language interfaces, parameter learning, handicapped persons/special needs.

Additional Key Words and Phrases: Multimedia Systems and Tools, Sign language Communication Systems, Gesture recognition, Avatar animation, Usability study.

ACM Reference Format:

Tamás Aujeszky and Mohamad Eid, 2015. A Gesture Recogintion Architecture for Arabic Sign Language Communication. Multimedia Tools and Applications xx, xx, Article 39, 18 pages.

1. INTRODUCTION

Hearing-impairment is a form of disability that affects more than 120 million people in the world [1]. To these people, communication is a substantially harder task than to people not suffering from its consequences. Two burdens they often face are lack of convenient means of real-time digital sign language based communication,

This work is supported by the National Science Foundation, under grant CNS-0435060, grant CCR-0325197 and grant EN-CS-0329609.

Author's addresses: Tamás Aujeszky, Division of Engineering, NYUAD, United Arab Emirates, email: ta727@nyu.edu; Mohamad Eid, Division of Engineering, NYUAD, United Arab Emirates, email: mohamad.eid@nyu.edu.

DOI:http://dx.doi.org/10.1145/0000000.0000000

and low level of knowledge of sign language among people without hearing disabilities. This is particularly true in the Arabic world.

These are the problems that we are seeking to address. We are aiming to achieve this by designing, building, and evaluating a real-time sign language communication system. Since more than 86% percent of hearing-impaired people come from low and middle income countries [1], it is also our aim to make this system affordable, to maximize the potential reach of the system. The solution is a system comprising of a relatively cheap piece of hardware sensor available on the worldwide market and software designed to implement the desired functionality.

Due to the lack of Arabic sign language communication systems, in addition to concerns about variations (dialects) of Arabic sign language across the Arab world, we are directing the focus of this project to the Arabic sign language. The Arabic sign language (ArSL) possesses most of the general properties of sign languages around the world. However its documentation is in a relatively early stage. As most of the other sign languages, ArSL also has multiple country variants and dialects. The difficulty of standardization is due to the relatively low level of awareness, and this is another reason why this project would gain significance.

Arabic sign language (ArSL) has more than 9000 gestures and uses 26 static hand gestures and 5 dynamic gestures to represent the Arabic alphabets [2]. Existing literature on Arabic sign language recognition has focused on isolated signs and small dataset [3]. This paper focuses on real-time and continuous Arabic sign language communication and learning. A sample of common Arabic signs is shown in Figure 1.

Fig. 1. Samples of Arabic (left: Greetings, middle: Child, right: friend) [4]

2. RELATED WORK

Sign language recognition is a relatively new area of research, but it is getting consistently more popular year by year. The reason for the novelty of this topic is the quality of sensors needed to perform the process. Until very recently, such equipment were costly, and scarcely available. In this decade, however, digital cameras and personal computers became both affordable and accessible enough that the prospect of a sign language recognition system for the masses ceased to be impossible due to previous hardware-based limitations. While there continues to be room for improvement for sensors, we are at a state where continuous, real-time sign language recognition and translation systems are being developed worldwide, and approaching real-life application. Sign language recognition is now a prominent topic in the area of machine learning due to its importance and easiness to present to the public.

Up until very recently, the most favored direction for sign language recognition was image-based recognition. In this setup, a camera records frames of a signer signing while a computer processes the frames using selected classification methods to decide

about the gestures. Mohandes and colleagues [4] [5], provide a review of the state of the art in image-based recognition methods for Arabic Sign Language. The challenges listed in these articles, including the detection of signer regardless of background and detection of signing objects (hands) justify the advantage of a depth-based approach has over the purely image-based path. All publications note that for larger dictionary, improved accuracy and faster translation are necessary for real-life application. An example to such image-based processing using a so-called Support Vector Machine (SVM) classifier can be found in [6], where Quan achieves around 90% accuracy recognizing the Chinese Sign Language (CSL) equivalents of the Latin alphabet.

The appearance of the Microsoft Kinect [7] sensor on the market produced a shift in the landscape of research of sign language recognition, by containing a combination of an affordable and acceptably reliable depth sensor and an RGB camera. The research in [8] and [9] introduce Discriminative Exemplar Coding and Latent SVM, respectively, to recognize gestures corresponding to American Sign Language (ASL) based on the data feed of the Kinect. They achieve good accuracy over a vocabulary of 73 signs in both cases. Similarly, in [10] Agarwal and Thakur achieve 90% accuracy using an SVM classifier, though over a small vocabulary (the gestures corresponding to the digits 0-9 in Chinese Number Sign Language. Moreover, in [11] Memis and Albayrak couple the Kinect's RGB and depth feed to achieve 90% accuracy using 2D Discrete Cosine Transform (DCT) and the K-Nearest Neighbor (KNN) classification on a set of signs from Turkish Sign Language.

While there are a number of different sign languages among the previous publications, Arabic Sign Language has also been the subject of research in a couple of cases. In [12], the authors establish 93% accuracy for a dataset of 300 words using Hidden Markov Model (HMM) quantifier. In [13] Shanableh and Assaleh use KNN and Bayesian classifiers to contrast with HMM, yielding comparable results. The authors in [14] use HMM to achieve 94% word7 recognition and 75% sentence recognition accuracy. In [15] Tolba, Samir and Abul-Ela propose a graph matching technique to be used for continuous recognition of sentences in Arabic Sign Language. The model uses decision trees and decomposition of gestures into static postures. They achieve at least 63% accuracy when translating multi-word sentences, using an algorithm with quadratic runtime. A subsequent approach is detailed in [16], where a post processing algorithm is presented, responsible for correcting possible translation errors using a semantic-oriented approach.

Lastly, a number of publications detail how a sign language translation system could be used in education and/or communication. In [17], Sagawa and Takeuchi focus on the challenges of sign language education. They present a system that includes a virtual avatar performing gestures based on Japanese Sign Language (JSL). The prototype is received well although further refinements are needed for usability in education. In [18] Buttussi, Chittaro and Coppo propose a similar system (with a case study in Italian Sign Language – ISL) using Web3D technologies for display of avatars, which is very similar to our approach. In [19] Halawani and Zaitun introduce a communication system using avatars and speech recognition, based on Arabic Sign Language.

It can be concluded from the state-of-the-art that a substantial amount of research has been done on the various forms of sign language recognition, based on a number of different sign languages. However it is also visible that there does not seem to be any form of research pertaining to using the Kinect as a basis of a system responsible for recognition of signs from Arabic Sign Language. This paper is an effort to present ArSL, a system that enables real-time communication and learning of Arabic sign language. The system can also be used as a learning tool for hearing people to learn about Arabic sign language.

