Multimedia Tools and Applications (2022) 81:34525–34546
https://doi.org/10.1007/s11042-021-11830-y

1174: FUTURISTIC TRENDS AND INNOVATIONS IN MULTIMEDIA SYSTEMS USING BIG DATA, IOT AND CLOUD TECHNOLOGIES (FTIMS)

An AI-based Approach for Improved Sign Language Recognition using Multiple Videos

Cameron Dignan[1] · Eliud Perez[1] · Ishfaq Ahmad[1] · Manfred Huber[1] · Addison Clark[1]

Received: 19 October 2020 / Revised: 2 August 2021 / Accepted: 21 December 2021 /
Published online: 28 February 2022

Abstract
People with hearing and speaking disabilities face significant hurdles in communication. The knowledge of sign language can help mitigate these hurdles, but most people without disabilities, including relatives, friends, and care providers, cannot understand sign language. The availability of automated tools can allow people with disabilities and those around them to communicate ubiquitously and in a variety of situations with non-signers. There are currently two main approaches for recognizing sign language gestures. The first is a hardware-based approach, involving gloves or other hardware to track hand position and determine gestures. The second is a software-based approach, where a video is taken of the hands and gestures are classified using computer vision techniques. However, some hardware, such as a phone's internal sensor or a device worn on the arm to track muscle data, is less accurate, and wearing them can be cumbersome or uncomfortable. The software-based approach, on the other hand, is dependent on the lighting conditions and on the contrast between the hands and the background. We propose a hybrid approach that takes advantage of low-cost sensory hardware and combines it with a smart sign-recognition algorithm with the goal of developing a more efficient gesture recognition system. The Myo band-based approach using the Support Vector Machine method achieves an accuracy of only 49%. The software-based approach uses the Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) methods to train the Myo-based module and achieves an accuracy of over 80% in our experiments. Our method combines the two approaches and shows the potential for improvement. Our experiments are done with a dataset of nine gestures generated from multiple videos, each repeated five times for a total of 45 trials for both the software-based and hardware-based modules. Apart from showing the performance of each approach, our results show that with a more improved hardware module, the accuracy of the combined approach can be significantly improved.

Keywords Assistive technology · Hearing impaired · Sign language · EMG · Video processing

✉ Ishfaq Ahmad
iahmad@cse.uta.edu

[1] Computer Science and Engineering, University of Texas at Arlington, Arlington, TX, USA

1 Introduction

From 9 to 22 out of every 1,000 Americans have a severe hearing impairment or are functionally deaf, with nearly 2 million profoundly deaf (which is roughly 0.7% of the U.S. population)– the numbers are even more staggering around the world. Furthermore, approximately 7.5 million people in the United States have trouble using their voices [13]. The most vulnerable groups are males between 15 and 24 years old, infants, toddlers, and the elderly over 75 years. These NIH statistics further indicate that 500,000 hospitalizations/year of such people occur due to injuries, resulting in mild/moderate problems affecting independent living, while 200,000 patients/year may have suffered from severe problems requiring institutionalization or close supervision, seriously hampering their quality of life. Technologically, our world is poised to embrace the next revolution of wireless-based technologies for anywhere and anytime usage with unprecedentedly advanced applications. The computing revolution that began with digitizing documents, photographs, and records so that they could more easily be manipulated is not over yet. Only its form has changed. The current wave in the technological revolution is the emerging wireless communications and Internet of Things (IoT) for enhanced computing and communications capabilities. IoT connects and views a world of networked smart devices, including applications from personal electronics, where everything is interconnected. In IoT, multimedia objects transform into smart objects which are able to sense, interpret and react to the environment enabled by the combination of the Internet and evolving wireless technologies such as Radio-frequency Identification (RFID). This technological revolution enables new means of communication not just between people and things but also between things themselves.

A reliable system for converting sign language to text can allow millions of people with communication disabilities to communicate more effectively with the general population. See Table 1 for an image and description of each letter in the American Sign Language.

Currently, two main approaches are used for sign language gesture recognition. The first is a hardware-based approach, using gloves, phone's internal sensors, electromyography (EMG) devices, or other sensors to track the position and movement of the hand. The second is a software-based approach, where the video is taken of the hands, and computer vision techniques are used to classify the gestures. Specialized gloves can be the most reliable and versatile. However, other hardware is less accurate. For hardware-based approaches, the user is required to have the necessary device throughout the entire day; this can be uncomfortable or hard to keep track of. Conversely, the software-based approach only requires a smartphone or computer with a camera. However, lighting conditions, background composition, and contrast can all affect the accuracy of the system.

We propose a combination of hardware and software techniques to form an alternative approach to sign language gesture recognition. Specifically, the *Myo* Armband developed by Thalmic Labs is used to read EMG and inertial measurement unit (IMU) data from the forearm and a Support Vector Machine classifier from the Open-Myo repository on GitHub [1] is adapted to classify sign language gestures. For the software-based unit, the "SIGN LANGUAGE GESTURE RECOGNITION FROM VIDEO SEQUENCES USING RNN AND CNN" repository from GitHub [24] is used. This unit is trained on a dataset of Argentinian Sign Language, so that is what we use for our system as a whole. The user of our prototype would perform sign language gestures while wearing the *Myo* armband and being video-taped, and once both modules (the vision-based module and the *Myo*-based module) have made a prediction, the results will be combined and weighted for a final prediction. Although this system uses Argentinian Sign Language, the same algorithm could

Table 1 The Letters of American Sign Language[a]

Letter	Image	Description	Letter	Image	Description	Letter	Image	Description
A		Make a fist upward with the thumb pointing up.	J		Like I, but swing the pinky finger to make a J shape in the air.	S		Make a fist and rest the thumb over the other fingers.
B		Hold hand up, point 4 fingers upwards and tuck the thumb into the middle of the palm.	K		Raise the middle finger and the index finger, and make sure they are separated. Place the thumb in between the two raised fingers, resting on the middle finger.	T		Make a fist and tuck the thumb in between the middle and index finger.
C		Cup fingers into a C shape.	L		Point the index finger vertically up and the thumb horizontally, so that the index finger and thumb form an L shape.	U		Point the middle and index fingers up and keep them together.
D		Put fingers into an O shape and raise the index finger.	M		Place the fingers into a fist, with the thumb over the pinky finger and the three other fingers over the thumb.	V		Point the middle and index fingers up and keep them separated.
E		Tuck the thumb into the middle of the palm and rest all of the other fingers over the thumb.	N		Place the fingers into a fist, with the thumb over the pinky and ring finger and the two other fingers over the thumb.	W		Point the middle, index, and ring fingers up, and keep them separated.
F		Put the index fingers and the thumb together to make an O shape and point the remaining fingers upward.	O		Cup the fingers and the thumb into an O shape.	X		Hold the index finger up and bend the tip of the finger to point at the person being talked to.
G		Take the index finger and the thumb, separate them 1 inch from each other and turn them so the index is pointing to the person being talked to.	P		Like K, but with the the fingers pointing down instead of up.	Y		Make a fist and point the thumb and pinky finger out to the sides.
H		Point the index and middle finger out so that the back of the hand is facing the person being talked to.	Q		Like G, but with the index finger and thumb pointing down, and the index finger facing the person being talked to.	Z		Use the index finger to make a Z shape in the air.
I		Hold the pinky finger upwards and rest the thumb over the other fingers.	R		Point the index and middle fingers up and wrap the index finger around the middle finger.			

