SENSOR-BASED CONTINUOUS ARABIC SIGN LANGUAGE RECOGNITION

by

Noor Ali Tubaiz

A Thesis Presented to the Faculty of the

American University of Sharjah

College of Engineering

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in

Computer Engineering

Sharjah, United Arab Emirates

June 2014

Approval Signatures

We, the undersigned, approve the Master's Thesis of Noor Ali Tubaiz.

Thesis Title: Sensor-Based Continuous Arabic Sign Language Recognition

Signature	**Date of Signature** (dd/mm/yyyy)
_____	_____
Dr. Tamer Shanableh Associate Professor, Department of Computer Science and Engineering Thesis Advisor	
_____	_____
Dr. Khaled Assaleh Professor, Department of Electrical Engineering Thesis Co-Advisor	
_____	_____
Dr. Mohammad Jaradat Associate Professor, Department of Mechanical Engineering Thesis Committee Member	
_____	_____
Dr. Ghassan Qadah Associate Professor, Department of Computer Science and Engineering Thesis Committee Member	
_____	_____
Dr. Assim Sagahyroon Head, Department of Computer Science and Engineering	
_____	_____
Dr. Hany El-Kadi Associate Dean, College of Engineering	
_____	_____
Dr. Leland Blank Dean, College of Engineering	
_____	_____
Dr. Khaled Assaleh Director of Graduate Studies	

Acknowledgements

All Praise and Glory be to Allah, the Most Gracious and the Most Merciful, for His blessings and for giving me the will to complete my thesis.

First of all, I would like to express my profoundest appreciation and gratitude for my respectful advisors, Dr. Tamer Shanableh and Dr. Khaled Assaleh, for their support, guidance, and patience. Thank you for your precious time and may Allah reward and bless you.

I am sincerely thankful for Dr. Assim Sagahyroon, the Computer Science and Engineering department head, for his advice and encouragement. Besides, I would like to show my appreciation to Dr. Ghassan Qadah and Dr. Mohammad Jaradat for their valuable suggestions while examining this thesis. I also extend my great appreciation to: Dr. Abdulrahman Al-Ali, Dr. Michel Pasquier, Dr. Verica Gajic, Dr. Taha Landolsi, Dr. Khaled El Fakih, Dr. Kevin Loughlin, and Dr. Khaled Assaleh for their great teaching efforts during my Masters courses.

Furthermore, special thanks goes to Ms. Lalitha Murugan, Mr. Narayanan Madathumpadical, Ms. Salwa Mohammad, and my colleagues and dear friends for always being there to help and support me.

Last but not least, I am sincerely grateful to my family for their love and unlimited support and guidance throughout all the stages of my life.

To my family and professors.

Abstract

Arabic sign language is the most common way of communication between the deaf and the hearing individuals in the Arab world. Due to the lack of knowledge of Arabic sign language among the hearing society, deaf people tend to be isolated. Most of the research in this area is focused on the level of isolated gesture recognition using vision-based or sensor-based approaches. While few recognition systems were proposed for continuous Arabic sign language using vision-based methods, such systems require complex image processing and feature extraction techniques. Therefore, an automatic sensor-based continuous Arabic sign language recognition system is proposed in this thesis in an attempt to facilitate this kind of communication. In order to build this system, we created a dataset of 40 sentences using an 80-word lexicon. It is intended to make this dataset publicly available to the research community. In the dataset, hand movements and gestures are captured using two DG5-VHand data gloves. Next, as part of data labeling in supervised learning, a camera setup was used to synchronize hand gestures with their corresponding words. Having compiled the dataset, low-complexity preprocessing and feature extraction techniques are applied to eliminate the natural temporal dependency of the data. Subsequently, the system model was built using a low-complexity modified k-Nearest Neighbor (KNN) approach. The proposed technique achieved a sentence recognition rate of 98%. Finally, the results were compared in terms of complexity and recognition accuracy against sequential data systems that use common complex methods such as Nonlinear AutoRegressive eXogenous models (NARX) and Hidden Markov Models (HMMs).

Search Terms: Arabic Sign Language, Pattern Recognition, Sequential Data, Data Gloves, k-Nearest Neighbors (KNN), NARX, Hidden Markov Models (HMMs).

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1: Introduction

This chapter introduces sign language recognition and highlights its importance. It also presents pertinent background information and a literature review on the recent advances in sign language recognition. Toward the end of this chapter, the main contribution is briefly described, and finally, the organization of this thesis is outlined.

1.1 Sign Language Recognition

A series of manual and non-manual gestures such as hand movements and facial expressions indicating words, are referred to as sign language. It is a form of communication used mostly by people with impaired hearing.

Sign language recognition systems are used to convert sign language into text or speech to enable communication with people who do not know these gestures. Usually, the focus of these systems is to recognize hand configurations including position, orientation, and movements. Generally, there are three levels of sign language recognition: finger spelling (alphabets), isolated gestures (single gesture or word), and continuous gesturing (sentences).

Accordingly, these configurations are captured to determine their corresponding meanings, using two approaches: sensor-based and vision-based. While the former entails wearable devices to capture gestures, it is usually simpler and more accurate. On the other hand, vision-based approach utilizes cameras to capture the sequence of images. Although, the latter is a more natural approach, it is usually more complex and less sensitive.

In this thesis, we constructed a sensor-based continuous Arabic sign language (ArSL) dataset and proposed a framework to optimize its recognition accuracy.

1.2 Literature Review

Gesture recognition has many applications ranging from sign language recognition to virtual reality. As a result, many research studies were conducted in this area.

Two main approaches are used for acquiring gestures. The first approach is vision-based, which provides more freedom to the user by accommodating natural interactions. However, there are some drawbacks in using this approach, such as varying captured data background, light intensity in the environment, and the computational requirement of its image processing techniques. On the other hand, sensor-based approach has simpler data acquisition and processing requirements. Moreover, it can provide more accurate readings depending on the type of its sensors. Additionally, it can be connected wirelessly to provide a more flexible movement to its user.

In a review of vision and sensor based approaches, Mohandes et al. [1] presented a summary of ArSL recognition levels in addition to the main features used in each approach as shown in Figure 1.1.

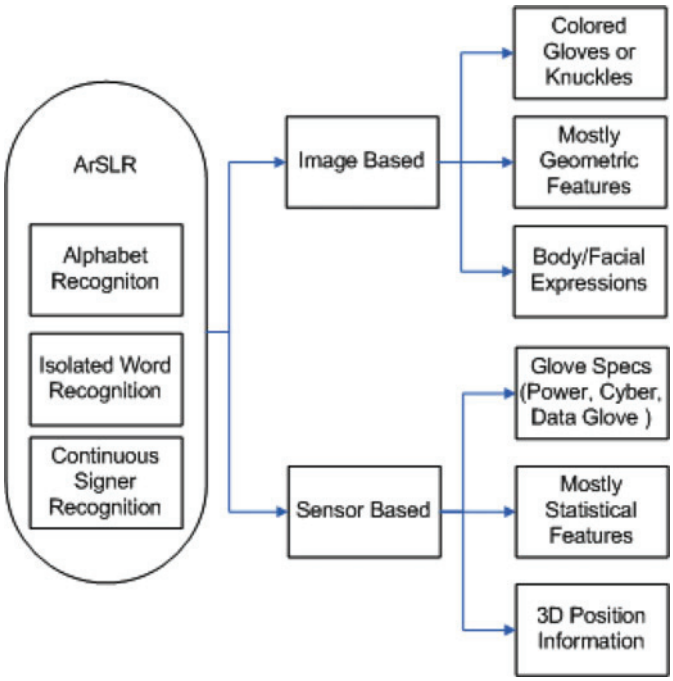


Figure 1.1: The Main ArSL Recognition Classes [1]

The topic of this work, sign language recognition, is an important application of gesture recognition with different challenges [2] such as:

- The temporal nature of the data
- The level of recognition (isolated or continuous gestures)

- The validity of the system being developed (user-dependent or user-independent)
- The required feature extraction techniques for a proper classifier

A review of the two main approaches for sign language recognition systems is presented in the following sections.

1.2.1 Vision-based systems. Many research works were conducted for Arabic sign language recognition using the vision-based approach. Referring to the three main levels, thorough research work has been done for alphabet recognition systems accomplishing high accuracies. For the next level, isolated gestures, ArSL recognition systems were developed for medium-sized datasets including less than 300 signs. On the other hand, few continuous ArSL recognition systems were proposed with limitations.

Starting with vision based alphabets, where each alphabet is represented by a still image illustrating a sign language letter, Al-Jarrah and Halawani [3] presented a system based on adaptive neuro fuzzy inference system (ANFIS) networks. For 30 Arabic manual alphabets an accuracy of 93.55% was achieved.

Using a polynomial classifier, Assaleh and Al-Rousan [4] developed an automatic ArSL recognition system using a dataset of 42 alphabets. They applied some geometric feature extraction methods on the data obtained from a colored glove. For more than 200 samples of data, 93.4% recognition rate was achieved.

Apart from using colored gloves, a more natural system based on ANFIS models was presented by Al-Jarrah and Al-Omari in [5]. They utilized a feature extraction technique based on boundary and region properties. Before training the system using the hybrid learning algorithm, two algorithms were used to identify the fuzzy inference system: the subtractive clustering algorithm and the least-squares estimator. A recognition rate of 97.5% was achieved for a dataset of 30 ArSL alphabets using 10 rules per ANFIS model and 100% for 19 rules per ANFIS model.

In order to overcome some drawbacks of vision-based approach, Abul-Ela et al. [6] presented a robust ArSL recognition system. A frontal view camera was used and

featured an extraction technique, that is suitable the large background or non-uniform lighting cases, was proposed. The maximum recognition rate they achieved was 90%. They recommended using more than one camera for better accuracy.

With the advances in Smartphones technologies, recently Elhenawy and Khamiss [7] proposed a system, called AndroSpell, for vision-based posture recognition. Their prototype was built on the top of the cameraphone. As a beginning, they tested the system with 10 postures and showed an accuracy of 97%.

For the second recognition level, an isolated gesture is represented by a sequence of images. Hence, additional processing is required to deal with the temporal dependencies.

Using hidden Markov models, Al-Rousan et al. [8] built an automatic isolated ArSL recognition system. An overview of the system is given in Figure 1.2. For a dataset of 7860 samples from 30 isolated words, an online mode recognition accuracy of 93.8% was obtained for user-dependent mode and 90.6% for user-independent mode.

An innovative feature extraction technique based on the concept of accumulated differences was proposed by Shanableh et al. [9]. This technique was followed by a discrete cosine transform and zonal coding to eliminate the temporal dependency of the data. Accordingly, lower complexity classification approaches can be used such as KNN and polynomial classifiers. Using this technique, Shanableh and Assalah [10] built an isolated ArSL recognition system with unrestricted clothing or image background. The obtained average classification rate was 87% for a dataset of 23 Arabic words performed by three different signers. Other variations of feature extraction techniques based on motion estimation and motion vectors were proposed in [11], [12].

In order to provide a more practical system for deaf people, a reliable continuous gesture recognition system is required.

In an effort to recognize continuously signed ArSL gestures, Assaleh et al. [13] presented a system based on HMMs and spatio-temporal feature extraction, proposed by Shanableh et al. in [9]. A dataset constituting of 40 sentences without imposing

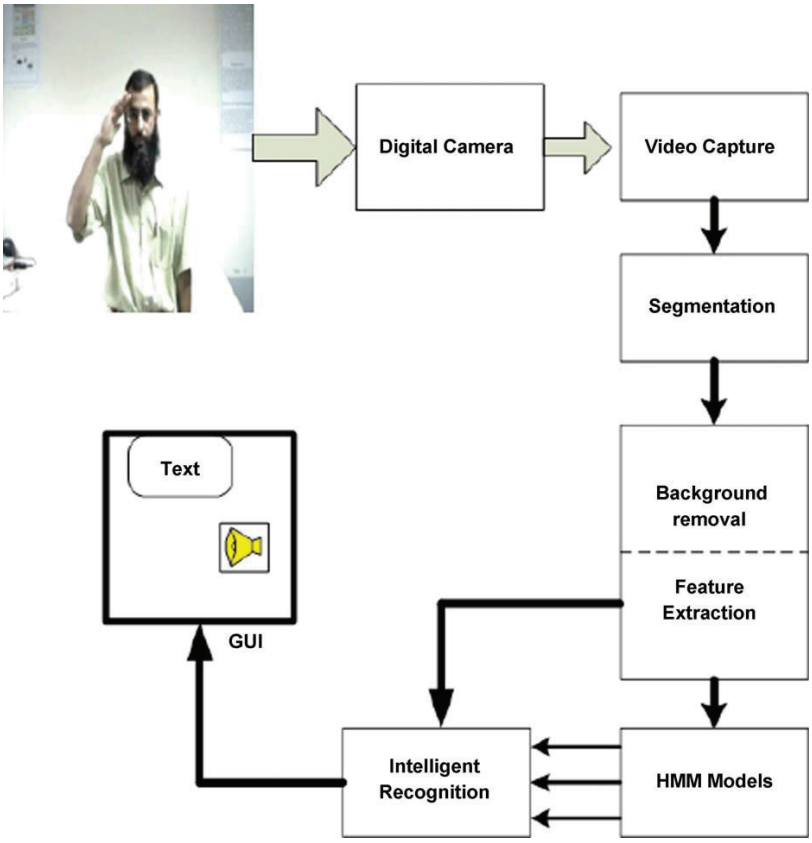


Figure 1.2: An Overview of Isolated ArSL Recognition System [8]

grammar on the sentence structure was formed using 80 words. The achieved word and sentence recognition rates were 94% and 75% respectively.