3. FRAMEWORK FOR ARABIC SIGN LANGUAGE COMMUNICATION SYSTEM

The Framework for Arabic Sign Language Communication System is a proposed solution that acts as an affordable sign language translator and a sign language based communications system, while maintaining high accuracy and having extremely high commercial accessibility among its advantages as well. The system comprises of hardware and software part as well as the network it is attached to. A prototype has been developed using the Microsoft Kinect device and a Unity Game Engine [20] sample as its foundation. The following sections describe the system architecture and the relevant components of the solution.

3.1 ArSL System Architecture

Figure 2 shows an outline of the system. The system architecture is composed of three components: Hardware, Software, and the network. The hardware component includes a gesture input device as well as a visual display. The software component comprises the sign identification center, the sign media center, and the sign language data storage repository. The network component is in charge of communication Arabic sign language data over a computer network.

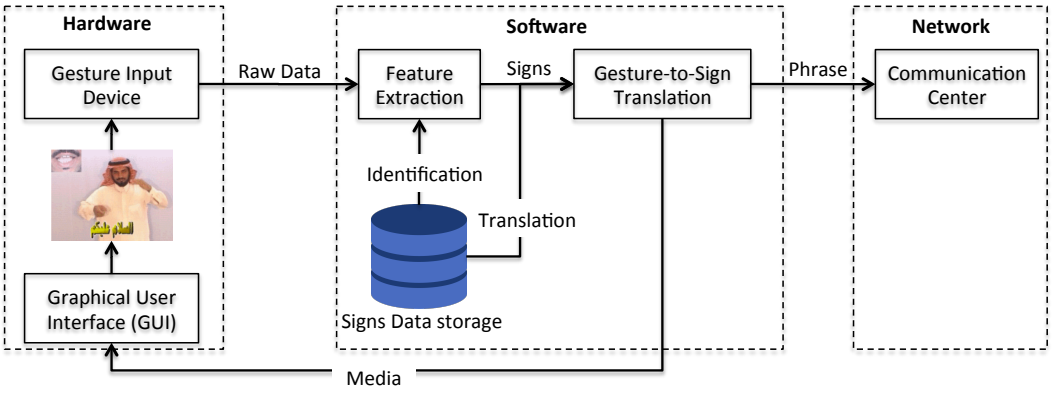

Fig. 2. Diagram for the ArSL System Architecture

3.1.1 Hardware

The hardware parts of the system are responsible for providing the input and output through which the user can interact with the system. The Gesture input device is a 1st generation Microsoft Kinect that performs the data acquisition and sends it to the Sign identification center. The Display devices (sound system and computer display) are responsible for displaying the output information arriving from the Sign media center. In its current form the system supports visual information only.

3.1.2 Software

The software part of the system is responsible for executing feature extraction and gesture translation, as well as providing a clean Graphical User Interface (GUI). The Sign identification center converts the raw data to a set of known signs depending on the existing and implemented vocabulary. It gets information about signs from the Sign language data storage. The Sign media center converts the signs acquired from the Sign identification center to the requested media and language, which it then transfers to the Display devices or the Communication center. It takes translation data from the Sign language data storage. The Sign language data storage holds both the dictionary of the sign languages as well as translation dictionaries from one language to another. Due to the prototype stage of the project, the size of the data storage is limited.

3.1.3 Network

The Communication center receives media from the Sign media center and sends it through the network to its intended destination. This part is currently not implemented, as it does not directly concern the usability testing.

3.1.4 System Flowchart

A flowchart that broadly depicts the way the system works is shown in Figure 3. The skeleton data stream is used to map user's movements to virtual model joints, which are used to update joints positions accordingly. Meanwhile, the skeleton data stream is utilized to iterate through a range of atomic gestures. A proper sequence of multiple atomic gestures leads to a recognition of a specific Arabic sign. These signs will be translated to text and be displayed via the User Interface. Meanwhile, the video stream (RGB stream) is retrieved from the Kinect sensor directly and displayed on the User Interface.

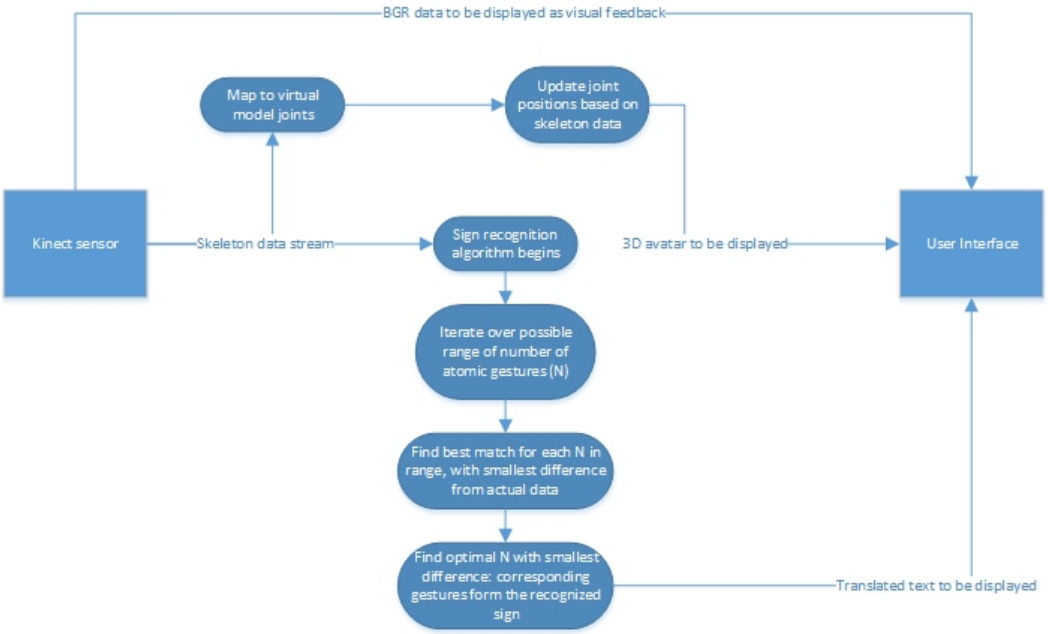

Fig. 3. System Flowchart Diagram

3.2 Feature Extraction

The first essential software building block of the system is feature extraction. The Microsoft Kinect device provides three streams of data as input to the software, and these data are used by the system to extract the moves of the signer. The first stream is the RGB feed which equates to a 24-bit True color image with 640×480 (VGA) resolution. This stream is used to give a visual feedback to the user as part of the Graphical User Interface.

The second stream contains the depth (D) data. This data is obtained using an infrared projector and an infrared camera, both installed on the Kinect device. The projector projects a previously defined pattern of dots into the signing space, and the camera tracks the arrangement in which these are reflected. Contrasting the acquired infrared image with the original one gives results in a measurement of depth for each dot.