[a]Images from [3]

be applied to other sign languages as well. The software module would have to be trained on a relevant dataset of segmented hands; although this system uses colored gloves for easy segmentation, any other method will work as well. For the hardware module, *Myo* armband data would have to be gathered for each sign used in the new sign language.

This paper extends the previous work in [56]. The contributions of this paper are as follows: further surveying of the field of sign language recognition is provided, with additional categorization and analysis of different techniques. The results of our experiments are discussed further and contrasted to other similar works. Additional conclusions and future work are provided.

This paper is organized as follows: Section 2 will cover some background for the field of sign language recognition. In Section 3 we will give a more in-depth background and technical overview of the hardware and software components specific to our project. In Section 4 we will describe the setup of our experiment. Section 5 will provide the results of our experiment, and Section 6 will conclude the paper and discuss further steps for future research.

2 Background

There are three major steps to any sign gesture recognition system: acquisition of data, extraction of features from the data, and classification of the data by running a recognition algorithm on the extracted features. The two main methods of acquiring data define the two most common approaches to gesture recognition: in the hardware-based approach, the data is acquired through sensors on the hands, and in the software-based approach, the data is acquired through a camera [13]. Hence, for each of these approaches, we will examine the issues of feature extraction and recognition.

2.1 Hardware-Based Approach

Three different methods of acquiring data using the hardware-based approach are through gloves, through the internal sensors in a smartphone, and through devices that collect EMG and IMU data, such as a *Myo* Armband [13]. Figure 1 provides an overview of these methods.

In many of the studies using gloves, data came from ten flex sensors and an IMU that give the posture and orientation of the hand. There are various algorithms that were used to classify the hand states as gestures, including template matching, minimum mean square error algorithm, and SVM. Each has its own benefits and drawbacks: for instance, SVM is generally very time-consuming but very accurate compared to the other methods [13]. Glove-based approaches are able to achieve high levels of accuracy. For example, the work in [47] showed an accuracy of 85% on five number signs. The work in [48] obtained accuracies of 99% using different classification methods on the twenty-six letters of ASL.

Using the internal sensors in a smartphone can be more convenient, as no additional hardware is required beyond the smartphone itself (Table 2 provides a general overview of existing works that utilize this method). However, phones can generally detect only two types of gestures: turn gestures and transition gestures. Hence, phones are better equipped to recognize dynamic gestures than they are to recognize static gestures (Tables 3 and 4 provide a general overview of existing works for static gestures and dynamic gestures, respectively). In previous studies that have used the internal sensors in a smartphone, the accelerometer continuously detected the three axis points of the phone in space, so the motion could be calculated by a vector containing the sum of the derivatives of the current axis with the previous axis. Again, there are multiple methods that were used for recognizing the hand gestures. One of the most common is the Dynamic Time Warping (DTW) algorithm. This algorithm does not require a lot of training data and takes into account the fact that different people will take different lengths of time to perform the same gesture, hence making it a popular choice [13]. Although limited in the kinds of gestures that can be detected, phone internal sensors are able to achieve high accuracy classification of appropriate gestures. [14] is able to classify 12 different motions with an accuracy of 94%. Although these are not signs from a sign language, this work could be utilized to allow other sign language systems to better classify dynamic gestures.

The final common option is a bio-potential approach, as described by [37]. Sequeira et al. propose a system for converting Indian sign language to text using Inertial Measurement Unit (IMU) and surface electromyography (sEMG) signals from the surface of the forearm. There is not as much research on the bio-potential approach as there is on the other hardware-based approaches. However, the bio-potential approach shows a lot of promis, because more data

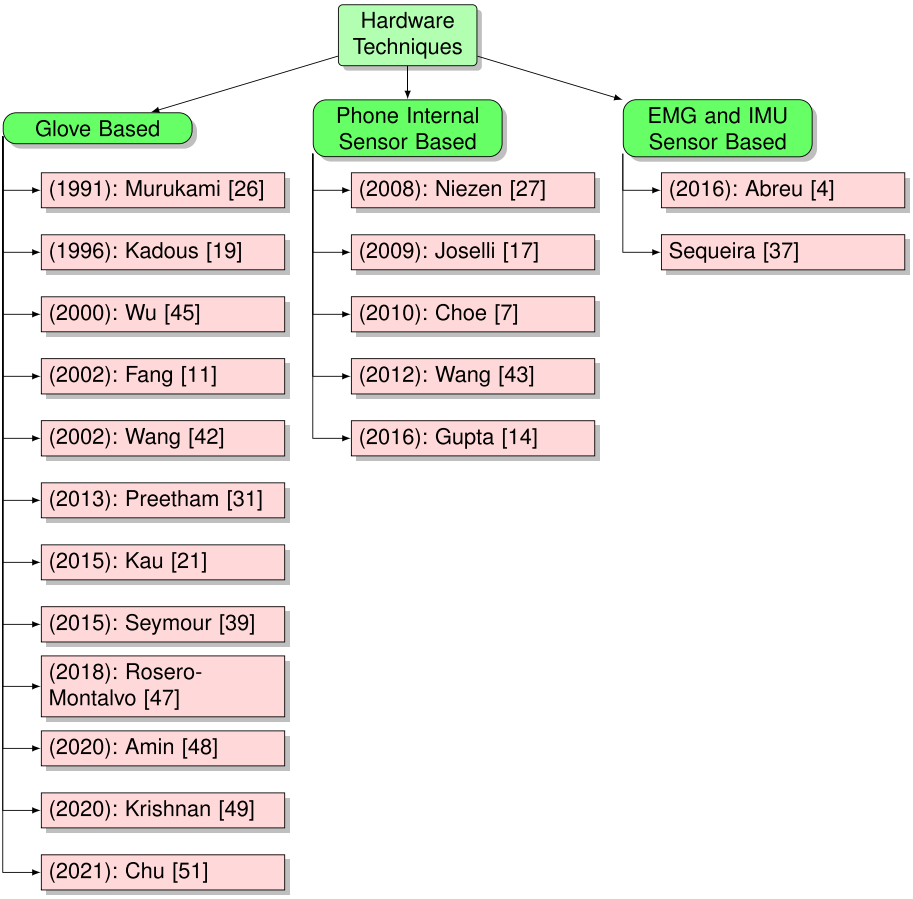

Fig. 1 Existing hardware techniques for sign language gesture recognition, displayed chronologically

can be collected than in the smartphone-based approach, and the set of sensors is not as cumbersome as in the glove-based approach. While [37] uses a specific set of sensors, it is also possible to collect EMG and IMU data through a *Myo* Armband [2]. Section 3 will go into further detail.