Other vision-based recognition systems were proposed for other sign languages around the world including American sign language [14]-[16], Persian sign language [17], German sign language [18], and Polish sign language [19].

1.2.2 Sensor-based systems. Sensor-based recognition systems depend on instrumented gloves to acquire the gesture's information. Generally, equipped sensors measure information related to the shape, orientation, movement, and location of the hand. Accordingly, the selection of sensors is essential as it has a substantial effect on the following stages in the recognition system.

Starting with alphabet recognition, Bui and Nguyen in [20] used microelectronic mechanical system (MEMS) accelerometers to develop a data glove for posture recognition in Vietnamese sign language (VSL). The proposed glove contains 6 ADXL202 sensors; one for each finger and another on the back of the palm. Readings obtained from these sensors are combined into one vector with 12 measurements (two axes per sensor). Acquired raw data is transformed into relative angles between the fingers and the palm. Next, to start classification, letters are divided into groups according to the palm sensor readings as illustrated in Figure 1.3.

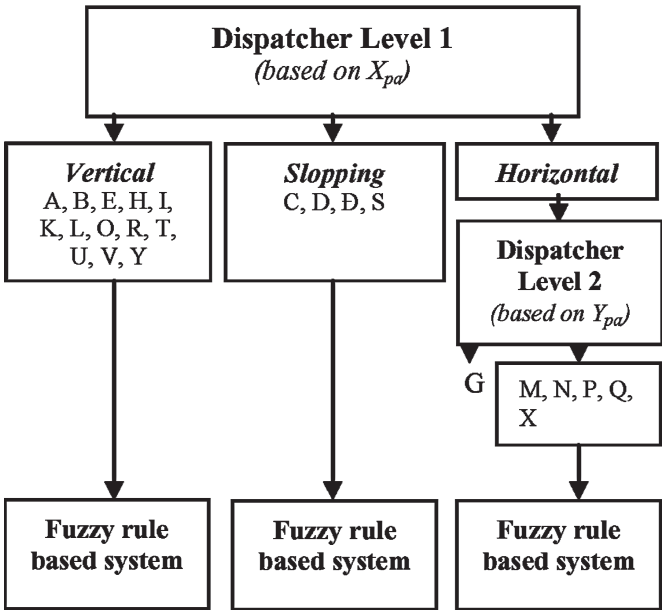


Figure 1.3: An Overview of VSL Classification System [20]

Then, the relative angle degrees between each finger and the palm were modeled into five levels of bending. Static position of the hand is a condition to start the recognition process for any posture. This was verified by comparing the current sensor readings with the previous one. Afterwards, 22 fuzzy rules were developed for character recognition. Finally, all fingers' degrees were calculated to form a data set and find the best match with one of the fuzzy rules. The system used 23 Vietnamese letters with 20 of them being recognized perfectly. The remaining 3 letters are approximately similar;

therefore, to improve their recognition accuracy, some Vietnamese spelling rules were applied.

For Arabic sign language, several isolated gesture recognition systems were proposed using off-the-shelf devices.

Using Power Glove, Mohandes et al. [21], [22] developed an automatic ArSL recognition system using statistical features and a Support Vector Machine (SVM) classifier for a dataset of 120 words. Their design is cost-effective; however, it has some limitations due to the type of data from the glove that has inexact controls.

On the other hand, CyberGlove provides high-accuracy joint-angle measurements as shown in Figure 1.4. For sign language application, information about hand movement and orientation is required; however, these measurements represent fingers' movements and bending only. Therefore, hand-tracking devices are commonly used with this CyberGlove to complement these information.

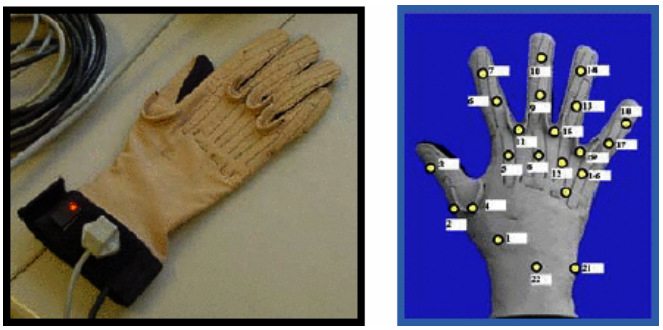


Figure 1.4: CyberGlove Sensors [23]

In [24], CyberGloves and two hand-tracking devices were used to collect a dataset of 100 two-handed ArSL signs with 20 samples for each gesture. From each hand, 22 joint-angle measurements were reported from the Cyberglove. Additionally, the hand location (x, y, z) and orientation (roll, pitch, yaw) measurements were obtained from the hand-tracking device with reference to a fixed point. Next, in an attempt to recognize isolated gestures, the author used PCA to reduce the dimensionality of the second-order statistics features obtained from the sub frames of the signs. Then, SVM

was used with 75% of the dataset for training and 25% for testing to achieve an accuracy of 99.6%.

It is from here and with using the same dataset that Mohandes and Deriche [23] separated the features obtained from the CyberGlove and the hand-tracking system to test the effect of fusing their features in different levels. First, they tested the system with the separated features to get 84.7% from the hand-tracking system features and 91.3% from the CyberGlove system. Next, a traditional feature-based fusion of the two systems was examined and an accuracy of 96.2% was achieved. Finally, both features were combined at the decision level using the Dempster-Shafer Theory of Evidence as shown in Figure 1.5, where FOB refers to the name of the tracking device (Flock of Birds). They showed that fusion at the decision level is an effective approach as it lead to an accuracy of 98.1%.

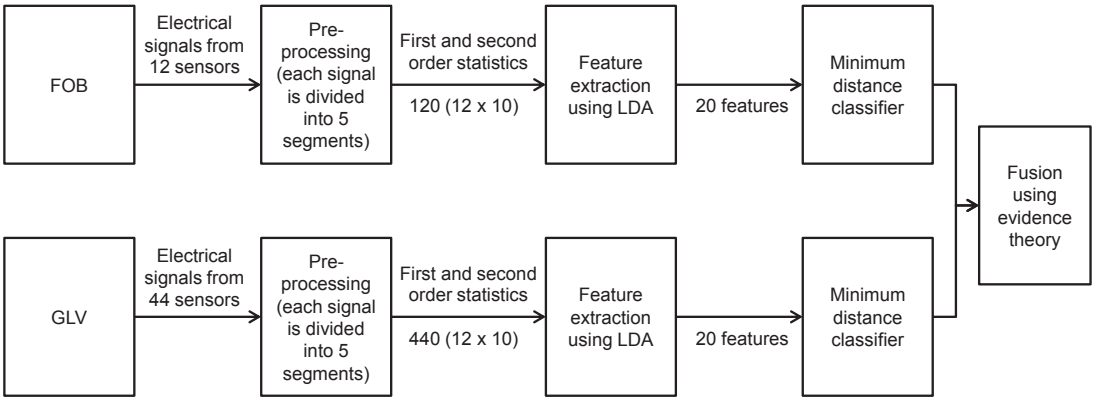


Figure 1.5: An Overview of ArSL Recognition by Decisions Fusion using Dempster-Shafer Theory of Evidence [23]

The requirement of using hand trackers makes Cyberglove a non-ideal option for sign language recognition. Therefore, DG5-VHand Gloves were found to be a better selection for this application.

In [25], Assaleh et al. proposed a low-complexity classification system based on a method of accumulated differences to eliminate the temporal dependency in ArSL data. The system was designed for isolated gesture recognition using two DG5-VHand

2.0 data gloves. This glove measures the amount of bend in each finger; therefore, five measurements were reported. Moreover, it measures the hand orientation and movement using a 3D accelerometer. From this data, features were extracted using an approach based on the accumulated differences and statistical parameters. Next, stepwise regression was applied and k-Nearest-Neighbor (KNN) classifier was used. The achieved recognition rates were 92.5% and 95.3% for user independent and user dependent modes respectively.

While some sensor-based continuous recognition systems were developed for American and Chinese sign languages, research on continuous ArSL is still limited to vision-based systems.

Recently, Kong and Ranganath [26] proposed a segment-based probabilistic method for continuous American sign language (ASL) recognition. They used one Cyberglove with three polhemus trackers to form a dataset of 74 single-handed sentences of 107 sign vocabulary. Their approach is based on segmenting a continuous sentences into meaningful signs (SIGN) and movement epentheses (ME) using a Bayesian network. These MEs are then discarded and SIGNs sub-segments are merged. For the recognition model, a two-layer conditional random field (CRF) classifiers were proposed using a semi-Markov CRF decoding scheme as illustrated in Figure 1.6. Their signer independent system achieved a recall rate of 86.6% and 89.9% precision.

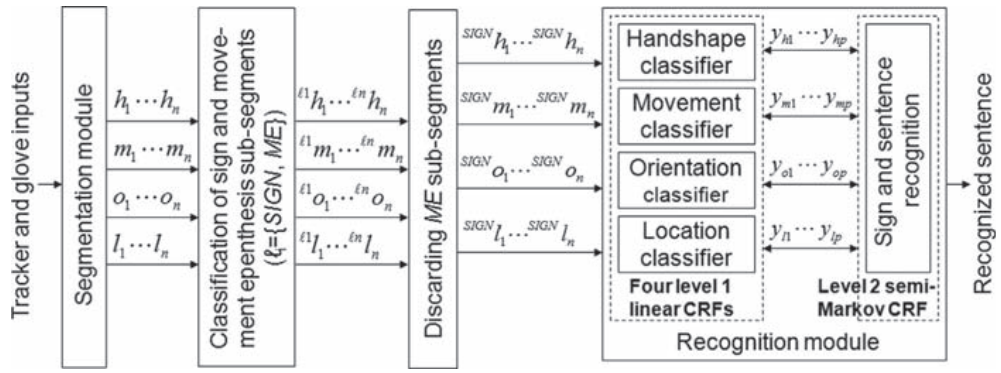


Figure 1.6: ASL Recognition System using Segment and Merge Approach [26]

Gao et al. in [27] developed user-independent Chinese sign language recognition system for both isolated and continuous signs. Two Cybergloves with 18 sensors each and three polhemus 3SPACE-position trackers were used as input devices. As the proposed framework was modeled for different signers, the self-organizing feature maps (SOFM) technique was used to extract important features with reduced dimensionality. Then, hidden Markov models (HMMs) were used for isolated signs recognition. Using this model, an accuracy of 82.9% was achieved for 5113 isolated signs. Afterwards, for continuous SLR, where the start and end points of meaningful gestures must be determined, an improved simple recurrent network (SRN) was employed to perform automatic data segmentation before recognition. In addition, SOFM enhances separation between meaningful signs and movement epentheses where distinct fluctuation will appear. Therefore, severe fluctuations located close to SRN segmentation points can be referred to as moving epentheses and discarded during implementation. SRN has three encoded outputs to identify left, right and interior of a segment as follows:

- $[1\ 0\ 0]$ Output is the left boundary of a segment
- $[0\ 1\ 0]$ Output is the right boundary of a segment
- $[0\ 0\ 1]$ Output is the interior of a segment

Finally, SRN outputs were utilized as HMM states where the Lattice Viterbi algorithm was employed to optimize the word sequence searching criteria. Training parameters obtained from SOFM/HMM models for isolated sign recognition were used by this algorithm as computing models for sign candidates. The structure of SLR system based on SOFM/SRN/HMM is shown in Figure 1.7. For a dataset of 400 continuous Chinese sign language sentences collected from 3 different signers, the obtained recognition rate was 86.3%.