The results have a sensitivity of 11-bit (2048 levels), while the distance in which the Kinect is able to perform depth measurement is generally between 1m and 4m. The resolution of the depth image is also 640 × 480 (VGA). It has to be noted that interfering with the projected infrared image also affects the result. As an example, it is advised to only use the Kinect indoors, where the infrared radiation of the Sun does not blind the infrared camera. Consequently, the proposed setup is developed for indoor usage.

Finally, the third data stream provided by the Kinect is the skeleton stream, and it is the most important one for this application. The skeleton stream puts the depth data through a set of decision trees with a depth of 20 to decide which of the more than 100,000 pre-loaded positions are closest to the one that the subject has taken up. Each of these positions has a corresponding skeleton, which is a set of 20 joints with defined positions. These joints represent the some of the biological joints of the human body, such as the ankles, knees, hips, shoulders, elbows or wrists as well as the head, the spine, the hands and the feet, as visible on the left half of Figure 4.

The joint positions are accessible through the Kinect API. However, these pieces of joint information consist only of a single 3D position vector, and therefore each joint is treated as a point. This is especially regrettable in the case of the hand, where one point does not nearly describe the position and the posture of the hand as accurately as would be needed for a perfectly functional gesture translation device. However, several gestures can be implicitly recognized by the variance of the single position vector assigned to the hand with respect to time. In addition, the second generation Kinect device employs an advanced skeleton model that contains the tip of the hand and the thumb as additional joints, treating the hand as a set of 3 key points (with the original hand joint included), which allows for a great improvement in gesture recognition. What has to be emphasized is that even though the current hardware (first generation Kinect) is moderately functional, it allowed the creation of a system that (in theory) can yield near-perfect results when the hardware part is switched to the advanced version. The algorithms of the system – which make up the theoretical basis of the solution - do not change, and therefore the current hardware only poses a limitation on the current set of results, but not the eventual capability of the system.

In addition to a more extensive skeleton model employed, the second generation Kinect hardware also offers 1080p (1920x1080) resolution for the RGB and depth cameras, compared to the VGA (640x480) resolution of its original counterpart. Other notable upgrades include a 60% wider field of vision for the cameras (which allows a shorter minimal distance of 3ft compared to 6ft with the first generation device), a

maximum traceable number of users of 6 (up from 2), and the ability to determine a user's heart rate and facial expression [21].

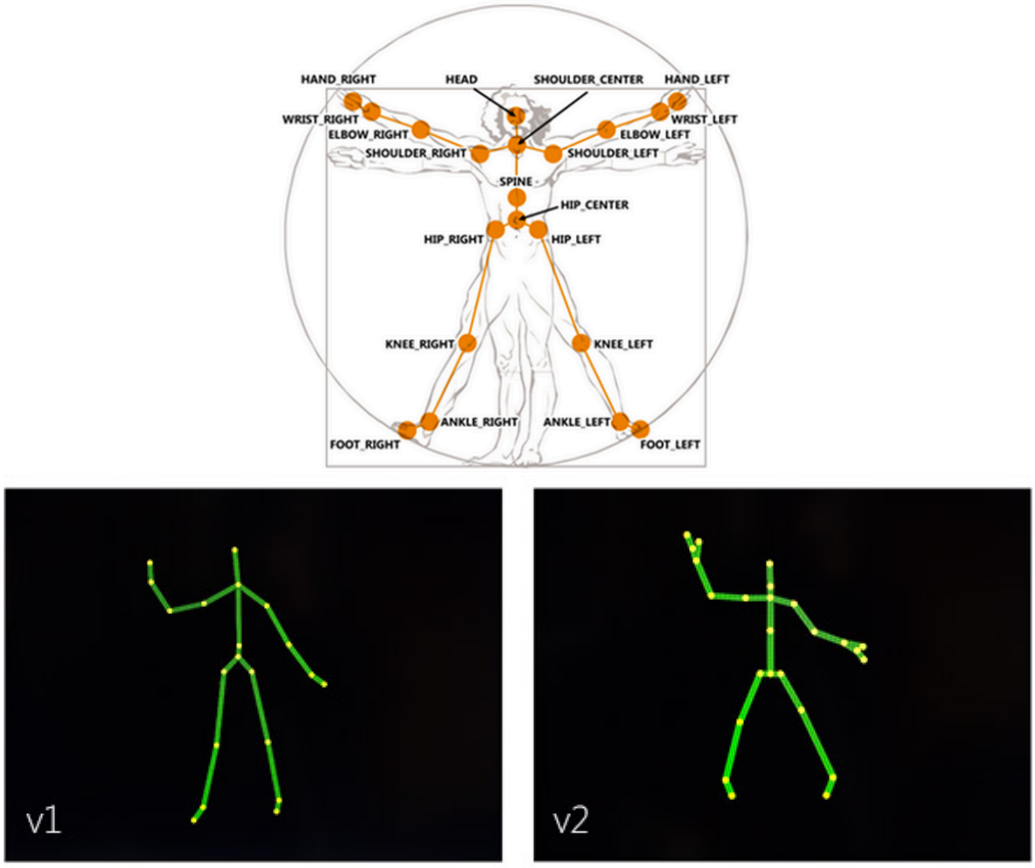

Fig. 4. Comparison between Kinect skeleton models of the first generation device (left) and the second generation device (right)

3.3 Gesture-to-Sign Translation

Gesture translation is the process of converting the data acquired from the skeleton stream into the information about which gesture is performed. This has been executed in the following fashion:

In the first step, the vocabulary of the system is defined. An online dictionary of Arabic sign language gestures was used for this step [22], and the signs had initially been selected based on how compatible they are with the limitations posed by the first generation Kinect hardware.

These signs were then broken down into sets of atomic gestures that comprise of the moving of a single joint. An atomic gesture is the combination of a handshape and its movement, and acts similarly to the phonemes and morphemes of spoken languages. A Unity-based gesture recognition demo [23] was used to experiment with gesture translation with simple pre-built atomic gesture recognition.

A summary of the sign detection algorithm is shown in Algorithms 1 to 3. Algorithm 1 implements the nearest-neighbor classification algorithm to match a set of atomic gestures representing an Arabic sign (noted as `sample_sign`) against well-defined Arabic signs (stored in the `signs` array). Note that the `normalize (sample_sign,`

n) function in Algorithm 1 performs normalization, which includes resampling, scaling with shape preservation, and translation to a reference point (ex. the origin). Algorithm 2 matches two signs by performing repeated alignments between their atomic gestures. Finally, Algorithm 3 computes the minimum-cost alignment between two signs starting from a particular atomic gesture. These algorithms are inspired by the work proposed in [25].