Table 2 A Selection of Existing Phone Internal Sensor-Based Systems[a]

System	Classification Method	Gesture Type	Processing	Voc. Size	Dependency
Choe 2010 [7]	DTW	Dynamic	Local	20	User-independent
Gupta 2016 [14]	DTW	Dynamic	Local	6	User-independent
Joeselli 2009 [17]	HMM + foreword-backward algorithm	Dynamic	Local	10	User-independent
Niezen 2008 [27]	DTW	Dynamic	Local	8	User-independent
Wang 2012 [43]	Own statistical method	Dynamic	Local	21	User-independent

[a]Table based on [13] and [28]

Table 3 A Selection of Other Existing Hardware Techniques for Static Gestures[a]

System	Sensors	Classification Method	Processing	Voc. Size	Dependency
Abreu 2016 [4]	Myo Armband	SVM	Local	20	User-independent
Kadous 1996 [19]	Gloves	Instance-based learning, Decision Tree	Local	95	User-independent
Preetham 2013 [31]	Gloves	Minimum Mean Square Error Algorithm	Local	-	-
Seymour 2015 [39]	Gloves	SVM	Local	31	User-independent
Rosero-Montalvo 2018 [47]	Gloves	Prototype Selection	Client–Server	-	User-independent
Amin 2020 [48]	Gloves	KNN, SVM, and Discriminant Analysis	Local	26	User-independent

[a]Table based on [13] and [28]

Table 4 A Selection of Other Existing Hardware Techniques for Dynamic Gestures[a]

System	Sensors	Classification Method	Processing	Voc. Size	Dependency
Fang 2002 [11]	Gloves	HMM, Self-organizing map, RNN	Local	208	User-independent
Kau 2015 [21]	Gloves	Template Matching	Local	5	User-independent
Murukami 1991 [26]	Gloves	RNN	Local	10	User-dependent
Sequeira [37]	Electrodes + Inertial Sensor	SVM	Local	-	-
Wang 2002 [42]	Gloves	HMMs model sequential subunits	Local	5119	User-dependent
Wu 2000 [45]	Gloves	Semi-Continuous Dynamic Gaussian Mixture Model	Local	274	User-dependent
Krishnan 2020 [49]	Gloves	Gradient Descent	Local	-	User-independent
Chu 2021 [51]	Gloves	SVM, DTW	Local	-	User-independent

[a]Table based on [13] and [28]

2.2 Software-Based Approach

As mentioned previously, the software-based approach is more challenging due to the multitude of variables, such as lighting conditions and background contrast, that can affect the accuracy of the results. The data collection step is fairly straightforward: the internal camera in a smartphone will suffice, albeit it is often necessary for photos or videos to be recorded in a controlled setting because of the feature extraction step. Prior research in image and video processing includes [54] and [55], which explore the collection, coding, and allocation of video data.

The real challenge with this approach is the feature extraction step after the video has been collected. Hand orientation and position can be detected in numerous ways, including via skin detection or Viola-Jones cascades of boosted rectangle filters. Skin segmentation algorithms are commonly used in order to separate the hands from background noise. Additional hand details are extracted using various methods as well. For instance, the number of open fingers can be measured by finding contours, and the palm area can be measured by finding the largest circle that fits in the hand region.

Common methods for classification in the software-based approach include SVM (Table 5), Template Matching (Table 6), and Hidden Markov Models (Table 7). Other methods that are used can be found in Table 8. Additionally, Fig. 2 provides a chronological overview of existing works in each of these domains

3 Technical Overview

3.1 Hardware-Based

The *Myo* Armband from Thalmic Labs has been used by [4] to collect muscle data from the arm, which is then fed through a classification algorithm in order to classify letters of the Brazilian Sign Language (LIBRAS).

Table 5 A Selection of Existing Software using SVM Classification[a]

System	Feature Extraction	Processing	Voc. Size	Dependency
Elleuch 2015 [10]	Skin detection HSV, convexity defects	Local	5	User-independent
Hays 2013 [15]	Skin detection YCrCb, canny edge	Local, client–server	32	User-independent
Jin 2016 [16]	Skin detection RGB, canny edge, SURF	Local	16	User-dependent
Joshi 2015 [18]	PCA	Local	5	User-independent
Lahiani 2015 [23]	Skin detection RGB, convexity defects	Local	10	User-dependent
Fayyaz 2019 [52]	SURF	-	-	User-independent

[a]Table based on [13] and [28]

The *Myo* Armband is designed to be worn just below the elbow on the fatty part of the forearm. It contains eight evenly spaced EMG sensors that collect data from the arm muscles. The *Myo* Armband also consists of an IMU that collects data using an accelerometer, gyroscope, and magnetometer. Because of the limitations of the armband, Abreu et al. make some assumptions to simplify the experiments in [4]. First, it is assumed that the *Myo* Armband is worn by the same person, as the EMG data can vary significantly between individuals. Additionally, the *Myo* Armband is always assumed to be placed in the exact same position on the forearm. If it is not, the sensors will be reading different muscles which makes the data impossible to compare with any accuracy. In [4], only EMG data was used, which means that the system could only be trained on static letters. Dynamic letters (those that involve moving the arm as well as positioning fingers) could also be trained, but this would require the use of the IMU data as well as the EMG data.

The training set used in [4] consisted of 28,500 EMG reading sample for each letter used. Only twenty letters were used. The data was preprocessed by taking the absolute value of the data in order to reduce the area of the feature space. After preprocessing, the SVM classifier was used to classify the data as each of the twenty letters. The one-vs-all method was used due to the fact that a gesture can be none of the twenty letters in the model. Therefore, each classifier assigns either "positive" or "negative" for its individual letter. If all twenty classifiers return "negative" for a gesture, then it is labeled as "none," meaning it was not any of the twenty letters in the model.