Another continuous Chinese sign language (CSL) recognition system was proposed by Zhang et al. in [28]. In this work, they used one 3D accelerometer (ACC) and five electromyographic (EMG) sensors placed as shown in Figure 1.8. ACC was used to complement the functionality of EMG sensor and overcome some of its problems that are related to the muscles' physiological nature and tiredness effects. As mentioned

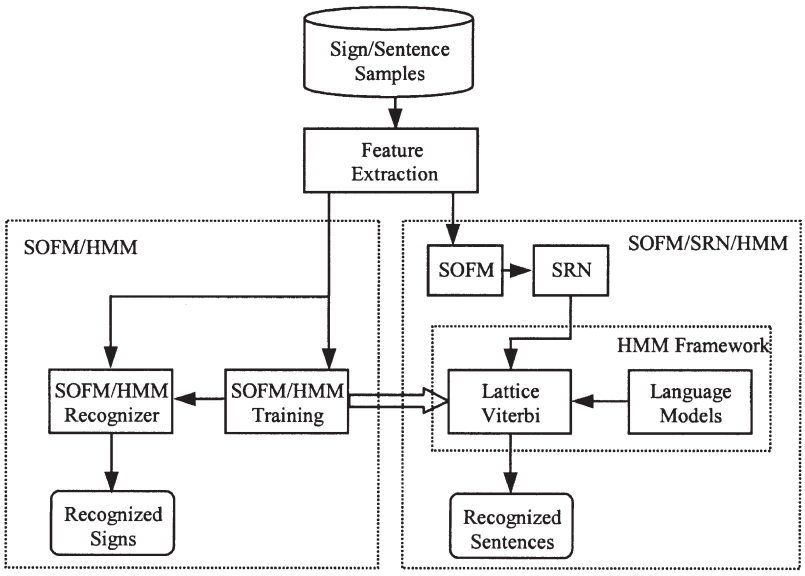


Figure 1.7: The Structure of SLR System Based on SOFM/SRN/HMM [27]

earlier, data segmentation is an important stage in continuous sign language recognition process. Therefore, an automatic detection technique based on the intensity of the electromyographic signals was used to detect the start and end of a meaningful sign. Next, segments' boundaries were determined by computing the instantaneous energy and moving average for the EMG channels. Then, using two threshold values for the onset and the offset, meaningful segment starts when the averaged energy stream is more than the onset threshold and lasts for a minimum of $100\ ms$ before its value goes below the offset threshold. According to these boundaries, ACC data was measured. Finally, a decision tree was used before applying a multi-stream hidden Markov models (MSHMM) to improve accuracy of the system. As a result, 93.1% word accuracy and 72.5% sentence accuracy were reported for 72 single-handed words forming 40 sentences and performed by two right-handed signers.

Research on sensor-based recognition systems has been conducted for other sign languages including Australian sign language [29], Korean sign language [30], and Taiwanese sign language [31].

For sign language recognition applications, in particular, it is important to consider the number of sensors it has, their type, and the accuracy of their readings. Among

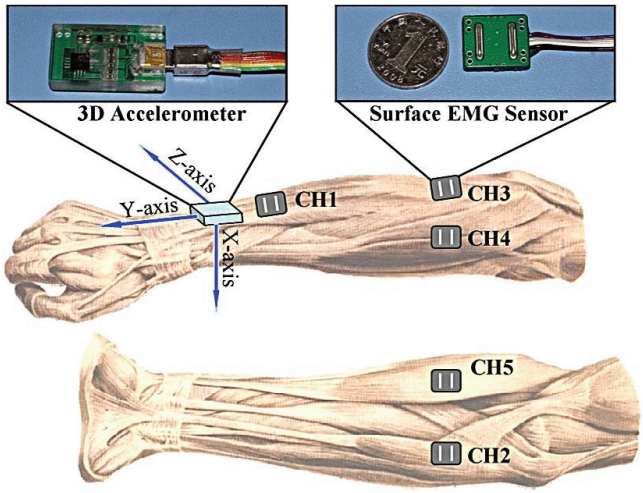


Figure 1.8: Sensor Placement of 3D-ACC and Multichannel EMG [28]

other types of gloves, the DG5-VHand 2.0 data glove was found to be more suitable for this application. Recently, DGTech Engineering Solutions released a newer version of this glove (Figure 1.9), which provides more accurate measurements and is based on Wi-Fi communication.

Figure 1.9: DG5 VHand Glove 3.0 [32]

1.3 Background

This section describes two classification techniques that pertain to this work. It details the k-Nearest Neighbors (KNN) classifier first, and then describes the hidden Markov models (HMMs).

1.3.1 k-Nearest Neighbor (KNN). In order to handle arbitrary distributions and overcome the problem of finding the best estimate of a probability density function in pattern recognition, nonparametric methods were introduced. The k-nearest neighbor rule is one of these methods that works directly on decision functions. As such, it estimates the $a\ posteriori$ probabilities $P(l_j|x)$, where l_j is an estimated label of a given instance x. Therefore, KNN is a function of the training data [33].

The k-nearest neighbor rule employs a predefined distance metric, which represents a function to calculate the distance between a given test pattern and the training data patterns. The choice of such function depends on the nature of the data. A commonly used metric is the L^p-norm or Minkowski metric as shown in Equation 1.1.

$$L^p(a,b) = \left(\sum_{i=1}^{d} |a_i - b_i|^p \right)^{1/p} \tag{1.1}$$

This is a general formula to calculate the Minkowski distance between two data points. It can be extended to calculate the distance between two vectors, which gives the following special cases according the value of p:

- $p = 1$, ℓ_1 norm or Cityblock distance.
- $p = 2$, Euclidean distance.
- $p = \infty$, infinity norm or Chebyshev norm.

The k-nearest neighbor is then classifies any unknown test instance by calculating the distance to all the instances in the training set. Finally, it uses the majority rule to vote for the most frequent class of the k nearest classes [34], [35].

Since the only requirement to train the system is to memorize the training data, k-nearest neighbor is considered as an instance-based learning classifier.

For a dataset with two classes, the event probability that the majority label l_m will be assigned to an instance x is:

$$\sum_{j=(k+1)/2}^{k} \binom{k}{j} P(l_m|x)^i [1 - P(l_m|x)]^{k-j} \tag{1.2}$$

where $P(l_m|x)$ is the $a\ posteriori$ probability of l_m given x.

The best probability of choosing l_m can be achieved when providing an infinite training set. In this case, for a reliable estimate, k must be large. However, this is not possible in practice; therefore, for a finite training set, k must be selected so that it optimizes this probability.

In summary, kNN is a simple and effective classifier with a low computational complexity of $O(N)$, where N is the number of training instances. The classification time for a single instance is proportional to N. The classification time will be affected in cases where very large training data is used. Therefore, tree-based approaches are used to represent the training set, and subsequently improve the efficiency. Moreover, using KNN can improve the system at any time by adding more representative samples to the training set.

1.3.2 Hidden Markov Models (HMMs). Hidden Markov model (HMM) is a time-domain process used as a machine learning tool for data representing a sequence of observations over time [36], [37]. The word 'hidden' comes from the fact that Markov model is used to retrieve a sequence of 'hidden' states from a sequence of observations. As an example, HMM is used to recognize the set of words corresponding to a recorded speech signal.

For a set of N states $S = \{s_1, s_2, ..., s_N\}$ and a sequence of T observations over time t, $O = \{o_1, o_2, ..., o_T\}$ such that $O \in S$ [38]. Two Markov assumptions are defined to manage the time series data:

- The limited horizon assumption, where the probability of being in state s_t depends only on the previous state s_{t-1}, i.e., each state in the model has sufficient information to predict the next state:

$$P(o_t|o_{t-1}, o_{t-2}, ..., o_1) = P(o_t|o_{t-1}) \tag{1.3}$$

- The stationary process assumption, where the probability of being in a state s_t given the previous state s_{t-1} is fixed:

$$P(o_t|o_{t-1}) = P(o_2|o_1), \quad for\ t \in 2, 3, ..., T \tag{1.4}$$

Having these two assumptions in mind, we define the state transition matrix A as shown in Equation 1.5:

$$A = \{a_{ij}\}, \quad for\ i, j = 1, 2, ..., N \tag{1.5}$$

such that a_{ij} represents the transition probability from the current state s_i at time t to the next state s_j at time $t + 1$.

By setting the parameters of the Markov model, the probability of any sequence of states, illustrated in Equation 1.6, can be computed. In addition, the maximum likelihood transition matrix $\hat{A}$ of that observed sequence can be estimated using Equation 1.7 to calculate the maximum likelihood probability for every s_i to s_j transition.

$$P(O) = P(o_t, o_{t-1}, ..., o_1|A) = \prod_{t=1}^{T} P(o_t|o_{t-1}; A) = \prod_{t=1}^{T} A_{o_{t-1}o_t} \tag{1.6}$$

$$\hat{A}_{ij} = \frac{\sum_{t=1}^{T} 1\{o_{t-1} = s_i \wedge o_t = s_j\}}{\sum_{t=1}^{T} 1\{o_{t-1} = s_i\}} \tag{1.7}$$

where I in the numerator and denominator represents an indicator function that has a value of one if the given condition is true, otherwise its value is zero.

The implementation of a Markov model when the observation of the actual states is not possible, but rather some probabilistic functions that represents the outcomes of these states is named as the 'hidden' Markov model. Accordingly, for the same notation used for Markov models described above, $O \notin S$ where the set of observations are not as the actual states anymore.

Besides, a model for the probability of the observation B is generated. Each element in the matrix B, annotated as b_{jk}, represents the probability of getting the observation o_k from hidden state s_j.

Now, for a given data with finite number of hidden states and set of observation sequences, the hidden Markov model calculates the parameters A and B to output the probability of any observed sequence and the maximum likelihood state assignment. To reduce the computational complexity of the system, the Baum-Welch and Viterbi algorithms were used as detailed in [38].

To summarize, the objective of HMMs is to determine the most likely series of actual (hidden) states for a given series of observations.

The drawback of using HMMs is their high computational complexity. In spite of using the forward procedure and Veterbi algorithm, the computational complexity reduced from $O(N^T)$ to $O(N \cdot T)$ only. Furthermore, multiple runs might be needed for HMM parameters to converge. Additionally, when applied to gesture recognition, the number of states per gesture and the number of Gaussian mixtures, required to model the states, should be determined empirically. These two parameters greatly affect the performance of HMMs, hence rendering the configuration of the tool for gesture recognition is very difficult, indeed.

1.4 Thesis Objectives and Contribution

Gesture recognition is a term that represents meaningful movements of the human head, body, hands, arms, and/or face [36]. Sign language is one of the applications

of gesture recognition that includes manual signs and non-manual signs (NMS) such as facial expressions. However in Arabic sign language, non-manual signs are rarely used.

In most of the Arabic speaking countries, deaf people use Arabic sign language (ArSL) to communicate with other members of their society. Due to the fact that most of the hearing community members do not understand Arabic sign language, deaf people have limited activities in the society. Subsequently, in order to help people with hearing disabilities to use the available educational and vocational opportunities, an automatic ArSL recognition system is a great help for those who use sign language to communicate with those who do not know it.

From the two types of sign language recognition systems, the proposed work is based on acquiring continuous gestures from two instrumented data gloves. One major advantage of using a sensor-based technique is the type of its readings that are generally accurate and not sensitive to the background motion, clothing, or illumination.

As the type of gesture's data is sequential and has temporal dependency, a special kind of system modeling is required. Typically, hidden Markov models (HMMs) are used since they provide good modeling for this kind of data. However, they require heavy computations and their tools are complicated and difficult to configure. These tools are originally implemented for speech recognition not gesture recognition. Therefore, we propose an easy to implement and configure recognition system for sequential data. The proposed system employed low-complexity feature extraction and recognition techniques and achieved recognition rates that outperformed those obtained using other sequential data models. The following points summarize the contribution of this thesis:

- Proposing a framework for continuous Arabic sign language recognition using a sensor-based approach in order to facilitate the communication between deaf and hearing individuals.
- Creating the first fully-labeled continuous ArSL dataset, which we intend to make available to the research community.