ALGORITHM 1. Sign recognition algorithm: Match a sample sign against a set of signs, from the Arabic sign language dictionary, by employing the Nearest-Neighbor classification rule. Returns a normalized mark in [0-1] where 1 denotes perfect match.

```
recognize (sample_sign, signs)
1: n ← N            // number of atomic gestures
2: preprocess (sample_sign, n)        // pre-process the sample sign with n gestures
3: mark ← ∞
4: for each sign in signs do
5:         preprocess (sign, n)        // each sign should be pre-processed for the n gestures
6:         d ← match (sample_sign, sign, n)
7:         if mark > d then
8:                 mark ← d
9:                 target ← sign
10: mark ← max ((2.0 – mark)/2.0, 0.0)        // normalize mark in [0-1]
11: return (target, mark)
```

ALGORITHM 2. Matching function algorithm: compare a sample sign and a sign from the Arabic sign language dictionary, by performing repeated alignments between their atomic gestures. ε controls the number of tested alignments. Returns the minimum alignment cost.

```
match (sample_sign, sign, n)
1: ε ← .50
2: step ←⌊n^(1−ε)⌋
3: min ← ∞
4: for i = 0 to n step step do
5:         d_1 ← distance (sample_sign, sign, n, i)
6:         d_2 ← distance (sign, sample_sign, n, i)
7: min ← minimum (min, d_1, d_2)
8: return (min)
```

ALGORITHM 3. Distance between two signs: Compute the minimum-cost alignment between atomic gestures of two different signs, with n atomic gestures and starting with with gesture begin. Assign decreasing confidence weights between 0 and 1 to point matching.

```
Distance (sample_sign, sign, n, begin)
1: matched ← new bool [n]
2: total ← 0
3: i ← begin        // start matching sign with sample_sign_i, starting from begin
4: do
5:         min ← ∞
6:         for each j such that not matched [j] do
7:                 d ← Euclidean-Distance (sample_sign_i, sign_j)
8:                 if d < min then
9:                         min ← d
10:                        index ← j
```

```
11:     matched [index] ← true
12:     weight ← 1 – ((i – begin + n) MOD n)/n
13:     total ← total + weight · min
14:     i ← (i + 1) MOD n
15: until i == begin          // all points are processed
16: return total
```

When the decomposition of the signs to atomic gestures was done, they were implemented into the source code as both a temporal composition of these gestures (first atomic gesture followed by the next one) and a spatial composition (when multiple atomic gestures are performed at the same time). The relevant numerical details related to relative joint position were specified manually so that the system produced a series of correct outputs when a test input was applied. This was followed by the gesture being added to the list of gestures the software is scanning for to recognize.

Finally, with each added gesture, a string corresponding to the translation is also added to the source code, to be displayed when the gesture is captured.

It is worth mentioning that some of the above practices were executed keeping in mind the limited size of the vocabulary that the prototype was aimed to have. However scaling up with respect to the vocabulary size is quite possible, adding gestures (gesture decomposition, translation text, etc.) is a process that can be automated fairly well with additional use of databases for a larger vocabulary.

3.4 Graphical User Interface

A screenshot of the Graphical User Interface (GUI) of the solution can be seen in Figure 5. It shows the program working in the gesture translation mode.

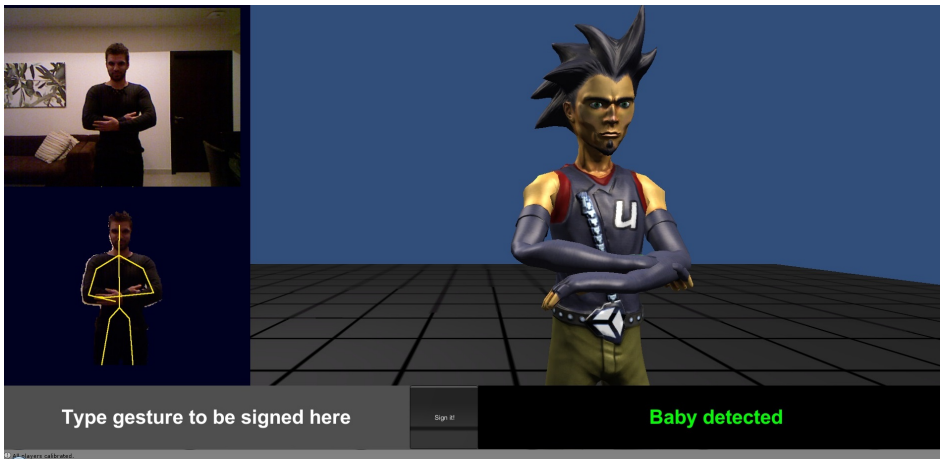

Fig. 5. Graphical User Interface (GUI) of the program in Translation Mode

As stated above, the solution uses the Unity Game Engine as its basis. It consists of a single scene in which a generic avatar replicates the signs of the user. The avatar is easy to replace with any humanoid 3D model as long as it is properly keyed with the relevant Forward Kinematics (FK) and Inverse Kinematics (IK) effectors that the Kinect's skeleton model requires. Using FK means specifying the exact motion of a joint in its own coordinate system, which also affects the underlying child joints in the hierarchy, but their relative position will remain unchanged (such as how

rotating the wrist will affect the fingers but won't bend or extend them). IK motion, however, means ordering the linear displacement of a joint with respect to the global coordinate system in the simplest way, which also affects the parent joints and their relative position (such as how pulling a finger away from the torso also results in extending the elbow). Therefore IK effectors are usually placed at the end of the hierarchy (fingers, feet) while FK effectors usually correspond to the real-life joints (elbows, shoulders, knees).

Other than the avatar and the camera through which the avatar is viewed, there are no more Unity-based objects in the software. This means everything else visible on the screen is dynamically generated. This includes the left panel of the GUI, which gives real time feedback for the signer and a bottom bar that is reserved for input and output information.

The feedback panel on the left consists of two screens. The upper screen acts as a rearview mirror for the user, presenting them with the exact image that the BGR camera of the Kinect sees. This helps users position themselves properly with respect to the field of view of the camera as well as reminding them if they perform a gesture incorrectly, and they also get a sense of reassurance that the program sees them when it is running.

The lower half of the feedback panel takes this feedback one step further for training and development purposes. The image combines the BGR feed with the information from the depth camera and only shows the part of the image where the depth is in the range of the detected user. This results in essentially stripping the background off the image and showing only the most relevant source of information – the user. The user also sees the visual representation of the skeleton stream laid over the body of the user. This provides great help for the user in figuring out in the case of a non-satisfactory output whether the problem is with his posture or the system's vision. The Baby sign pictured on Figure 5 is a great example to this need, as the system may run into difficulties when trying to infer the position of the hands while being covered by the arms.

During Education Mode (that is, when gesture demonstration is selected instead of gesture translation) the extracted user video disappears, as there is no need for the system to track the user. Instead, its place is taken up by a video / gif animation of the expression to be shown. This functionality can be seen on Figure 6. Here, each frame of the gif animation is shown for better presentation of a moving image, although the version to be used consists of a single animation instead for better usability.