As shown in Table 9, only a small portion of letters were able to be labeled accurately in real-time. As it stands, the *Myo* Armband is not a reliable tool for sign language

Table 6 A Selection of Existing Software using Template Matching Classification[a]

System	Feature Extraction	Processing	Voc. Size	Dependency
Gandhi 2015 [12]	Background subtraction	Local	-	-
Kamat 2016 [20]	Skin detection RGB	Local	4	User-dependent
Raheja 2015 [32]	Sobel Edge Filter, PCA	Client–server	10	User-dependent
Saxenal 2014 [36]	Skin detection RGB, PCA	Client–server	10	User-dependent
Shrenika 2020 [53]	Canny edge detection	Local	-	-

[a]Table based on [13] and [28]

Table 7 A Selection of Existing Software Techniques using Hidden Markov Models[a]

System	Feature Extraction	Classification Method	Processing	Voc. Size	Dependency
Assan 1998 [5]	2D moment based	HMM	Local	262	User-dependent
Bauer 2002 [6]	2D moment based	HMMs model sequential subunits	Local	100	User-dependent
Kobayashi 1997 [22]	2D moment based	Partly-HMM	Local	6	User-independent
Starner 1998 [40]	2D moment based	HMM	Local	40	User-dependent
Tanibata 2002 [41]	2D moment based + protrusions	HMMs model LH;RH	Local	65	User-dependent

[a]Table based on [13] and [28]

Table 8 A Selection of Other Existing Software Techniques[a]

System	Feature Extraction	Classification Method	Processing	Voc. Size	Dependency
Cui 2000 [9]	2D segmented hand, position	Recursive PCA + MDA	Local	28	-
Hakkun 2015 [38]	Viola-Jones Haar Filters	KNN	Local	8	User-dependent
Huang 1998 [8]	2D FD, hand orientation, motion vector between frames	3D Hopfield NN	Local	15	-
Masood 2017 [24]	CNN	RNN	Local	46	User-independent
Matsuo 1998 [25]	3D hand position	Rule-Based	Local	38	User-dependent
Prasuhn 2014 [30]	Skin detection HUV, HOG	Brute-force Matching	Client–Server	26	User-dependent
Rao 2016 [33]	Gaussian and Sobel Edge Filter + PCA	MDC	Local	18	User-independent
Saxena 2014 [35]	Sobel Edge Filter	Backpropagation Algorithm	Client–Server	5	User-dependent
Warrier 2016 [44]	Skin detection RGB	Geometric Matching	Client–Server	11	User-dependent
Yang 2002 [46]	2D pixel motion trajectories	Time-Delay NN	Local	40	-
Makarov 2019 [50]	_	CNN	Local	-	-

[a]Table based on [13] and [28]

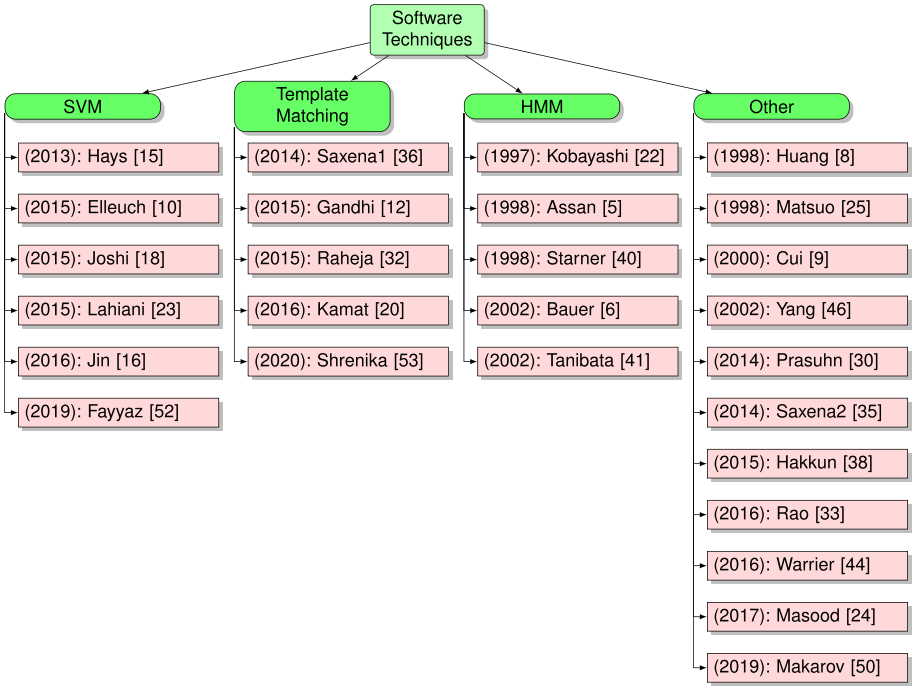

Fig. 2 Existing software techniques for sign language gesture recognition, displayed chronologically

recognition. Furthermore, the obstacles that inspired the simplifying assumptions in the original work, namely the fact that EMG readings vary greatly from person to person and also between different positionings on the same person's forearm, further reduce the usability of the *Myo* Armband in sign language recognition systems. However, although the *Myo* Armband cannot be used by itself for sign language gesture recognition, it does have potential to be used in addition to another tool, such as a software-based module, in order to attempt to improve the accuracy of the overall system. This idea will be explored further in Sect. 4 of this paper.

3.2 Software-Based

The "SIGN LANGUAGE GESTURE RECOGNITION FROM VIDEO SEQUENCES USING RNN AND CNN" software [24] has been made available on GitHub. The first step in this software is to separate the videos into individual frames, and in each of these frames, the hands are extracted from the rest of the image. Next, the spatial features of each image are extracted using CNN. Specifically, the Inception-v3 model of the TensorFlow library is used to classify the spatial features of each image.

Some example frames are shown in Fig. 3. Further samples are provided in Fig. 4, which shows two different participants performing the sign for "man." Fig. 5 shows two different participants performing the sign for "music." While "man" only uses the right hand, "music" uses both hands. These signs are dynamic, so each of these frames is from a video where the person is moving their hands in a specific way. Each video has a plain colored

Table 9 Real Time Classification Accuracy

Letter Gesture	Accuracy
A	4%
B	19%
C	49%
D	64%
E	76%
F	8%
G	40%
I	55%
L	77%
M	8%
N	57%
O	47%
P	8%
Q	48%
R	91%
S	46%
T	4%
U	5%
V	22%
W	95%

Fig. 3 Samples from the Argentinian Sign Language Dataset [34]

Fig. 4 Two participants performing the Argentinian sign for "man" [34]

Fig. 5 Two participants performing the Argentinian sign for "music" [34]

background, dark clothes, and colored gloves; this is to make hand extraction simpler. Examples of some frames after hand extraction are shown in Figs. 6 and 7. The positions of each hand within the frame is stored. The handshapes and motion are then classified.