- Employing a statistical feature extraction technique to fetch representative features from the sequence-based data.
- Proposing a low-complexity classifier based on KNN that can handle time dependent data. The proposed classifier serves as an alternative for typically used classifiers such as HMMs, which involve heavy computations and usually difficult to configure.

1.5 Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2, the sensor-based continuous Arabic sign language dataset is introduced. Chapter 3 presents the proposed preprocessing and feature extraction techniques. Chapter 4 introduces the proposed classification approach that is based on KNN. Then, an evaluation of our proposed framework is discussed in Chapter 5. Finally, Chapter 6 concludes this thesis and gives directions for future work.

Chapter 2: The Dataset

This chapter explains the procedure used to create a sensor-based continuous Arabic sign language (ArSL) dataset. It begins with a list of the sentences constituting this dataset, and then moves to an overview of the system and data collection setup. Toward the end of the chapter, data labeling as well as the final data format are explained. We intend to make this fully labeled dataset publicly available for the research community.

2.1 Dataset Description

In order to build an automatic continuous Arabic sign language recognition system, a dataset is required for validation. Accordingly, an 80-word lexicon was used to form 40 sentences with unrestricted grammar and sentence length. These sentences represent common situations for hearing-impaired individuals. To the best of our knowledge, it is the first dataset for sensor-based continuous ArSL. The choice of its sentences was based on the vision-based continuous ArSL dataset introduced in [13]. Consequently, after building the proposed system, obtained results can be compared with that vision-based system.

A list of these sentences along with their English translation is given in Table 2.1. Seven of these sentences can be performed using the right hand only, whereas the remaining 33 sentences include gestures that involve both hands.

Table 2.1: List of the Dataset Arabic Sentences and Their English Translation

No.	Arabic Sentence	English Translation
1.	ذهبت الى نادي كرة القدم.	I went to the soccer club.
2.	انا احب سباق السيارات.	I love car racing.
3.	اشتريت كرة ثمينة.	I bought an expensive ball.
4.	يوم السبت عندي مباراة كرة قدم.	On Saturday I have a soccer match.
		Continued on next page

Table 2.1 – continued from previous page

No.	Arabic Sentence	English Translation
5.	في النادي ملعب كرة قدم.	There is a soccer field in the club.
6.	غدا سيكون هناك سباق دراجات.	There will be a bike racing tomorrow.
7.	وجدت كرة جديدة في الملعب.	I found a new ball in the field.
8.	كم عمر أخيك؟	How old is your brother?
9.	اليوم ولدت أمي بنتاً.	My mom had a baby girl today.
10.	أخي لا يزال رضيعا.	My brother is still breast feeding.
11.	إن جدي في بيتنا.	My grandfather is at our home.
12.	اشترى ابني كرة رخيصة.	My kid bought an inexpensive ball.
13.	قرأت أختي كتاباً.	My sister read a book.
14.	ذهبت أمي الى السوق في الصباح.	My mother went to the market this morning.
15.	هل أخوك في البيت؟	Is your brother home?
16.	بيت عمي كبير.	My uncle's house is big.
17.	سيتزوج أخي بعد شهر.	In one month my brother will get married.
18.	سيطلق أخي بعد شهرين.	In two months my brother will get divorced.
19.	أين يعمل صديقك؟	Where does your friend work?
20.	أخي يلعب كرة سلة.	My brother plays basketball.
21.	عندي أخوين.	I have two brothers.
22.	ما اسم أبيك؟	What is your father's name?
23.	كان جدي مريضاً في الأمس.	Yesterday my grandfather was sick.
24.	مات أبي في الأمس.	Yesterday my father died.
25.	رأيت بنتاً جميلةً.	I saw a beautiful girl.
26.	صديقي طويل.	My friend is tall.
27.	أنا لا آكل قبل النوم.	I do not eat close to bedtime.
28.	أكلت طعاما لذيذا في المطعم.	I ate delicious food at the restaurant.
29.	أنا أحب شرب الماء.	I like drinking water.

Continued on next page

Table 2.1 – continued from previous page

No.	Arabic Sentence	English Translation
30.	أنا أحب شرب الحليب في المساء.	I like drinking milk in the evening.
31.	أنا أحب أكل اللحم أكثر من الدجاج.	I like eating meat more than chicken.
32.	أكلت جبنة مع عصير.	I ate cheese and drank juice.
33.	يوم الأحد القادم سيرتفع سعر الحليب.	Next Sunday the price of milk will go up.
34.	أكلت زيتوناً صباح أمس.	Yesterday morning I ate olives.
35.	سأشتري سيارة جديدة بعد شهر.	I will buy a new car in a month.
36.	هو توضأ ليصلي الصبح.	He washed for morning prayer.
37.	ذهبت إلى صلاة الجمعة عند الساعة العاشرة.	I went to Friday prayer at 10:00 o'clock.
38.	شاهدت بيتاً كبيراً في التلفاز.	I saw a big house on TV.
39.	في الأمس نمت عند الساعة العاشرة.	Yesterday I went to sleep at 10:00 o'clock.
40.	ذهبت الى العمل في الصباح بسيارتي.	I went to work this morning by my car.

For this thesis, one user performed 10 repetitions for each sentence, yet the dataset was built in a way that allows room for more sentences and repetitions. The following sections describe the data collection and labeling procedures.

2.2 Data Collection

For the proposed sensor-based continuous ArSL recognition system, two DG5-VHand 2.0 data gloves were used. Among other types of gloves, the DG5-VHand glove was chosen mainly because of its sensors' resolution and Bluetooth interface. As such, it is suitable for the proposed application, which requires an efficient number of sensors with accurate readings. In addition, Bluetooth communication in the system minimizes the imposed movement limitation, which usually arises in sensor-based systems. In the literature, similar gloves were used for collecting isolated sign language gestures [25]. More specifications of this version of the data glove are summarized in Table 2.2.

Table 2.2: DG5-VHand Glove 2.0 Specifications

Number of Sensors:	· 5 proprietary flex sensors—for high stability · 3 degrees of integrated tracking
Resolution:	10 bit, 1024 position per finger
Output Interface:	· Platform independent USB or Wireless Bluetooth Interface · High update rate · On-board processor (20 MHz)
Software Bundled:	· Complete C++ SDK with program samples · Bundled software (Control Panel, Mouse Emulation, Sound Glove) · Updatable firmware · PDA support (for off-line MOCAP applications)
Package Includes:	· Glove with the sensors · Embedded Control board · Embedded Tracker · Manual · USB cable or USB/Bluetooth cable · CD ROM containing the drivers and the software suite
Price (per hand):	DG5-VHand Glove USB, Right/Left Handed - $585.00 DG5-VHand Glove Wireless - Bluetooth, Right/Left Handed - $750.00

DG5 VHand 2.0, Figure 2.1, has a TTL to Bluetooth adapter with an external battery. Consequently, a PC with a Bluetooth connection is needed for communication and data collection. In addition, for the purpose of manual data labeling, which is necessary for the training phase, a digital camera was used to record a video segment for each repetition. This data collection setup is illustrated in Figure 2.2.

Figure 2.1: DG5-VHand Glove 2.0 [39]

The selected data glove has five embedded bend sensors and one 3-axes accelerometer as shown in Figure 2.3. Bend sensors, also known as flex sensors, measure

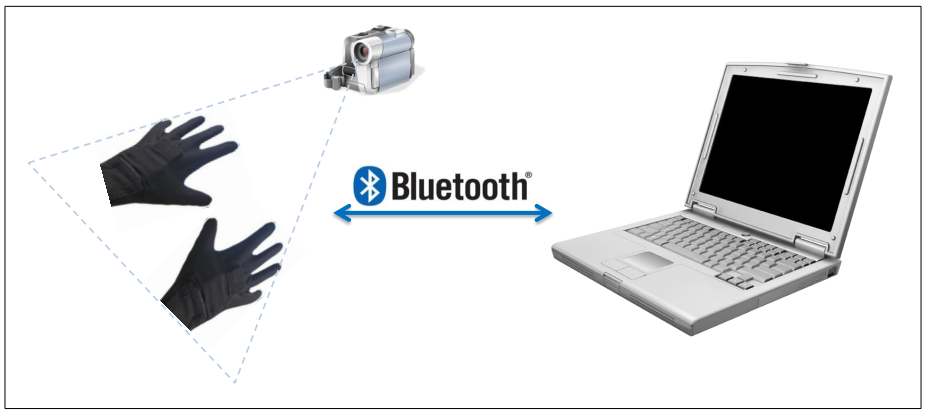


Figure 2.2: The Data Collection Elements

the amount of bend on the sensor and change it into electrical resistance. As the amount of bend increases, the resistance value increases.

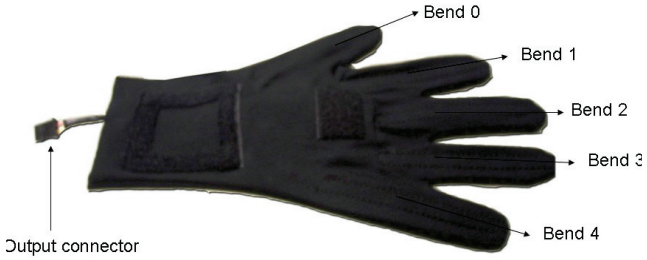


Figure 2.3: DG5-VHand 2.0 Sensors [39]

Hand movement measurements provided by the 3-axes accelerometer, include both dynamic and static accelerations. While the dynamic acceleration along the three main axes represents the instantaneous hand movement as shown in Figure 2.4(a), the static acceleration gives roll and pitch rotational angles as demonstrated in Figure 2.4(b). Static acceleration is caused by the gravity force. For instance, if the hand was rotated along the x axis (pitch), then y static measure, or the gravity component, will be changed. As a result, pitch angle can be extracted from y static acceleration.

The five flexion measurements for thumb, index, middle, ring, and little fingers are reported as bend percentages in the range (0.0% - 100.0%). Furthermore, hand orientation is reported as two rotational angles for roll and pitch in the range ($-90°$ to $+90°$). The last three readings represent the hand's instantaneous acceleration for x,

(a) Axes Reference

(b) Roll and Pitch Angles

Figure 2.4: Illustration of DG5-VHand 2.0 Axes Reference and Hand Orientation

y, and z axes with values between $-2g$ and $+2g$. Eventually, ten measurements, or features, will be reported with the ranges summarized in Table 2.3.

Table 2.3: DG5-VHand 2.0 Ranges of Measurements

Data Type	**Range**
Flexion measurement	0.0% to 100.0%
Roll and pitch rotational angles	$-90°$ to $+90°$
x, y and z axes	$-2g$ to $+2g$

To establish communication with the glove, *DataGlove manager* software, Figure 2.5, was used. It provides the option to visualize and save collected data in a file for further processing. The acquired data file is saved in Microsoft Excel Worksheet (.xls) with each row containing a time stamp in milliseconds, five flexion measurements, two hand orientation angles, and three instantaneous acceleration measurements as follows:

Time	Thumb	Index	Middle	Ring	Little	Roll	Pitch	Ax	Ay	Az

The right hand glove file is reported with the above sequence, whereas the left hand glove follows a slightly different order for the flex sensors' readings as given below:

Time	Little	Ring	Middle	Index	Thumb	Roll	Pitch	Ax	Ay	Az

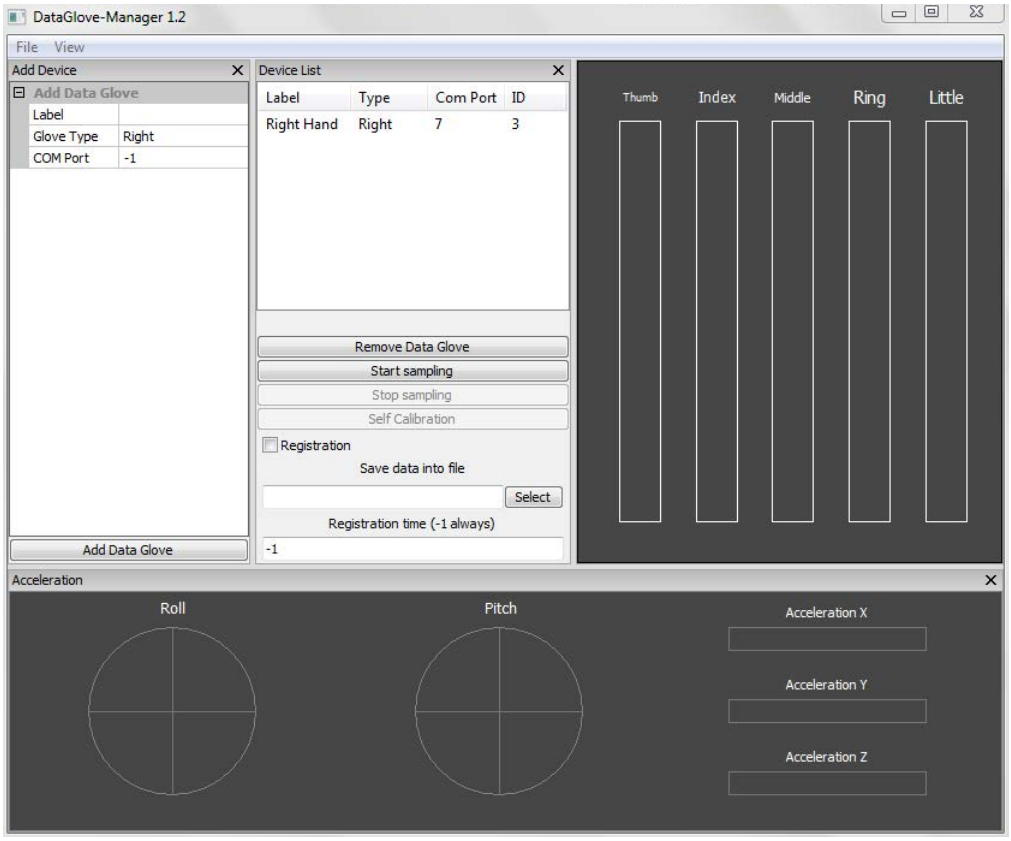


Figure 2.5: DataGlove Manager Graphical User Interface

For single-handed sentences, readings are collected from the right hand glove only. As a result, ten features are reported, whereas sentences that use both hands generate twenty features combined from right and left hand gloves.

2.3 Labeling

As an essential step in supervised learning, labeling is required to set classes for glove readings and prepare them for the training and modeling stage. In continuous ArSL systems, sentences are composed of several gestures. Hence, boundaries must be located carefully to mark successive gestures. Therefore, a video segment was recorded for each repetition to carry out manual labeling.

The transition from the ending of one sign to the beginning of the next one is called movement epenthesis (ME). It connects adjacent gestures but does not hold

significant information. In the literature, MEs were handled in three different approaches. First, while considering meaningful gestures, some researchers ignored MEs completely. Therefore, sentences were handled as concatenated words. However, this segmentation violates the purpose of natural continuous gesturing. On the other hand, in the second approach, some scholars labeled MEs explicitly as in [26]. The third approach considered them as part of their adjacent gestures. In this work, the last approach was adopted by holding a consistent process of finding time boundaries between adjacent gestures for all repetitions. Next, time boundaries obtained from the video segments were synchronized with the sensor readings using time stamp field in their corresponding Excel files.

For single-handed sentences in Table 2.4, twenty repetitions for each sentence were performed by one user. Accordingly, twenty Excel files were registered from the right hand glove for each sentence. The resultant labeled files with ten features and their corresponding labels were saved in new comma separated values files (csv) for further processing.

Table 2.4: Single-Handed Sentences

Arabic Sentence	**English Translation**
عندي أخوين.	I have two brothers.
ما اسم أبيك؟	What is your father's name?
كان جدي مريضا في الأمس.	Yesterday my grandfather was sick.
مات أبي في الأمس.	Yesterday my father died.
رأيت بنتاً جميلةً.	I saw a beautiful girl.
أكلت زيتونا صباح أمس.	Yesterday morning I ate olives.
كم عمر أخيك؟	How old is your brother?

For the rest of the sentences from Table 2.1, ten repetitions for each sentence were performed by one user. In this case, ten Excel files were registered from each hand. Next, for each repetition, labeling was carried out separately for the right and

left hand depending on their time stamps. Then, these two files were combined to have a total of 20 features and a label associated with them. It has been observed that the sampling rate is non uniform; therefore, right and left hand files were synchronized by deleting unmatched labels. This deletion has a negligible effect as the sampling rate of $30\ Hz$ is relatively high. Finally, 10 files for each repetition with 20 features and a label were saved in csv files. The number of rows in the file depends on the length of its sentence. A summary of this data labeling process is given in Figure 2.6.

After manual labeling, a MATLAB code was written to save these csv files into a cell array of feature vectors (*FVs*). As such, each repetition r of sentence s was saved in a cell *FVs*$\{s\}\,\{r\}$ as shown in Equation 2.1.

$$FVs\,\{s\}\,\{r\} = [FV_1 \;\; FV_2 \;\; \cdots FV_T]^{'} \tag{2.1}$$

where the symbol ($'$) represents a matrix transpose in MATLAB to indicate that each column represents one feature vector FV_t. Accordingly, $FVs\,\{s\}\,\{r\}$ is a 2D matrix of T feature vectors where T depends on the required amount of time to capture the gestures of that repetition.

While adding the repetitions of single-handed sentences, 10 more features are augmented to their feature vectors with zero values to represent unmoving left hand readings. Subsequently, the size of any cell *FVs*$\{s\}\,\{r\}$ becomes $20\ Features$ by $SentenceLength\ (T)$.

Finally, in order to manipulate feature vectors easily, labels were saved in a different cell array *Labels*$\{s\}\,\{r\}$ such that they are assigned to their corresponding feature vectors when needed.

Any new sentence or repetition can be added simply to this cell array by providing its relevant data file(s) and controlling related parameter(s) in the code.

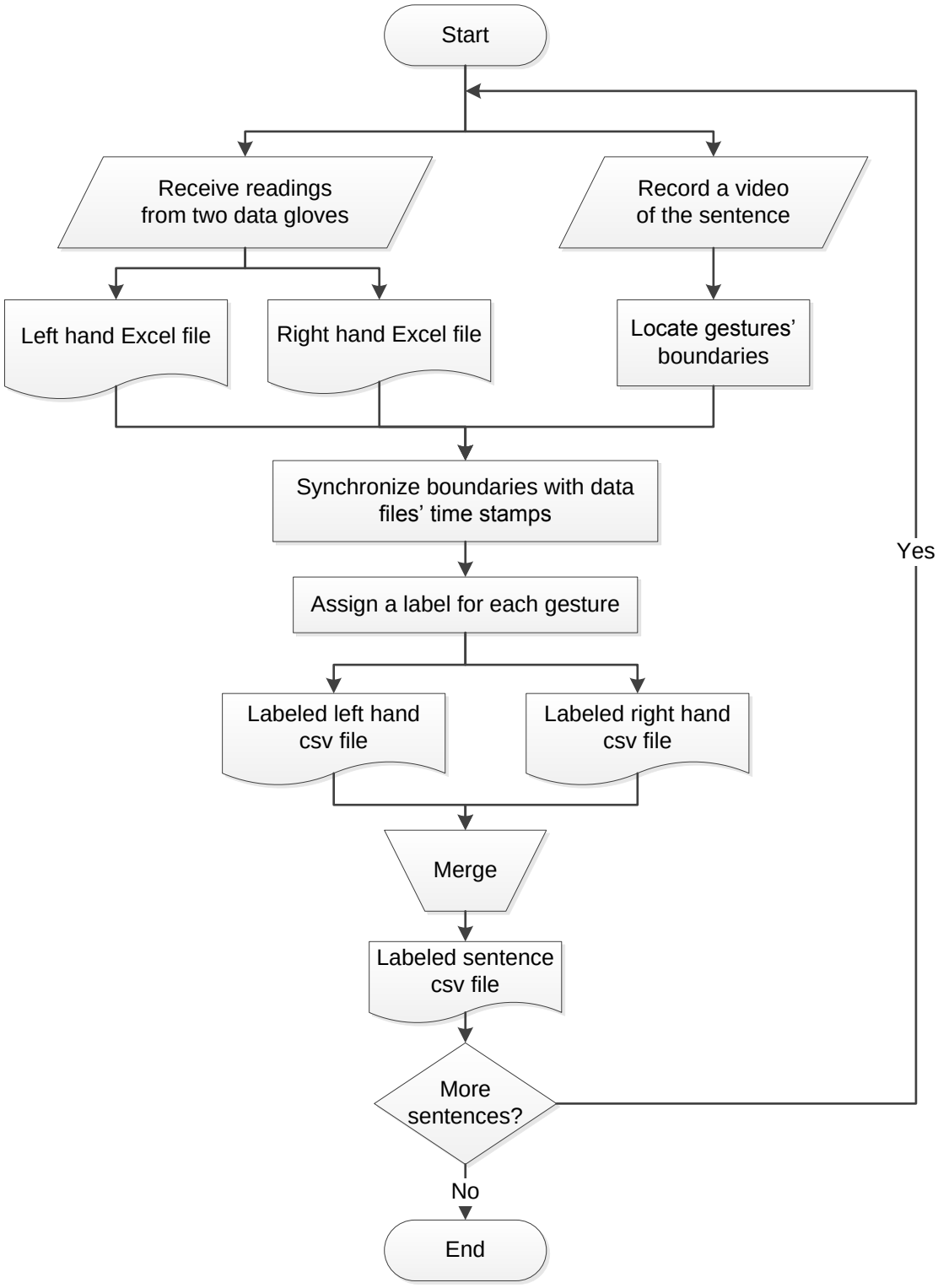


Figure 2.6: Data Labeling Flowchart

Chapter 3: Preprocessing and Feature Extraction

The previous chapter explained the approach used for data collection and labeling. This chapter proposes a number of preprocessing techniques including reducing the number of feature vectors and normalization. The chapter also introduces a number of feature extraction techniques such as window-based statistical features. Toward the end of this chapter, a brief explanation of the reviewed vision-based system [13] is provided. Accordingly, in the experimental results section we compare our work against this vision-based system.

3.1 Preprocessing

Preprocessing, from its name, is a vital stage that takes place before further processing. It prepares the data by applying different techniques such as normalization, signal denoising, resampling, and removing unnecessary data points.

3.1.1 Reducing the number of feature vectors. It has been observed that collected data files have approximately 30 readings per second. Given the physical speed of moving the hand and the fingers, this sampling rate is relatively high; therefore, some feature vectors might be redundant. Subsequently, downsampling techniques were applied to reduce the number of feature vectors and hence improved recognition time and usage of the dataset storage. Algorithm 3.1 describes the procedure used to reduce the number of feature vectors by a factor Q.

Two downsampling approaches were employed with $Q = 2$ and 3. The first approach was done using MATLAB function "`resample`" to decimate time-domain data by Q. The second method was implemented by simply keeping every Q^{th} observation from the original data sequence and discarding others. Afterwards, the length of the resultant feature vectors and their corresponding labels were decreased by the factor Q.

Figure 3.1 shows the effect of downsampling on the right hand index flex sensor readings using both approaches with $Q = 2$. This feature, i.e., right hand index flex

Algorithm 3.1 Reducing the Number of Feature Vectors

```
Data: Original feature vectors and their corresponding labels
Result: Downsampled feature vectors and their corresponding labels
for all sentences do
    for all repetitions do
        if a set of feature vectors FVs has the same label L then
            downsample FVs by a factor Q
            FVs_new ⇐ downsampled FVs
            Labels_new ⇐ L
        end if
    end for
end for
```

sensor, from sentence 15 in Table 2.1 was chosen arbitrarily to illustrate some examples of the methods described throughout this chapter.

A slight difference between the two downsampling approaches can be noticed from the previous figure. Though they produce approximately similar vector lengths and shapes, the "`resample`" function changes the values slightly. This is due to the embedded low pass filtering in the "`resample`" command.

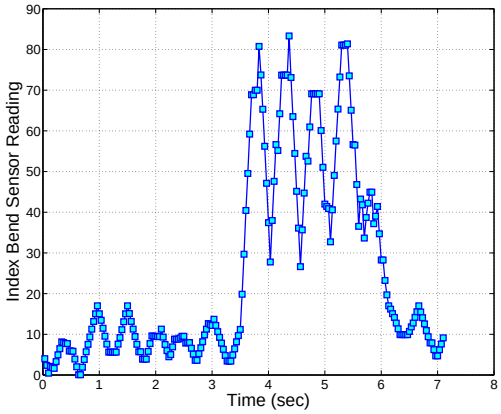
3.1.2 Normalization. As detailed in section 2.2, features coming from the data glove have different scales. Accordingly, they must be normalized to have a common range before further analysis. For this purpose, the z-score is used to standardize each feature in the training and testing sets. As a result, a standardized feature will have a zero mean and unit variance.