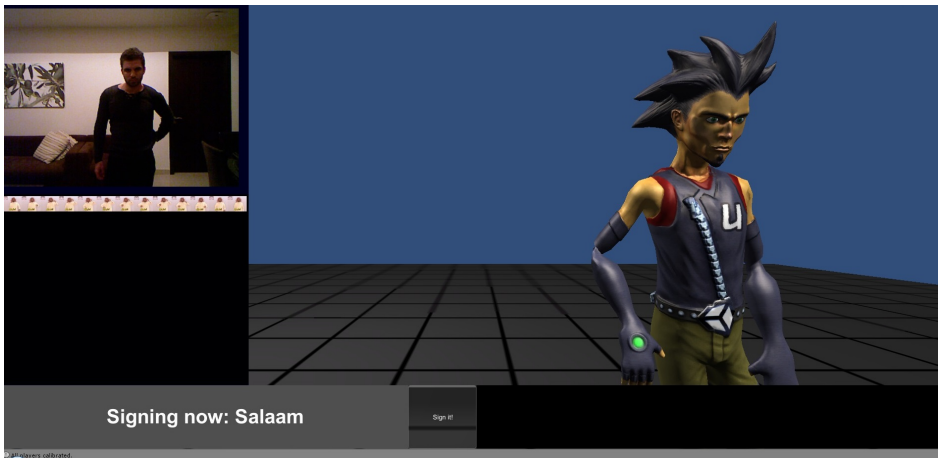

Fig. 6. GUI of the program in Education Mode (sign playback)

While the left panel of the GUI provides the visual feedback information, the bottom bar is responsible for the textual input / output functions. The bar is divided into three regions. The right region is reserved for the translation display. This is where the translation of the signs takes is indicated, and consequently this region is disabled during Education Mode.

The left region of the GUI is an input field that reads "Type gesture to be signed here" by default. This is to be overwritten with the text to be signed.

The last remaining piece of the GUI is the middle portion of the bottom I/O bar. This is where a button with the label "Sign it!" is located. This button is responsible for toggling between Translation Mode and Education Mode. The program starts in the former by default. All the user has to do in order to achieve sign playback is the following: type the expression to be signed into the input field on the left side of the bottom bar and then click the button. This will automatically invoke all the change in functionality described above that corresponds to Education Mode. Clicking the button once again, however, will restore Translation Mode.

3.5 Avatar Animation/Display

Education Mode is one of the two fundamental functionalities of the system. Eventual goals for the system regarding this mode will be to provide a representation of the sign to be gestured via the avatar in a way that the avatar moves exactly as it should in order to show how the sign should be performed. Since the avatar is deeply bound to the Kinect skeleton stream by the sample, it is a better solution to employ a second (but possibly identical) humanoid 3D model for signing purpose. This new avatar is also part of the Unity scene, but is only visible when needed – in this case Unity's camera is instructed to switch its field of view to the signer avatar away from the one used for the feedback. This switch is naturally reversed as the system goes back to Translation Mode.

For the prototype, the abovementioned secondary avatar is replaced with a gif-based animation stream on the GUI that displays the sign to be used.

4. PERFORMANCE EVALUATION

After the development of a system it is essential to gauge its efficiency from the eyes of the user, hence conducting a well thought usability test that can assist in optimizing the system to better serve its purpose. This section reports on the

experimental procedure, the experimental setup, and the analysis of the collected data.

4.1 Experimental Setup

Ten adult subjects participated in the experimental study, primarily students of age 20–23 years from New York University Abu Dhabi. Out of these subjects, 6 were male and 4 were female. The purpose of this usability testing experiment was solely to determine the performance of the system; the performance of the subjects was not being evaluated.

Although the details of the experimental procedure will be provided in the next section, it is important to highlight the variables and constants of the experiment: the system setup, location of the experiment, initial briefing, sequence of actions performed, and the number and order of trials were kept constant across all subjects and experiments conducted.

We were primarily evaluating three variables through the actions performed by the subjects: Task Completion Time (TCT), false positives and false negatives. The TCT is defined as the total time taken to perform a sign, including time it takes to process the video stream, detect the sign and provide the visual feedback through the GUI. The false positive parameter describes the number of times a gesture was detected without it being performed. The false negative parameter represents the number of times the detected gesture was wrong.

The experimental setup consisted of a first generation Kinect sensor and a Dell Precision T5610 desktop computer. The specifications of the Kinect sensor employed are BGR camera resolution (640 x 480 (VGA) @ 30 fps), Depth camera resolution (320 x 240 (QVGA)), skeleton model is 20 joints, and number of skeletons tracked is up to 2. The specification data for the Kinect sensor is acquired from [30]. The specifications of the Dell Precision T5610 desktop include an Intel Xeon E5-2609 v2, dual core, 2.5 GHz CPU, 16GB RAM, Windows 7 Professional, 64-bit operating system, and a 21”, 1920 x 1200 display [31].

4.2 Experimental Procedure

Subjects were brought to the experimental setup in order to take part in the experiment. The subject was given a short entrance briefing on the nature of the experiment. The briefing included a demonstration of each of the gestures used in the experiment, two of which are shown in Figure 7. The subject was then asked to sign a consent form and provided an entrance questionnaire to fill out.

The subject then conducted three sessions of interaction, separated by a 10-15 minutes break. Each session comprises three trials, representing common Arabic signs (such as Shukran, Salaam, Friend). Each trial is composed of six signs (as shown in Table 1). While the subject conducted the trial sets, timing data and the number of system false detections (false positive and false negative) were recorded. After conducting the three sessions, the subject was given an exit questionnaire to record their qualitative feedback regarding the system. The exit questionnaire aimed at evaluating user’s satisfaction, intuitiveness, and acceptance of the system.

Fig. 7. (Left) Subject performing the 'baby' gesture. (Right) Subject performing the 'shukran' or 'thank you' gesture.

Table 1. One session is composed of three trial sets of Arabic signs

First Trial Set	**First Trial Set**	**First Trial Set**
Shukran	Salaam	Friend
Friend	Friend	Friend
Shukran	Shukran	Shukran
Salaam	Shukran	Salaam
Salaam	Salaam	Shukran
Friend	Friend	Salaam

4.3 Results and Analysis

Following are the graphs that show the results in relation to the specified parameters. Although usability testing was conducted with only 10 subjects due to resource constraints, we may still attempt to draw some inferences from the data, though not necessarily conclusive.

Figure 8 shows the time taken for each subject for each of the trial sets. We can see that as subjects use the system their performance in conducting the trial sets increases. This shows that the system has high learnability due to its simplicity and user friendliness. The average time taken for the first trial set was 13.88 seconds, or 2.33 seconds for a given sign to be performed, processed, and detected. This time reduces to an average of 13.09 seconds for the second trial (2.18 seconds per sign) and 11.99 seconds for the third trial (1.99 seconds per sign).