The first step after the features are extracted with the CNN is to change how the videos are represented so that they can be input into the RNN to train on the temporal features. There are two approaches used in this software. The first, Approach 1, represents each video as a sequence of n-dimensional vectors, where n is the number of classes. Each vector represents one frame and contains the predictions made by the CNN for that frame. In Approach 2, each frame is again represented by a vector. However, instead of the vectors being n dimensional for the number of classes, they have 2048 dimensions where each dimension is one output of the last pool layer of the CNN. Therefore, the prediction is not made by the CNN before the vectors are fed into the RNN. In this way, predictions are not made for each frame in the CNN, but are predicted in the RNN.

Next, the RNN must be trained. This training is different depending on whether Approach 1 or Approach 2 was used in the previous step. For Approach 1, the RNN is trained on the "softmax-based representation" of gestures. For Approach 2, the RNN is trained on the "pool layer-based representation" of gestures. The only difference in the RNN training is in the input layer, as it must match the format of the input data, which is either predictions made by the CNN or the output of the final pooling layer. After the input layer, the network is the same and training is performed the same way.

Both approaches were tested on a dataset of 46 gestures in Argentinian Sign Language [34]. The dataset that was used had specific background and clothing restrictions in order to aid in hand extraction. Participants wore plain black shirts and different colored gloves on each hand. The background was plain white. Having these constraints makes it easier to extract the hands from the background, and differentiate right and left hands, but it is not necessary for training the system. Images are converted to grayscale after segmentation occurs, so the color of the gloves does not affect the training of the model. If a different

Fig. 6 The right hand shape extracted from the sign for "man" [34]

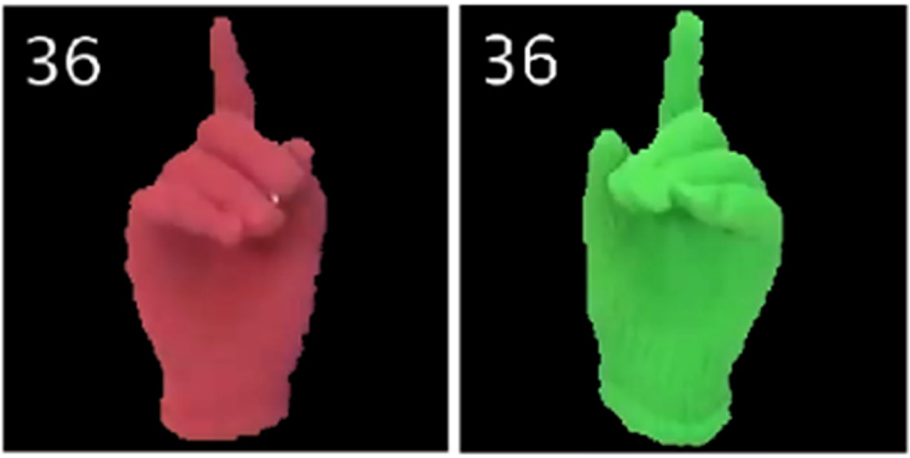

Fig. 7 The right and left hand shapes extracted from the sign for "music" [34]

method of hand extraction were used, the model should also work on a dataset without colored gloves. Approach 1 had an accuracy of 80.87%, and Approach 2 had an accuracy of 95.21%. Both approaches, especially Approach 2, are very effective at recognizing sign-language gestures using a software based approach.

4 Experimental Setup

The goal of this experiment is to combine hardware and software techniques for sign language recognition in order to test if the overall accuracy can be increased. Specifically, we sought to answer if combining a classifier using EMG data from the *Myo* Armband with a classifier for data extracted from smart-phone video can increase the overall accuracy of sign language gesture recognition.

The idea behind the software is that the user would perform sign language gestures while wearing a *Myo* Armband and simultaneously being filmed by a smart-phone camera. The videos are then fed through the software-based module while the EMG data is fed through the hardware-based module. Both modules output their prediction matrices, which are combined to form a final prediction for a gesture. The software-based module that we used is the "SIGN LANGUAGE GESTURE RECOGNITION FROM VIDEO SEQUENCES USING RNN AND CNN" program from GitHub.

The classifier for the *Myo*-based module is a modified version of one found in the "Open Myo" repository on github [1]. Muscle data was collected using a modified version of the "save_emg_signals.py" file, where each reading of muscle data was saved as a list of 12 integers. The first 8 integers in the list represent the EMG readings from the 8 sensors on the *Myo* Armband, and the last 4 integers in the list are the real numbers from a quaternion reading that represents the orientation of the *Myo* Armband in space. We found that the quaternion readings had more of an effect on the accuracy of the classifier than the EMG readings, so accordingly, the quaternion readings were given a stronger weight. The data

was dumped into a Pickle file, so that it could be loaded into a modified version of the ``emg_clf_test.py" file from the Open-Myo repository for classification.

Our modified version of the "emg_clf_test.py" file from the Open-Myo repository works as follows: first, the dictionary containing training data and the dictionary containing test data is loaded into the program. Next, the two dictionaries are reformatted into two lists of arrays, organized according to the specific iteration of each gesture. After that, the training data and test data are both segmented—that is, they are placed into a list of arrays, where there is one array for each data point (of which there are 12) from each individual gesture. The program will decide a "number of segments", which is the length of each of these arrays. This number is important because it is the number of predictions that the program ultimately ends up making for each iteration, and whichever gesture is predicted the largest number of times for an iteration is the "final prediction" of the program for that iteration. Then, two feature matrices are created (again, one for training data and one for test data), which contain data about the features for each gesture, such as the mean, root mean square, and variance of data points. The last of the pre-processing steps is to create a target matrix that contains the actual values that should be predicted by the classifier, as well as dimensionality reduction (the 12-dimensional data is scaled down to 2 dimensions).

After the above preprocessing steps, we begin training the model. The training data is fed through a Support Vector Machine (SVM) classifier from the Python Scikit-Learn Library [29] and each segment of each gesture is given a prediction. In order to create a prediction matrix that can be combined with the output from the software-based module, the likelihood of a particular gesture is calculated based on the predictions for each segment of the iteration. Our model was trained using only nine gesture from the Argentinian sign language database: "man," "skimmer," "music," "green," "bright," "drawer," "away," "learn," and "accept." Our test data included five iterations of each gesture. If we suppose the pre-processing step segmented the data into six segments, then the final prediction matrix will be a list of 45 lists, where each list corresponds to a specific iteration and provides the likelihood of that iteration being each of the nine gestures.

The prediction matrix is formatted in this way to match the format of the predictions matrix from the "rnn_eval.py" file of the "SIGN LANGUAGE GESTURE RECOGNITION FROM VIDEO SEQUENCES USING RNN AND CNN" module. Therefore, once the prediction matrix is created with our modified version of the Open-Myo file, it is dumped into a pickle file so that it can be loaded into "rnn_eval.py" and combined with the prediction matrix from the software-based module. The weights of the two modules can be adjusted to give priority to one over the other.