The z-score of any data point x from a set of data X with sample mean $\bar{x}$ and standard deviation s can be obtained using Equation 3.1.

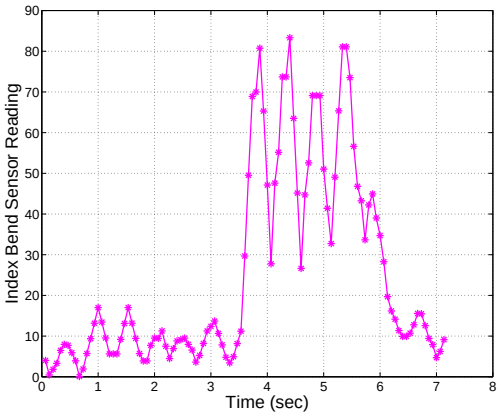
$$z = \frac{(x - \bar{x})}{s} \tag{3.1}$$

where z is the z-scored value that represents the distance measure of x from $\bar{x}$ in terms of s.

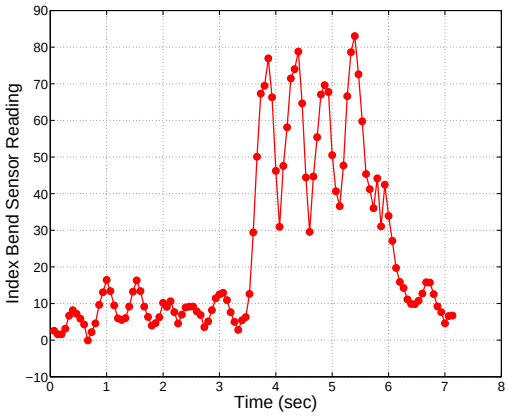
This method preserves the original dataset shape properties as illustrated in Figure 3.2 for the same example of index flex sensor readings of sentence 15.



(a) Original Signal



(b) Take Every Other Sample



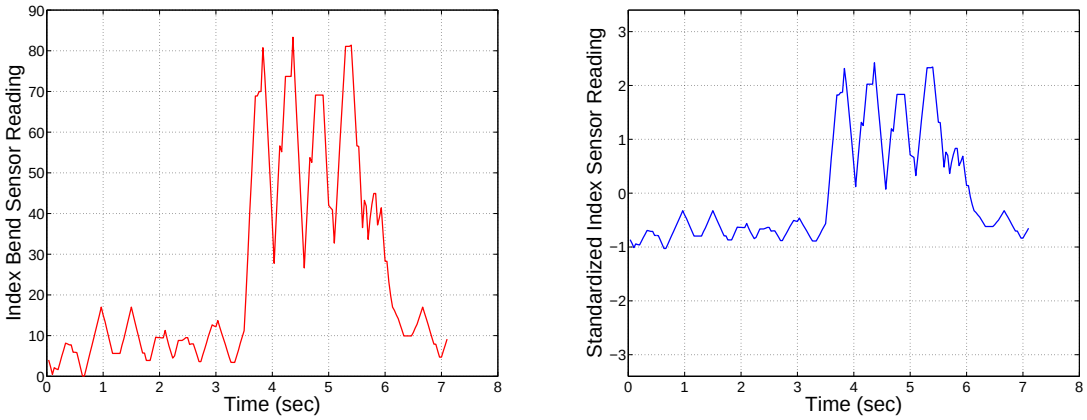
(c) Resampled with a ratio of (1:2)

Figure 3.1: An Example of Original and Reduced Feature Vectors of a Signal

In the proposed solution, 70% of the dataset was used for training and 30% for testing. Next, the mean and standard deviation of the training set were calculated to standardize it as shown in Equation 3.2. Then, $\bar{X}_{train}$ and S_{train} were saved and reused to standardize the test set as given in Equation 3.3.

$$Z_{train} = \frac{(X_{train} - \bar{X}_{train})}{S_{train}} \tag{3.2}$$

$$Z_{test} = \frac{(X_{test} - \bar{X}_{train})}{S_{train}} \tag{3.3}$$



(a) Original Readings

(b) Standardized Readings

Figure 3.2: Standardization Effect

where Z_{train} is the z-scored training set that includes N standardized vectors for N features such that:

- $Z_{train} = \{Z_{f_1}, Z_{f_2}, ..., Z_{f_N}\}$
  - $Z_{f_i} = \{z_1, z_2, ..., z_T\}$
  - $i \quad = \{1, 2, ..., N\}$
  - T is the total number of feature vectors in the original set X_{train}
- $X_{train} = \{X_{f_1}, X_{f_2}, ..., X_{f_N}\}$
  - $X_{f_i} = \{x_1, x_2, ..., x_T\}$
- $S_{train} \ = \{s_{f_1}, s_{f_2}, ..., s_{f_N}\}$
- $\bar{X}_{train} = \{\bar{x}_{f_1}, \bar{x}_{f_2}, ..., \bar{x}_{f_N}\}$

This approach was employed to ensure the validity of the proposed system in a practical scenario where one test sentence is presented to the classifier and hence, does not have a representative mean and standard deviation. Therefore, these statistical measures are used from the training phase.

3.2 Feature Extraction

The next step in the system is to extract useful features from raw data to make it more precise and concise. In the experimental results section, the validity of these fea-

tures is evaluated by different measures, such as sentence, word, and class recognition accuracies.

An essential advantage of using sensor-based systems is the type of captured gesture data. It usually does not require heavy computational methods to extract features as compared with vision-based systems. In this work, original data received from one DG5-VHand glove represents the amount of bend in each finger in addition to the hand acceleration and orientation. These raw features hold important information yet are noisy to some extent. Therefore, to improve the overall system performance, some dynamic and statistical approaches were applied separately to get more representative features as detailed in the following sections.

3.2.1 Velocity and acceleration. This technique is based on extracting dynamic information from the raw data by computing the velocity and acceleration of feature vectors. Assuming zero initial displacement, velocity is defined as the difference between consecutive observations. Therefore, a vector of zeros was appended before the first feature vector. Referring to Equation 2.1 where $i = \{1, 2, \cdots, N\}$, velocity vector (V_i) can be obtained using Equation 3.4.

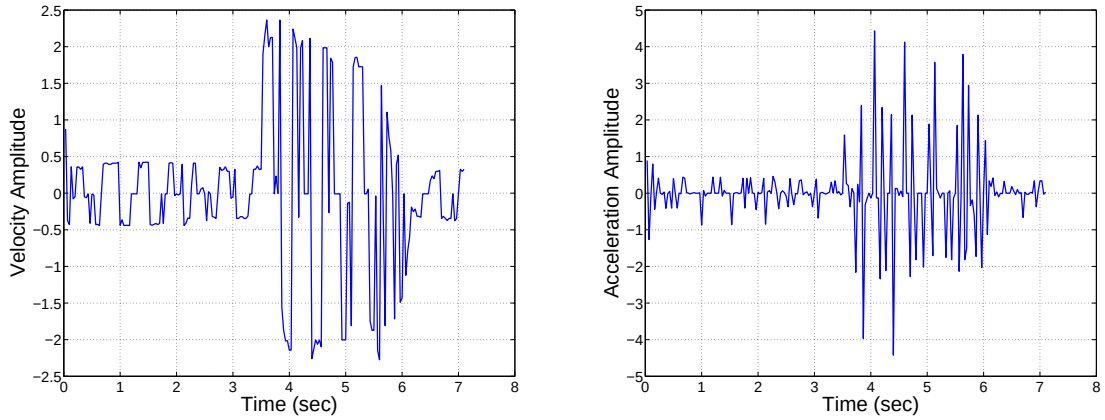
$$V_i = FV_i - FV_{i-1} \tag{3.4}$$

This feature emphasizes the occurrence of high differences between successive readings as illustrated in Figure 3.3(a) for Index_r feature.

After calculating the first difference, i.e., velocity, the second difference is calculated to find the acceleration feature vector (A_i) as shown in Equation 3.5.

$$A_i = V_i - V_{i-1} \tag{3.5}$$

Eventually, the original feature vector FV_i is augmented with either one or two of these dynamic features as demonstrated in Equation 3.6.



(a) Velocity Feature (b) Acceleration Feature

Figure 3.3: An Example of Dynamic Features

$$FV_i = [FV_i \quad V_i \quad A_i] \tag{3.6}$$

The first and second differences represent high pass filters as shown in Figure 3.3; therefore, it is good for low frequency noise attenuation. However, for high frequencies it might pass significant information along with its noisy components.

3.2.2 Statistical features. In order to get more descriptive features from the dataset, some statistical measures were used such as the mean and standard deviation. These two measures are usually used together to provide quantitative summaries about a set of samples. The central tendency of a sequence of feature vectors is measured by the mean, which reports their central value. Mathematically, simple averaging of the entire population is referred to as the mean (μ). For a subset of T observations $X = \{x_1, \ x_2, \ \ldots \ , x_T\}$, an estimate of the population average is defined as the sample mean ($\bar{x}$), Equation 3.7.

$$\bar{x} = \frac{1}{T} \sum_{k=1}^{T} x_k \tag{3.7}$$

On the other hand, the standard deviation (σ) measures the statistical standard deviation or dispersion from the sample average. Hence, an estimate of the population standard deviation is called sample standard deviation (s), which can be found using Equation 3.8.

$$s = \left(\frac{1}{T-1} \sum_{k=1}^{T} (x_k - \bar{x})^2 \right)^{1/2} \tag{3.8}$$

For sign language data, these two measures represent the mean and standard deviation of the hand movement and orientation for some time portion. Accordingly, the proposed feature extraction technique determines a sliding window mean and standard deviation for each feature separately. As such, it takes a set of observations according to a predefined window size and then calculates $\bar{x}$ and s for each feature as shown in Equation 3.9 and 3.10 respectively.

$$\bar{x}_i = \frac{1}{w} \sum_{k=i-\frac{w-1}{2}}^{i+\frac{w-1}{2}} x_k \tag{3.9}$$

$$s_i = \left(\frac{1}{w-1} \sum_{k=i-\frac{w-1}{2}}^{i+\frac{w-1}{2}} (x_k - \bar{x}_i)^2 \right)^{1/2} \tag{3.10}$$

where w is an odd number that denotes a given window size and i is the current feature from a set of N features such that $i = \{1, 2, ..., N\}$.

Consequently, for each feature vector, N window-based sample mean values $\bar{X}_{FV} = \{\bar{x}_1, \bar{x}_2, ..., \bar{x}_N\}$ and standard deviations $S_{FV} = \{s_1, s_2, ..., s_N\}$ are formed. In the end, one of these vectors or both can be appended to their original features as illustrated in Equation 3.11.

$$FV_i = [FV_i \quad \bar{X}_{FV_i} \quad S_{FV_i}] \tag{3.11}$$

Afterwards, the window is slid by one feature vector and the previous step is repeated as described in Algorithm 3.2.

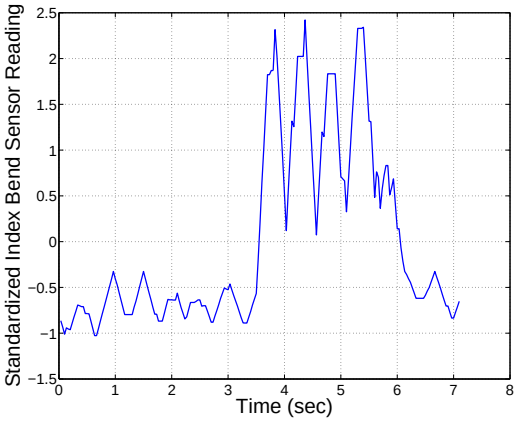
Algorithm 3.2 Window-Based Mean and Standard Deviation

Data: Original feature vectors
Result: Feature vectors with window-based mean and standard deviation
Define window size (w)
StartInd = `ceil` (w/2);
EndInd = `floor` (w/2);
for all sentences **do**
  for all repetitions **do**
    append $\frac{w-1}{2}$ zero vectors to the beginning and end of the repetition
    while $StartInd \leqslant i \leqslant (Size(FVs) - EndInd)$ **do**
      Find feature-wise $\bar{X}_{FV_i}$ and S_{FV_i} in the range ((i - EndInd) : (i + EndInd))
      Concatenate $\bar{X}_{FV_i}$ and S_{FV_i} with the original features FV_i
    end while
    Delete the appended zero vectors
  end for
end for

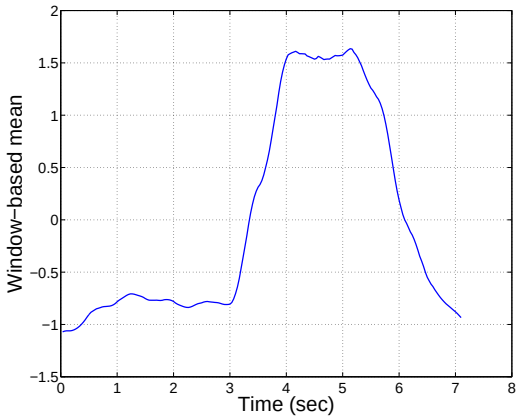
In statistics, the unweighted mean of an equal number of data from the past and future is known as the simple moving average. It provides a sequence of averages from sequential subsets of the dataset. The purpose of using this approach is to reduce short-term fluctuations and reserve long-term trends. As such, it is considered as an example of a low-pass filter and its result is a smoothed version of the original signal. Figure 3.4 illustrates an example of the sliding window mean and standard deviation for a given input signal.