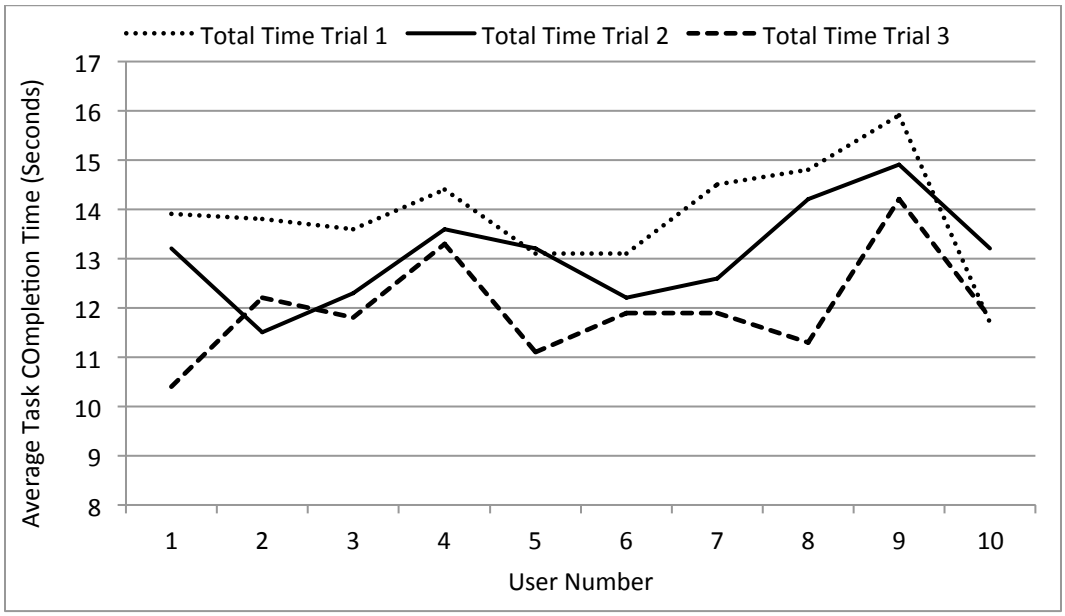

Fig. 8. The Task Completion Time for each subject for three trial sets

Figure 9 shows the distribution of the participants with respect to the average time take by them to perform a sign. It is evident that the majority of participants were performing the sign and receiving the feedback from the system within 2.05–2.20 seconds. It is important to mention at this stage that most of the participants who took longer time (around 2.2 seconds) were the ones who have never used the Microsoft Kinect™ before. Although due to the limited size of the participants it cannot be concluded that on average our system will always take less than 2.2 seconds, this deductive reasoning is worth mentioning nonetheless. More importantly, a delay of 2.2 seconds per sign would still make the system acceptable for soft real-time communication.

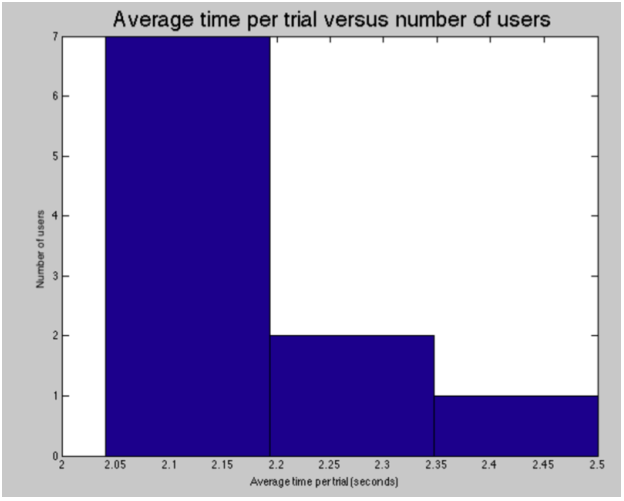

Fig. 9. Histogram showing the distribution of the average time for trial sets for each subject

Figure 10 shows the simple breakdown of the number of total false detections during the course of the experiment. This average is calculated by summing up the false detections across all sessions, trials and for all the participants. It is apparent that 50% of the participants experienced 1.0–1.5 false detections on average for the three sessions, which corresponds to a sign detection rate of 96%. Moreover, on average not more than 2 false detections were experienced by all the participants for the three sessions – corresponding to a minimum sign detection rate of 92%. We also noticed that as subjects continued to perform trials, their ability to perform the signs improved and practically the sign detection rate was all around 96% for second and third sessions.

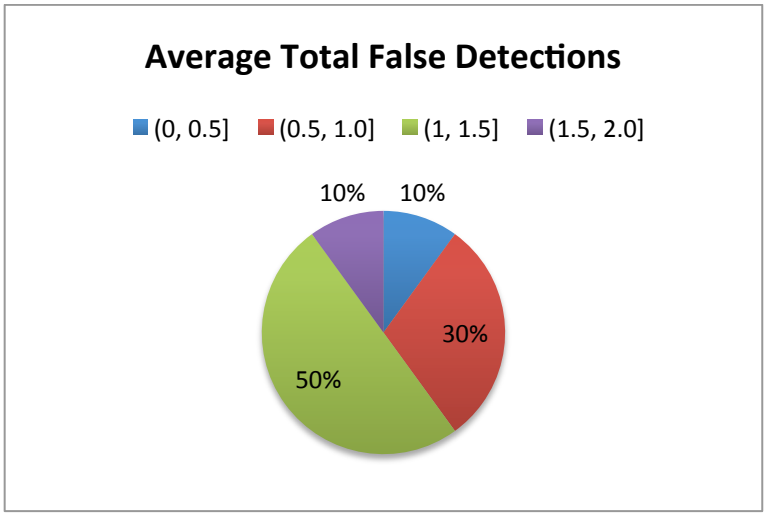

Fig. 10. Pie chart showing the distribution of the average number of total false detections for each subject.

Figure 11 and 12 show the level of correlation between some aspects of the qualitative data and the quantitative data. Figure 11 shows that there was an evident relationship between the number of false detections for a subject and his/her perception of the ease of interactivity with the system. This nature of relationship was expected even before conducting the analysis. Users perceived a false detection as a consequence of not performing the sign correctly. It is only natural for the participants to think that it is difficult to interact with the system if there were on average 2 false detections for the 3 sessions (a total of 54 signs). A similar trend was seen in figure 12 where a correlation is established between the average false detections and the level of intuitiveness.