As stated before, the idea of this software is to perform a gesture while wearing the *Myo* Armband and being filmed simultaneously. However, due to limitations of the *Myo* Armband (discussed further in the conclusions of this paper), the eight sensors must be placed in the exact spot on the forearm for training and testing data; otherwise, the EMG data will vary significantly between sessions. Therefore, the training and testing datasets had to be collected in a single session without removing or repositioning the *Myo* Armband. Due to time constraints, we were forced to use separate reading for the software-based and hardware-based modules, so they were not able to be recorded simultaneously.

The video data used was a subset of the Argentinian Sign Language dataset [34], and was the same subset as used in [24]. The 50 iterations of the nine gestures are in the Argentinian Sign Language dataset, so we split them into a training set of 45 iterations and a test set of five iterations. The hardware-based module data was collected separately, with 50 iterations of each gesture used for training and 5 used for testing. The *Myo* Armband data was collected

from one person in a single sitting, without removing or repositioning the armband, in order to obtain the most accurate data possible.

5 Results

Below are the classification results of the *Myo*-based module in Table 10. 45 iterations were performed; five for each of the nine gestures used. Note that 22 of the 45 gesture iterations are predicted correctly. Also note that Fig. 8 provides a visualization of the muscle data from the *Myo* Armband after the pre-processing steps of the SVM classifier.

Next are the results for the software-based module (Table 11), in the same format as the prediction table for the *Myo*-based module. This module predicts 40 of the 45 iterations correctly, all gestures except for “skimmer", which it predicted as “bright". This is not unexpected, as these are two very similar gestures. “Bright" involves waving the hand away from and then back towards the body while holding all fingers and the thumb up, while “skimmer" is the same except for not holding up the thumb.

The combination of these 2 matrices gives us our final result. Unfortunately, as the *Myo*-based module also had a zero accuracy for “skimmer," there is no way to combine these two matrices such that more than 40 of the 45 gesture iterations are identified correctly, meaning that with these results, it is impossible to improve upon the results from the software-based module on its own.

Based only on our results, the combination of hardware and software techniques seems unable to improve the overall accuracy of sign language gesture recognition systems. Not only did the combination of the software-based module “SIGN LANGUAGE GESTURE RECOGNITION FROM VIDEO SEQUENCES USING RNN AND CNN” with the EMG and IMU classifier for the *Myo* Armband not improve the accuracy, but the accuracy also decreases as the weight given to the predictions matrix from the *Myo*-based module is increased. This is unsurprising due to the small test size and the limitations of the *Myo* Armband with regards to sign language recognition. Some of the limitations are discussed below.

EMG data is dependent upon many different factors. The shape of an individual’s forearm is always different, which can make EMG data quite variable between people.

Table 10 Classification Accuracy of the *Myo*-Based Module

Gesture	Correct Classifications out of 5 trials
Man	0
Skimmer	0
Music	5
Green	0
Bright	5
Drawer	0
Away	5
Learn	3
Accept	4
Total	22 / 45

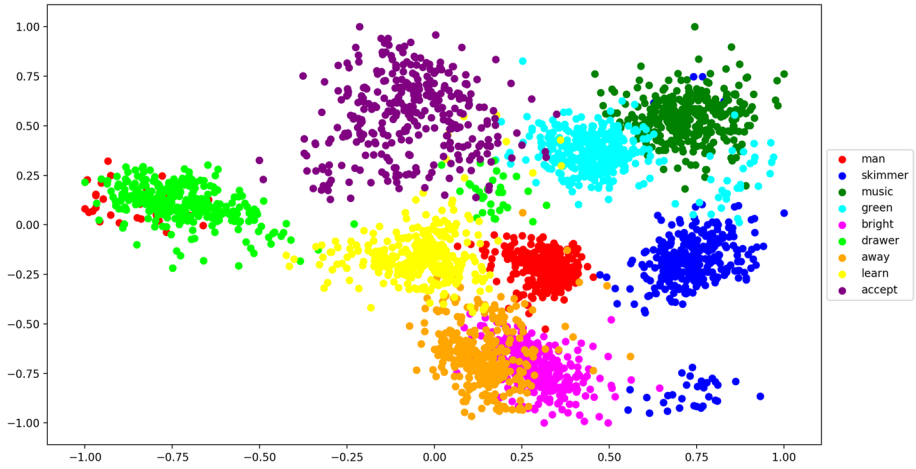

Fig. 8 Visualization of 12-dimensional gesture data from the *Myo* Armband, reduced to 2 dimensions [56]

Therefore, in order to use the *Myo* Armband to collect data for sign language recognition, the program must be trained on each individual user, which can severely limit the practical uses of such a system. In order to use such a system, each person would have to provide training data for all the signs that the system would have to recognize, which would be a very arduous and difficult process. Additionally, the *Myo* Armband is dependent not only on different forearms but also on where it is placed on the same forearm. When the device is removed and replaced for a second session, the previous training data may prove entirely useless, meaning that some amount of retraining would have to be done to allow the system to function properly again.

The *Myo*-based module is comparable to a system like [4] and has comparable accuracy to some signs being classified with very high accuracy and others with very low. The average accuracy across all signs for this system is 49% compared to the results from Table 9. The software-based module has an accuracy of 89%, which is similar to the results of [52] at 86%. However, the combination of the modules in this experiment does not improve the

Table 11 Classification Accuracy of the Software-Based Module

Gesture	Correct Classifications out of 5 trials
Man	5
Skimmer	0
Music	5
Green	5
Bright	5
Drawer	5
Away	5
Learn	5
Accept	5
Total	40 / 45

results of the classification. Given the limitations of the *Myo* armband discussed above, it appears that it may be unable to improve the accuracy of similar software-based systems.

6 Conclusions

The most significant limitation of our system is that the *Myo* Armband must be placed in the exact same position on the forearm for each session for the most accurate results; otherwise, more training data will have to be collected for each session. Clearly, the recreation of the training set for each time the armband is removed is impractical, especially given that the final system would include the majority of words in the particular sign language. Each gesture would have to be performed again each time the armband is removed. However, it is also impractical to attempt to ensure that the *Myo* Armband is placed exactly in the same position on the forearm for each use. A mark on the skin, such as a freckle, could be used to mark the location of one of the sensors, but there are seven other sensors on the device that may be shifted enough to change the collected data.