3.3 Vision-based Technique

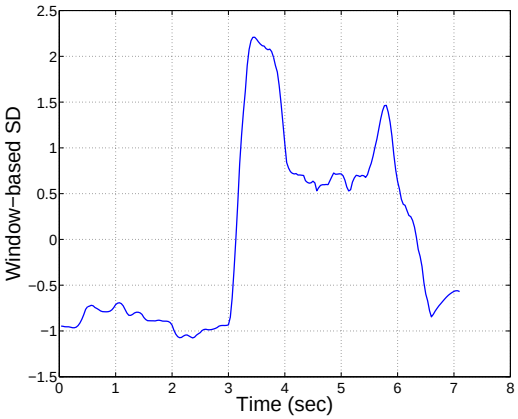
As mentioned previously, the proposed classification solution is intended to be an alternative for sequence-based classifiers such as HMM. A vision-based system using a similar dataset to ours is reported by Assaleh et al. in [13]. The system used HMM for training and testing. In this section, we review its preprocessing and feature extraction solutions. In the experimental results section, we compare our proposed work against this existing solution.



(a) Original Signal



(b) Window-based Mean



(c) Window-based Standard Deviation

Figure 3.4: An Example of Statistical Mean and Standard Deviation Features ($w = 31$)

3.3.1 Data collection and labeling. In the vision-based technique, the signer's gestures were acquired using a digital camera, as illustrated in Figure 3.5. For each sentence in Table 2.1, 20 repetitions were performed by one user with unrestricted clothing and background. Each video was obtained with a frame rate of $25Hz$ and a spatial resolution of 720×528. Then, the sequence of time dependent images was labeled manually with their corresponding gestures.

3.3.2 Feature extraction. The vision-based dataset has the same set of sentences described in section 2.1. In order to extract useful features from this data, elimi-

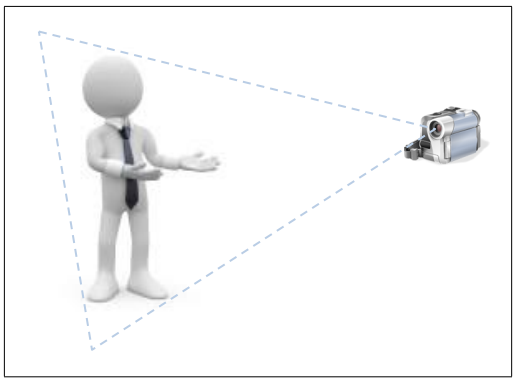


Figure 3.5: Vision-based System Elements

nate its temporal dependencies, and reduce its dimensionality, an innovative approach, proposed by Shanableh et al. [9], was employed. This technique starts by extracting motion information from a sequence of images. As such, it uses accumulated differences (ADs), which summarize the activity of a given sequence into one image $I_{g,i}^{(j)}$ as illustrated in Figure 3.6.

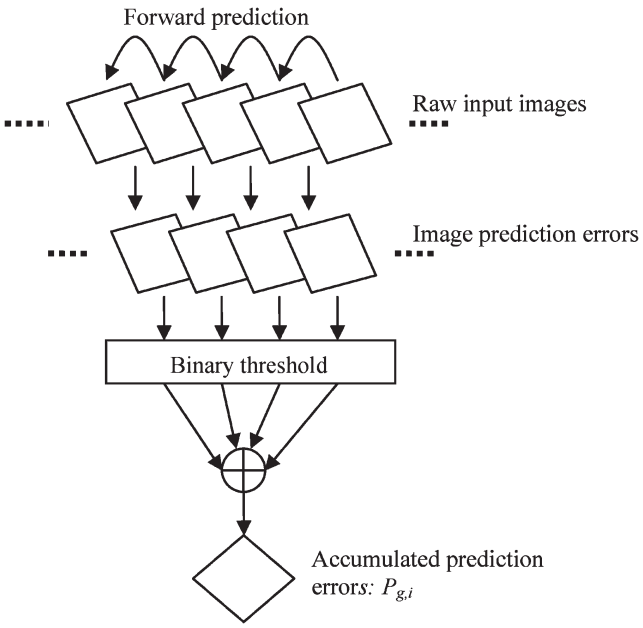


Figure 3.6: Illustration of the temporal feature-extraction technique [9]

An accumulated differences (ADs) image $AD_{g,j}$ of a gesture at index g and an image at index j of the i^{th} repetition can be found using Equation 3.12.

$$AD_{g,j} = \sum_{j=1}^{n-1} \partial_j \left(\left| I_{g,i}^{(j)} - I_{g,i}^{(j-1)} \right| \right) \tag{3.12}$$

where n is the total number of images in that repetition and ∂_j, Equation 3.13, is a binary threshold function of the j^{th} frame.

$$\partial(x) = \begin{cases} 1, if\ |x| \geqslant threshold \\ 0, if\ |x| < threshold \end{cases} \tag{3.13}$$

The $threshold$ value was defined as: $\mu + x\sigma$, where μ is the mean pixel intensity of $AD_{g,j}$ and σ is its standard deviation.

The above mentioned ADs technique removes the time dependency for one gesture. To adapt it for sentences, authors in [13] presented the overlapping sliding window approach that accumulates the differences between the video frames. The best window size can be determined empirically, then it is set to compute the ADs by sliding it by one video frame at a time.

The next step is to apply the 2-D discrete cosine transformation (DCT) to the ADs image in order to map it into a set of transform coefficients [40]. Let f be an $N \times M$ image, then its DCT coefficient $DCT(n, m)$ at specific row $n \in N$ and column $m \in M$ of the DCT matrix is given by Equation 3.14:

$$DCT(n,m) = \frac{2}{\sqrt{MN}} C(n)\, C(m) \ldots \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} f(i,j)\ \ cos\left(\frac{\pi n}{2M}.(2i+1)\right) cos\left(\frac{\pi m}{2N}.(2j+1)\right) \tag{3.14}$$

where $C(n)$ is a normalization factor equal to $\frac{1}{\sqrt{2}}$ for $n = 0$ and equals 1 otherwise.

The advantage of applying DCT is the resultant energy compaction; therefore, the important information in the image is saved in the top left corner of the DCT matrix. Subsequently, zig-zag scanning (or zonal coding) can be applied with a certain cutoff to choose a suitable number of coefficients. The output of zonal coding gives the resultant feature vector.

Finally, these feature vectors are labeled so that they can be used to train and evaluate the system. A general block diagram of this approach is given in Figure 3.7.

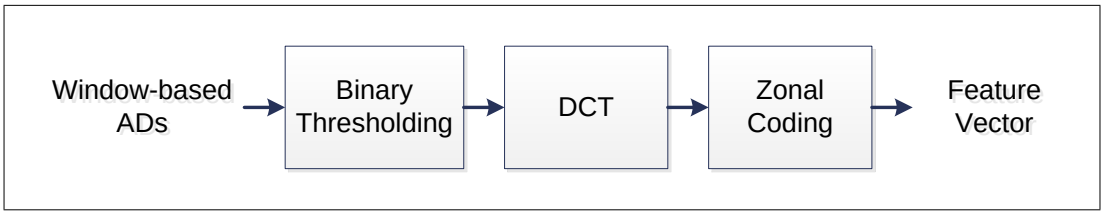


Figure 3.7: Block Diagram of the Feature-Extraction Technique

In the next chapter, we introduce our own classification technique. In the experimental results section, we use it to classify Arabic sign language using our sensor-based dataset and the reviewed vision-based dataset. The work in [13] used HMMs for training and classification; therefore, we also compare our classification results against those obtained by HMMs.

Chapter 4: Classification

In machine learning, the problem of mapping a new observation to a label is called classification. As part of a supervised learning, classification is based on a training set of correctly labeled observations.

In this study, an observation is a feature vector and a label or class is a word from a predefined word-lexicon. Since the total number of words is 80, the problem is a multiclass classification problem.

This chapter starts by introducing the proposed Modified k-Nearest Neighbors classifier. Then, an overview of a time-series modeling neural networks is presented. Finally, to lay the ground for the classification results in the following chapter, the system's evaluation parameters are described.

4.1 Modified k-Nearest Neighbors

Starting with one of the simplest machine learning algorithms, k-Nearest Neighbors (KNN) was used. As detailed in section 1.3.1, KNN is a non-parametric method and a type of instance-based classifiers. Thus, the only requirement for the training stage is storing all training instances and their labels. Then for classification, it computes the distances between any given test instance and all other training instances. Finally, it assigns the most frequent class of k nearest training samples to that test instance, where k is usually a small integer number that can be chosen empirically.

Since our data is sequential, we propose to modify KNN to be suitable for classifying such data. Following the experimental setup of the vision-based system reported in [13], the dataset was divided into 70% for training and 30% for testing. Thereafter, the predefined MATLAB function "`knnsearch`" was used to implement the conventional KNN classifier. Given a test sentence with T observations, where each observation FV_t is a set of features at time t, `knnsearch` searches the training set to determine the distance from each training data point to a given test instance. Then it sorts them in

an ascending order to report the closest k points $[L_{t1}, L_{t2}, ..., L_{tk}]$ for each observation in the test sentence.

Due to the temporal nature of the data, it is important to consider the context of the predicted label in the final prediction. Subsequently, the statistical mode approach was employed for each test sentence. It replaces a predicted label by the most common one in a subset of its surrounding predictions. To increase the accuracy of the prediction and take the context into account, the top three ($m = 3$) predicted labels are retained. Furthermore, the top three predicted labels of the previous and future feature vectors are also retained. This is preformed based on a window of feature vectors with size $ModeW$.

To compute the label of a current feature vector, the statistical mode is then computed from all of the above-mentioned labels. For further illustration, Figure 4.1 presents an example of this approach with a context window size $ModeW = 3$, three nearest neighbors ($k = 3$), and using the top three predicted labels ($m = 3$).

In addition, a flowchart of the modified KNN with the statistical mode approach is given in Figure 4.2.

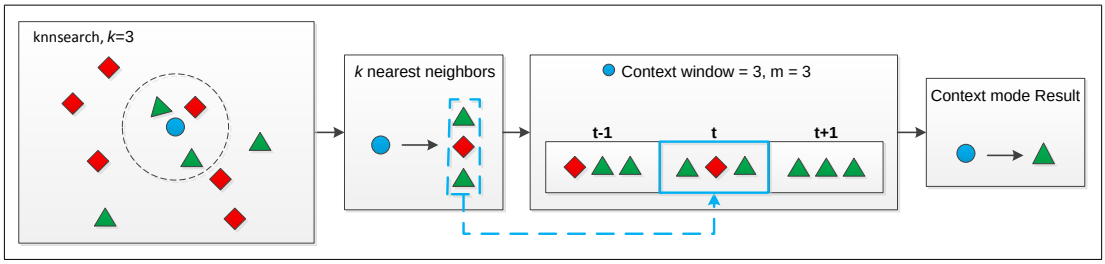


Figure 4.1: Illustration of the Modified KNN with the Statistical Mode Approach

Algorithm 4.1 provides further description of the statistical mode approach, where m labels are considered from each element in $Labels$ and w is a common context window size for each test sentence. Both w and m can be determined empirically.