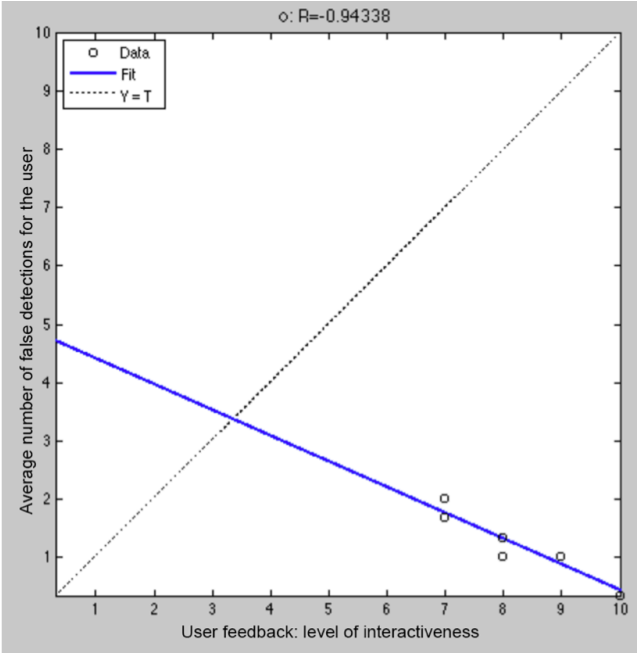

Fig. 11. Correlation of the number of false detections during the subject's trials vs the subject's perceived level of ease of interactivity with the system.

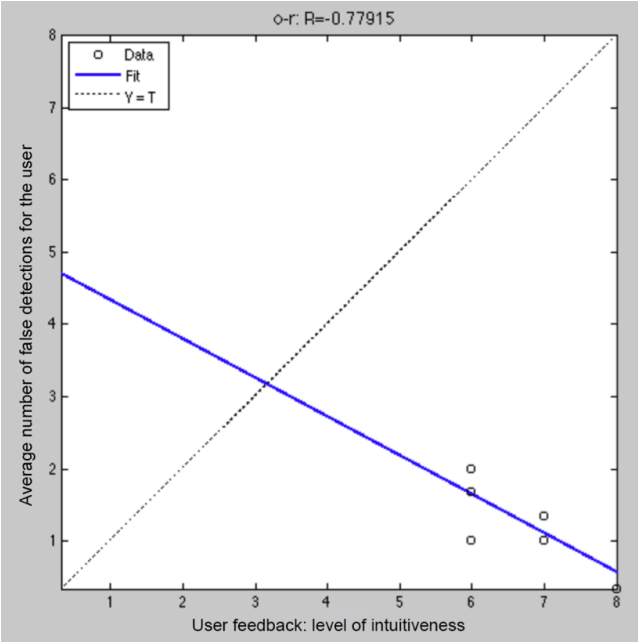

Fig. 12. Correlation of the number of false detections during the subject's trials vs. the subject's perceived level of intuitiveness of the system.

Although the usability experiment was conducted with only 10 participants, it has definitely given several key insights about the system. These insights can be used for both optimizing the system and developing a more comprehensive experimental framework. It is also worth mentioning that one of the most pleasing features of the system as perceived by the subjects was the interactive avatar. Participants felt that

it provided them with a great way to witness their own actions. Some of the participants even commented on the visual output being an efficient addition to the system as a learning tool. Overall, the performance of our system was perceived to be efficient; most of the subjects liked interacting with the system. Another question that was posed by few of the participants was to explore how our system will look as a finished product, and whether it would be possible to import their own avatars in the system.

5. DISCUSSION

From the above results we can conclude that the experiment yielded positive results. The depth-based method provided by the Kinect proved to be a reliable approach. Since the depth-sensing process takes place using an infrared projector and camera, the sensing is not susceptible to lighting conditions. However the limitation that keeps the Kinect from sensing accurately outside posed by the Sun still stands, therefore this solution is only to be used indoors.

The Kinect also fared well with regards to the real-time sign recognition constraint. This is essential to remain in effect for the final product in order to keep producing great user experience.

The error rate produced during this experiment was acceptable taking into account the fact that numerical values corresponding to relative joint positions and movements were fed manually to the system, after only a handful of test runs. Comparing our results with state-of-the-art development, we found that our system outperformed existing work (see Table 2). Note that none of the compared previous works in Table 4 have tried to measure the task completion time for detection signs. We believe this is an important quality parameter, as it would highly affect the ability of the system to act as a real-time communication system.

Table 2: Comparison with state-of-the-art development

Sign Detection Method	Accuracy	TCT (seconds)	Language
Jiang et al. (2014) [26]	92.36%	Not measured	Chinese
Verma et al. (2013) [27]	90.67%	Not measured	English
Oszust et al. (2013) [28]	89.33%	Not measured	Polish
Memis et al. (2013) [29]	94.44%	Not measured	Turkish
Our Method	96%	2.2	Arabic

However there are still a couple of factors that give rise to objectives needed to be accomplished in order to be able to build a viable product that reaches the goal of facilitating sign-language based communication. One of these factors is scaling the system up with respect to vocabulary size, since our solution contained only a handful of gestures. This challenge can be efficiently solved with building a decision tree using the decomposition of gestures into atomic gestures as mentioned earlier. The set of atomic gestures is rather bounded (American Sign Language, for example, uses 18 handshapes [24] in addition to the number of gestures for ArSL mentioned in the introduction), which allows for a real-time evaluation of signals even when using a larger dictionary. In addition, it is easy to see that the atomic decomposition and decision tree-based approach also promises reliable accuracy, since it is very similar to how sign-language is evaluated between humans (where the number of false detections had been low enough that sign-language could develop to a widely-used means of communication).

Other than building the decision tree, the system's accuracy could be evolved by adjusting the numerical values corresponding to relative joint movements during an atomic gesture through training the system. This would be especially handy since it directly affects the outcome of navigating the decision tree, which leads to the recognized gesture.

6. CONCLUSION

A system architecture, designed to perform both real-time communication between users of Arabic Sign Language and speakers of Arabic, and to serve as an educational tool for Arabic Sign Language, is proposed. The aim of the system is to be widely deployable and accessible. As such, it has to consist of components that are both affordable and generally available. The proposed system fulfills these requirements by being composed of the Kinect sensor, a PC and additional software. A functional prototype is presented, and is taken under usability experimentation, the results of which are overwhelmingly positive. The system exhibits high learnability and user friendliness, which are essential attributes for a real-life product. Furthermore, the average task completion time for completing a sign was shown to be around 2.2 seconds, this is comfortably within the range to make a real-time sign language communication system. Thus, the system appears to be a valid proof of concept, and it is a worthy target for further research.

Our immediate future work includes extending the current dictionary for Arabic sign language to cover way more than what is covered in the current implementation. Testing the system with larger set of signs would provide a better indication of the quality of the proposed system. Furthermore, we plan to move forward and implement and evaluate the real-time communication functionality in the proposed system. Finally, we plan to test the whole system again with people with deaf as they are the intended users of the system.

ACKNOWLEDGMENTS

The authors would like to thank Usama Afzal and Umair Saad for help in conducting the usability study for this work.

REFERENCES

[1] WHO (2008). "The global burden of disease": 2004 update ([Online-Ausg.] ed.). Geneva, Switzerland: World Health Organization. p. 35. ISBN 9789241563710..