It is also important to remember that the *Myo* Armband is not designed for gestures as complicated as those in sign languages. This can be seen easily by looking at the built-in recognizable gestures. A "wave out" of the hand is a very simple gesture, but many sign language gestures are far more detailed than a "wave out." Many words resemble each other closely, with the only differences being subtle details in the positioning of the fingers. Differentiating these minute details between gestures is an impossible task for the *Myo* Armband, given that the armband is incapable of tracking the orientation and position of each individual finger in space.

In our future work, a software-based module should be combined with other forms of hardware. While it may not be practical or as comfortable, it is likely that a combination of a glove-based system and the software-based module would show an increase in overall accuracy, given that both systems can be highly accurate on their own. This is in part due to the fact that gloves can track the alignment and position of each finger individually. Another question to be explored is whether there is a more reliable way to collect EMG data from the forearm, such as a device that is able to be placed in the same location more consistently and can then be used over many separate sessions, despite being removed and replaced. Additionally, and EMG device could be tested in conjuncture with a less controlled video dataset. As discussed, the software-based modules are dependent on lighting and background conditions, so using a real-world video dataset may decrease the accuracy of the software-based system. However, biopotential approaches such as EMG are not affected by these conditions and therefore might be able to improve the accuracy of the system overall if used in real-world environments.

Funding This project was funded by the National Science Foundation under Award Number:1757641.

Availability of Data and Material Not applicable.

Code availability Not applicable.

Declarations

Conflicts of Interest Not applicable.

References

1. Open-myo. https://github.com/Alvipe/Open-Myo. Accessed: 2019–07–08.
2. Welcome to myo support. https://support.getmyo.com/hc/en-us. Accessed: 2019–07–08.
3. Wikipedia: American manual alphabet. https://en.wikipedia.org/wiki/americanmanualalphabet. Accessed: 2019–07–09.
4. Abreu JG, Joao Marcelo Teixeira, Lucas Silva Figueiredo, and Veronica Teichrieb (2016) Evaluating Sign Language Recognition Using the Myo Armband Proceedings- 18th Symposium on Virtual and Augmented Reality, SVR 2016, pages 64–70
5. Assan M, Grobel K (1998) Video-based sign language recognition using hidden markov models Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 1371:97-109
6. Britta Bauer and Karl-friedrich Kraiss (2002) Video-Based Sign Recognition Using Self-organizing Subunits. pages 434–437
7. Choe BW, Jun-Ki Min, and Sung-Bae Cho (2010) Online gesture recognition for user interface on accelerometer built-in mobile phones. In Proceedings of the 17th International Conference on Neural Information Processing: Models and Applications - Volume Part II, ICONIP'10, pages 650–657, Berlin, Heidelberg, Springer-Verlag
8. Huang C-L and Wen-Yi Huang (1998) Sign language recognition using model-based tracking and a 3D Hopfield neural network. pages 292–307
9. Cui Y and Juyang Weng (2000) Appearance-based hand sign recognition from intensity image sequences. Computer Vision and Image Understanding
10. Hanene Elleuch, Ali Wali, Anis Samet, and Adel M. Alimi (2016) A static hand gesture recognition system for real time mobile device monitoring. International Conference on Intelligent Systems Design and Applications, ISDA, 2016-June: 195–200
11. Gaolin Fang, Wen Gao, Xilin Chen, Chunli Wang, and Jiyong Ma (2002) Signer-Independent Continuous Sign Language Recognition Based on SRN/HMM. pages 76–85
12. Gandhi P, Dalvi D, Gaikwad P, Khode S (2015) Image Based Sign Language Recognition on Android. Int J Eng Tech 1(5):55–60
13. Ghanem S, Christopher Conly, and Vassilis Athitsos. (2017) A Surveyon Sign Language Recognition Using Smartphones. pages 171–176. Association for Computing Machinery (ACM)
14. Gupta HP, Haresh S. Chudgar, Siddhartha Mukherjee, Tanima Dutta, and Kulwant Sharma (2016) A Continuous Hand Gestures Recognition Technique for Human-Machine Interaction Using Accelerometer and Gyroscope Sensors. IEEE Sensors J 16(16):6425–6432
15. Hays P, Raymond Ptucha, and Roy Melton (2013) Mobile device to cloud co-processing of ASL finger spelling to text conversion. 2013 IEEE Western New York Image Processing Workshop, WNYIPW 2013 -Proceedings, (1):39–43
16. Jin CM, Zaid Omar, and Mohamed Hisham Jaward (2016) A mobile application of American sign language translation via image processing algorithms. Proceedings - 2016 IEEE Region 10 Symposium, TENSYMP2016, pages 104–109
17. Joselli M and Esteban Clua (2009) gRmobile: A framework for touch and accelerometer gesture recognition for mobile games. SBGAMES2009 -8th Brazilian Symposium on Games and Digital Entertainment, pages141–150
18. Joshi TJ, N.Z. Tarapore, Shiva Kumar, and Vivek Mohile (2015) StaticHand Gesture Recognition using an Android Device. Int J Comput Appl 120(21):48–53
19. Kadous. Mohammed Waleed (1996) Machine recognition of auslan signs using power gloves: towards large lexicon recognition of sign language. Proceeding of the Workshop of Gestures in Language and Speech, pages165–174
20. Kamat R, Danoji A, Dhage A, Puranik P, Sengupta S (2016) MonVoix-An Android Application for the acoustically challenged people. J Commun Technol Electron Comput Sci 8(November):24
21. Kau LJ, Wan Lin Su, Pei Ju Yu, and Sin Jhan Wei (2015) A real-time portable sign language translation system. Midwest Symposium on Circuits and Systems, 2015-Septe(1):1–4
22. Kobayashi T and Haruyama S (1997) Partly-hidden Markov model and its application to gesture recognition. pages 3081–308
23. Lahiani H, Mohamed Elleuch, and Monji Kherallah (2015) Real time hand gesture recognition system for android devices. In International Conference on Intelligent Systems Design and Applications, ISDA, pages591–596. IEEE
24. Masood S, Adhyan Srivastava, Harish Chandra Thuwal, and Musheer Ahmad (2017) Real Time Sign Language Gesture Recognition From Video Sequences. 110017

25. Matsuo H, Igi S, Shan Lu, Nagashima Y Takata Y, Teshima T (1998) The recognition algorithm with non-contact for Japanese sign language using morphological analysis Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 1371:273-284
26. Murakami K and Hitomi Taguchi (1991) Gesture Recognition using Recurrent Neural Networks. Proc. SIGCHI Conf. Human Factors in Computing Systems, pages 237–242
27. Niezen G and GP Hancke GP (2008) Gesture recognition as ubiquitous input for mobile phones. International Workshop on Devices that Alter Perception, (Dap):17–21
28. Ong SCW and Surendra Ranganath (2005) Automatic sign language analysis: A survey and the future beyond lexical meaning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(6):873–891
29. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duch-esnay E (2011) Scikit-learn: Machine learning in Python. J Mach Learn Res 12:2825–2830
30. Prasuhn L, Yuji Oyamada, Yoshihiko Mochizuki, and Hiroshi Ishikawa (2014) A HOG-based hand gesture recognition system on a mobile device. 2014 IEEE International Conference on Image Processing, ICIP2014, pages 3973–3977
31. Preetham C, Girish Ramakrishnan, Sujan Kumar, Anish Tamse, and Nagendra Krishnapura (2013) Hand talk-implementation of a gesture recognizing glove. Proceedings - 2013 Texas Instruments India Educators' Conference, TIIEC 2013, pages 328–331
32. Raheja J, Singhal A, Sadab, and Ankit Chaudhary (2015) Android Based Portable Hand Sign Recognition System. pages 1–18
33. Rao GA and Kishore PVV (2016) Sign language recognition system simulated for video captured with smart phone front camera. Int J Electric Comput Engi 6(5):2176–2187
34. Ronchetti F, Facundo Quiroga, Cèsar Estrebou, and Laura Lanzarini. Handshape recognition for Argentinian Sign Language using ProbSom. Technical report.
35. Saxena S, Deepak Kumar Jain, and Ananya Singhal (2014) Hand gesture recognition using an android device. Proceedings - 20144th International Conference on Communication Systems and Network Technologies, CSNT 2014, pages 819–822
36. Saxena A, Deepak Kumar Jain, and Ananya Singhal (2014) Sign language recognition using principal component analysis. Proceedings - 20144th International Conference on Communication Systems and Network Technologies, CSNT 2014, pages 810–813
37. Sequeira S, Naik GM, Parab JS, and Gad RS. SIGN LANGUAGE RECOGNITION USING sEMG AND IMU.
38. Setiawardhana R,Hakkun Y, and Achmad Baharuddin (2015) Sign language learning based on Android for deaf and speech impaired people. Proceedings -, 2015 International Electronics Symposium: Emerging Technology in Electronic and Information IES 2015 pages114–117
39. Seymour M and Mohohlo Tsoeu (2015) A mobile application for South African Sign Language (SASL) recognition. IEEE AFRICON Conference, 2015-Novem:1–5
40. Starner T, Weaver J, Pentland A (1998) Real-time American sign language recognition using desk and wearable computer based video. IEEE Trans Pattern Anal Mach Intell 20(12):1371–1375
41. Tanibata N, Nobutaka Shimada, and Yoshiaki Shirai (2002) Extraction of hand features for recognition of sign language words. The 15thInternational Conference on Vision Interface, (August 2014):391–398
42. Wang C, Wen Gao, and Shiguang Shan (2002) An approach based on phonemes to large vocabulary Chinese sign language recognition. Proceedings - 5th IEEE International Conference on Automatic Face Gesture Recognition, FGR 2002, pages 411–416
43. Wang X, Paula Tarrío, Eduardo Metola, Ana Barbolla, and Josè R. Casar (2012) Gesture Recognition Using Mobile Phone's Inertial Sensors 151:173–184. 01
44. Warrier KS, Jyateen Kumar Sahu, Himadri Halder, Rajkumar Koradiya, and Karthik Raj V (2016) Software based sign language converter. International Conference on Communication and Signal Processing, ICCSP 2016, pages 1777–1780
45. Wu J, Gao W (2000) A Fast Sign Word Recognition Method for Chinese Sign Language 2:599–606
46. Yang MH, Narendra Ahuja, and Mark Tabb (2002) Extraction of 2D motion trajectories and its application to hand gesture recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(8):1061–1074
47. Rosero-Montalvo PD et al (2018) Sign Language Recognition Based on Intelligent Glove Using Machine Learning Techniques. IEEE Third Ecuador Technical Chapters Meeting (ETCM) 2018:1–5. https://doi.org/10.1109/ETCM.2018.8580268

48. Amin MS, Amin MT, Latif MY, Jathol AA, Ahmed N and Tarar MIN (2020) "Alphabetical Gesture Recognition of American Sign Language using E-Voice Smart Glove," 2020 IEEE 23rd International Multitopic Conference (INMIC) pp. 1–6 https://doi.org/10.1109/INMIC50486.2020.9318185.
49. Krishnan A, Vijay A, B. M and S. BS (2020) "Gesture Recognizer and Communicator using Flex Sensors and Accelerometer with Logistic Regression," 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS) pp. 391–394, https://doi.org/10.1109/ICISS49785.2020.9315897.
50. Ilya Makarov, Nikolay Veldyaykin, Maxim Chertkov, and Aleksei Pokoev (2019) American and russian sign language dactyl recognition. In *Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments* (*PETRA '19*). Association for Computing Machinery, New York, NY, USA, 204–210. https://doi.org/10.1145/3316782.3316786
51. Chu X, Liu J and Shimamoto S (2021)"A Sensor-Based Hand Gesture Recognition System for Japanese Sign Language," 2021 IEEE 3rd Global Conference on Life Sciences and Technologies (LifeTech) pp. 311–312 https://doi.org/10.1109/LifeTech52111.2021.9391981.
52. Sobia Fayyaz and Yasar Ayaz (2019) CNN and Traditional Classifiers Performance for Sign Language Recognition. In *Proceedings of the 3rd International Conference on Machine Learning and Soft Computing* (*ICMLSC 2019*). Association for Computing Machinery, New York, NY, USA, 192–196. DOI:https://doi-org.ezproxy.uta.edu/https://doi.org/10.1145/3310986.3311011
53. Shrenika S and Madhu Bala M (2020) "Sign Language Recognition Using Template Matching Technique," 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA), 2020, pp. 1–5 https://doi.org/10.1109/ICCSEA49143.2020.9132899
54. Yu-Kwong Kwok, Karlapalem K, Ahmad I and Moon Pun NG (1996) "Design and evaluation of data allocation algorithms for distributed multimedia database systems," in IEEE Journal on Selected Areas in Communications, 14(7):1332–1348 https://doi.org/10.1109/49.536483
55. Sun Yu, Ahmad I (Oct. 2004) A robust and adaptive rate control algorithm for object-based video coding. IEEE Trans Circuits Syst Video Technol 14(10):1167–1182. https://doi.org/10.1109/TCSVT.2004.833164
56. Dignan C, Perez E, Ahmad I, Huber M and Clark A (2020) "Improving Sign Language Recognition by Combining Hardware and Software Techniques," 2020 3rd International Conference on Data Intelligence and Security (ICDIS), pp. 87–92 https://doi.org/10.1109/ICDIS50059.2020.00018.