[2] Nashwa El-Bendary, Hossam M. Zawbaa , Mahmoud S. Daoud, Aboul Ella Hassanien, and Kazumi Nakamatsu. "ArSLAT: Arabic Sign Language Alphabets Translator". International Journal of Computer, Information Systems and Industrial Management Applications. ISSN 2150-7988 Volume 3 (2011) pp. 498 - 506.

[3] Tolba, M.F.; Samir, A.; Abul-Ela, M., "A proposed graph matching technique for Arabic sign language continuous sentences recognition", 2012 8th International Conference on Informatics and Systems (INFOS), pp. 14 - 20, 14-16 May 2012.

[4] Mohandes, M.; Junzhao Liu; Deriche, M., "A survey of image-based Arabic sign language recognition" 2014 11th International Multi-Conference on Systems, Signals & Devices (SSD), pp.1,4, 11 - 14 Feb. 2014.

[5] Mohandes, M.; Deriche, M.; Junzhao Liu, "Image-Based and Sensor-Based Approaches to Arabic Sign Language Recognition" IEEE Transactions on Human-Machine Systems, Volume 44 (2014), Issue 4, pp.551 - 557.

[6] Yang Quan; "Chinese Sign Language Recognition Based On Video Sequence Appearance Modeling" 2010 5th IEEE Conference on Industrial Electronics and Applications (ICIEA), pp.1537 - 1542.

[7] "Kinect for Windows is now Available" Kinect for Windows Blog, http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/31/kinect-for-windows-is-now-available.aspx Accessed April 7th, 2015.

[8] Chao Sun; Tianzhu Zhang; Bing-Kun Bao; Changseng Xu, "Discriminative Exemplar Coding for Sign Language Recognition With Kinect" IEEE Transactions on Cybernetics, Volume 43 (2013), Issue 5, pp.

1418 - 1428.
[9] Chao Sun; Tianzhu Zhang; Bing-Kun Bao; Changseng Xu, "Latent support vector machine for sign language recognition with Kinect" 2013 20th IEEE International Conference on Image Processing (ICIP), pp. 4190 - 4194.
[10] Agarwal, A.; Thakur, M.K., "Sign Language Recognition using Microsoft Kinect" 2013 Sixth International Conference on Contemporary Compouting (IC3), pp. 181 - 185.
[11] Memis, A.; Albayrak, S., "Turkish Sign Language recognition using spatio-temporal features on Kinect RGB video sequences and depth maps" 2013 21st Signal Processing and Communications Applications Conference, pp. 1 - 4.
[12] Mohandes, M.; Quadri, S.I.; Deriche, M., "Arabic Sign Language Recognition an Image-based Approach" 2007 21st International Conference on Advanced Information Networking and Applications Wokshops (AINAW '07), pp. 272 - 276.
[13] Shanableh, T.; Assaleh, K., "Video-based feature extraction techniques for isolated Arabic sign language recognition" 2007 9th International Symposium on Signal Processing and Its Applications (ISSPA), pp. 1 - 4.
[14] Assaleh, K.; Shanableh, T.; Fanaswala, Mustafa et al., "Vision-based system for continuous Arabic Sign Language recognition in user dependent mode" 2008 5th International Symposium on Mechatronics and Its Applications (ISMA), pp. 1 - 5.
[15] Tolba, M.F.; Samir, A.; Abul-Ela, M., "A proposed graph matching technique for Arabic sign language continuous sentences recognition" 2012 8th International Conference on Informatics and Systems (INFOS), pp. MM-14 - MM-20.
[16] Samir A.; Aboul-Ela, M., "Error detection and correction approach for Arabic sign language recognition" 2012 Seventh International Conference on Computer Engineering & Systems (ICCES), pp. 117 – 123
[17] Sagawa H.; Takeuchi M. "A Teaching System of Japanese Sign Language Using Sign Language Recognition and Generation" 2012 Proceedings of the Tenth ACM International Conference on Multimedia, pp. 137 - 145.
[18]] Buttussi, F.; Chittaro, L.; Coppo, M., "Using Web3D Technologies for Visualization and Search of Signs in an International Sign Language Dictionary" 2007 Proceedings of the Twelfth International Conference on 3D Web Technology, pp. 61 - 70.
[19] Halawani, S.M.; Zaitun A.B., "An Avatar Based Translation System from Arabic Speech to Arabic Sign Language for Deaf People" International Journal of Information Science and Education, Volume 2, Number 1 (2012), pp. 13 – 20.
[20] "The Best Development Platform For Creating Games" Unity, http://unity3d.com/unity Accessed April 8th, 2015.
[21] "Meet Kinect" Kinect for Windows, https://www.microsoft.com/en-us/kinectforwindows/meetkinect/default.aspx Accessed April 10th, 2015.
[22] Arabic Sign language Dictionary "للصم العربي الإشاري القاموس" http://www.menasy.com/index.htm Accessed November 15th, 2014.
[23] "Kinect with MS-SDK" Unity 3d Asset Store, www.assetstore.unity3d.com/en/#!/content/7747 Accessed November 15th, 2014.
[24] Tennant and Brown; Richard A. Tennant; Marianne Gluszak Brown (1998). The American Sign Language handshape dictionary. Gallaudet University Press. p. 407. ISBN 1-56368-043-2.
[25] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. 2012. Gestures as point clouds: a $P recognizer for user interface prototypes. In Proceedings of the 14th ACM international conference on Multimodal interaction (ICMI '12). ACM, New York, NY, USA, 273-280.
[26] Yongjun Jiang; Jinxu Tao; Weiquan Ye; Wu Wang; Zhongfu Ye, "An Isolated Sign Language Recognition System Using RGB-D Sensor with Sparse Coding", 2014 IEEE 17th International Conference on Computational Science and Engineering (CSE), pp.21-26, 2014.
[27] Verma, H.V.; Aggarwal, E.; Chandra, S., "Gesture recognition using kinect for sign language translation," IEEE Second International Conference on Image Information Processing (ICIIP), pp.96-100, 2013.
[28] Oszust, M.; Wysocki, M., "Polish sign language words recognition with Kinect", The 6th International Conference on Human System Interaction (HSI), pp.219-226, 2013.
[29] Memis, A.; Albayrak, S., "Turkish Sign Language recognition using spatio-temporal features on Kinect RGB video sequences and depth maps", 21st Signal Processing and Communications Applications Conference (SIU), pp.1-4, 2013.
[30] "Quick Reference: Kinect 1 vs Kinect 2" The Imaginative Universal, http://www.imaginativeuniversal.com/blog/post/2014/03/05/Quick-Reference-Kinect-1-vs-Kinect-2.aspx Accessed May 25th, 2015.
[31] "Dell Latitude E7240 specs" Engadget, http://www.engadget.com/products/dell/latitude/-e7240/specs/ Accessed May 25th, 2015.