For further enhancement of the predicted labels, a second technique is employed as a final stage in the proposed modified KNN. It uses a moving median filter for each

test sentence with an odd window size w. It is implemented using `medfilt1` MATLAB function, which is a one-dimensional median filter such that:

$$L_i = \textit{median}\left(Labels(i - \frac{w-1}{2} : i + \frac{w-1}{2})\right) \tag{4.1}$$

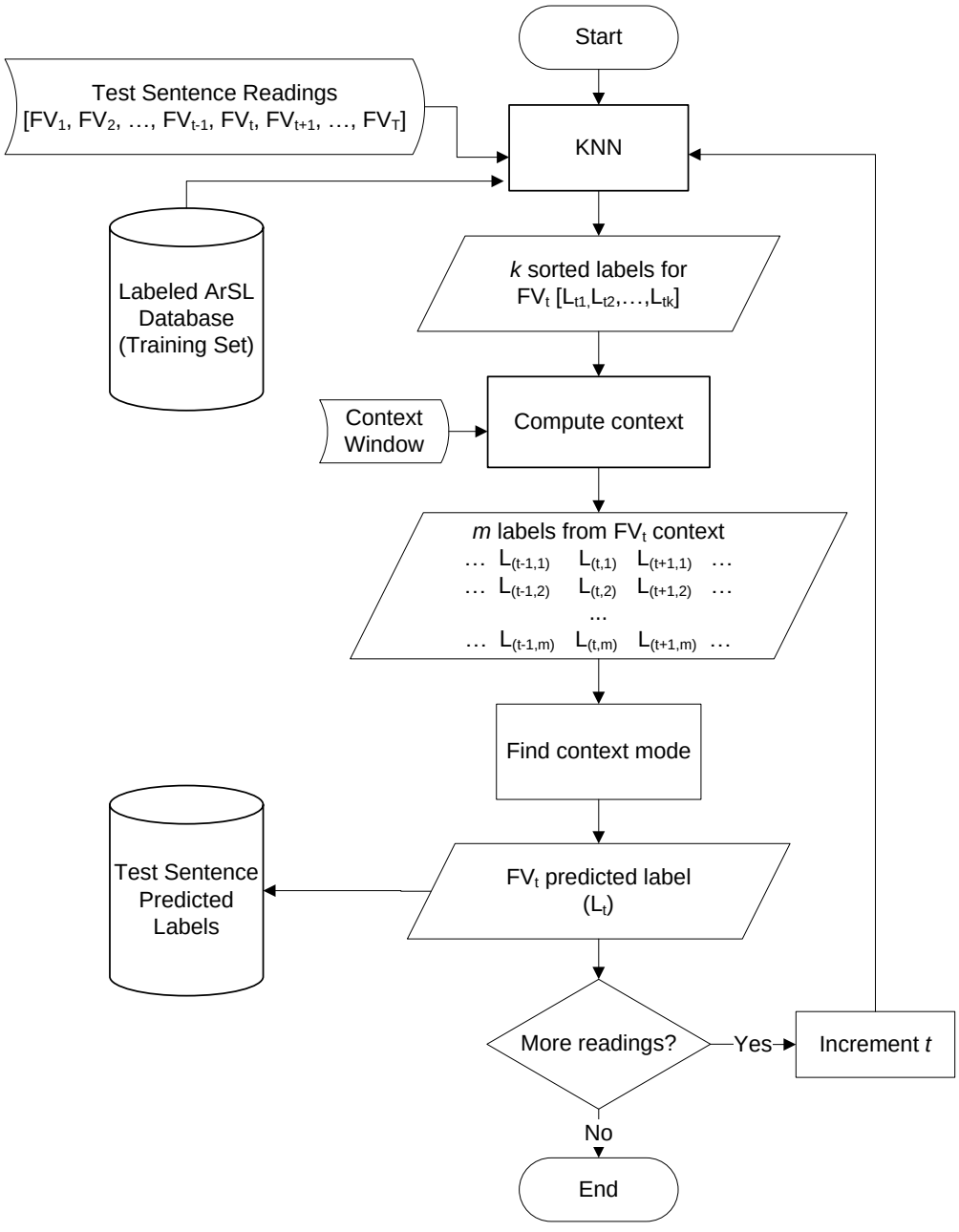


Figure 4.2: Proposed Modified KNN with Integrated Statistical Mode

Algorithm 4.1 Statistical Mode Approach

Input: Predicted labels of one sentence
Output: More accurate labels
Define window size (w)
StartIndex = `ceil` (w/2);
EndIndex = `floor` (w/2);
for T Labels **do**
  while $StartIndex \leqslant i \leqslant (T - EndIndex)$ **do**
    L_i = `mode` $(Labels(i - \frac{w}{2} : i + \frac{w}{2}))$
  end while
end for

This stage is illustrated with a flowchart shown in Figure 4.4 in addition to an example given in Figure 4.3 to highlight its effect on the accuracy of the sequential data.

The last step after classification is post-processing, where some rules are imposed on the resultant words and sentences such as word length thresholding and deletion of repetitive words. The former defines the minimum number of consecutive labels that make up a word. For instance if the $threshold = 5$, then a sequence of at least five labels is required to recognize a word at index L. On the other hand, the latter prevents successive replication of any word in a predicted sentence, which is an invalid case in Arabic in general and the dataset in specific.

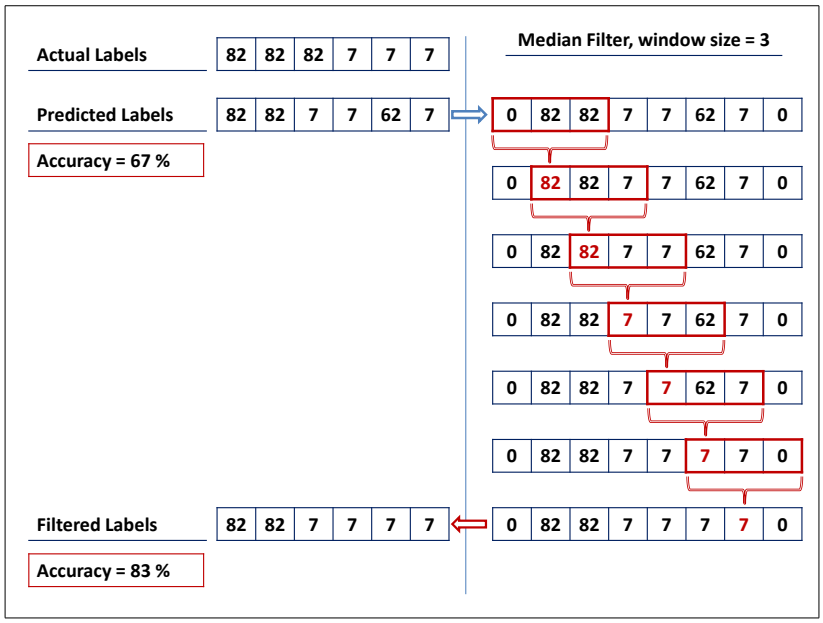


Figure 4.3: Illustration of the Median Filter Approach

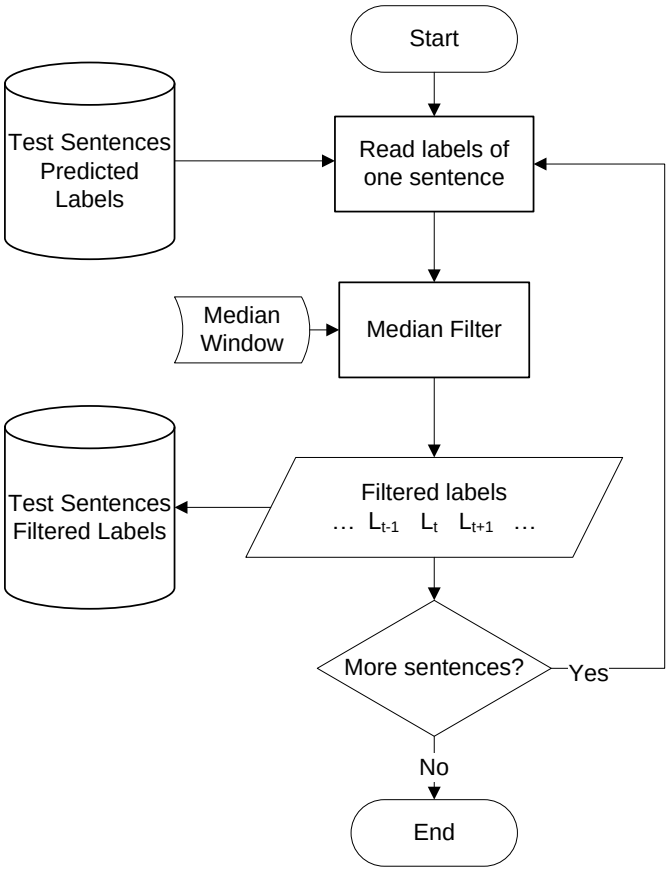


Figure 4.4: Proposed Modified KNN with Post Median Filter

4.2 Nonlinear Autoregressive with eXogenous Inputs (NARX)

Since our data is sequence-based then in addition to the proposed KNN technique, we can make use of machine learning tools that are designed to work with sequential data. Examples are hidden Markov models [41], recurrent neural networks [42], [43], and NARX [44]. In this work, we make use of NARX as it is a powerful machine learning tool.

NARX is a powerful time-series modeling neural network based on the linear ARX model. It is a recurrent dynamic neural network with feedback connections including a predefined number of layers. Figure 4.5 is an example demonstrating a NARX model that uses two-layer feedforward neural networks to approximate its function.

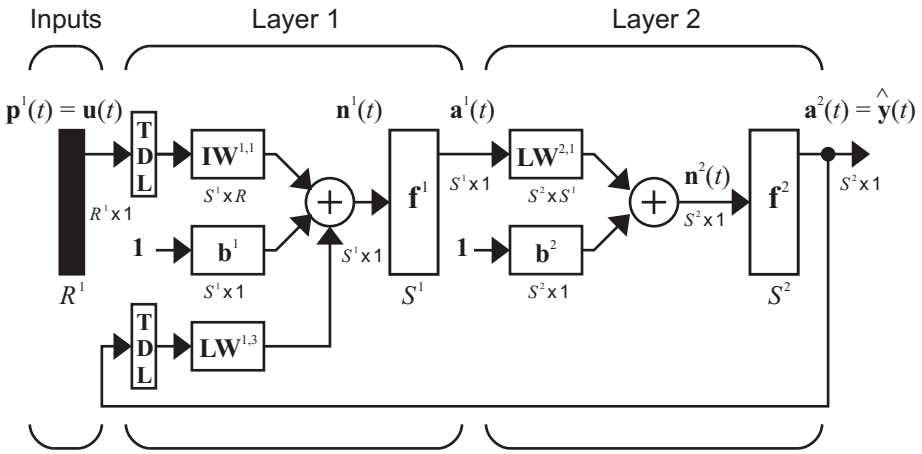


Figure 4.5: A NARX Model with Two Feedforward Neural Networks [45]

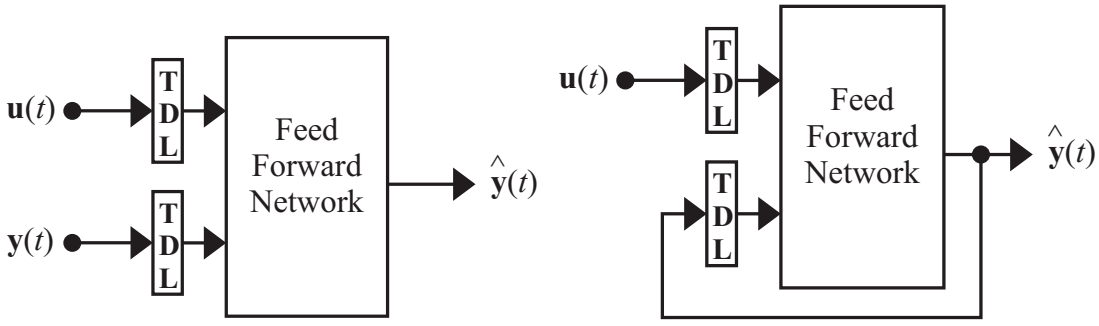
From its name, NARX employs regression to predict the next value of the output signal $y(t)$ using previous values from the dependent variable y and independent (exogenous) input variable u as shown in Equation 4.2.

$$y(t) = f(y(t-1), y(t-2), ..., y(t-n_y), u(t-1), u(t-2), ..., u(t-n_u)) \quad (4.2)$$

There are two network architectures that can be used when designing a NARX network as follows:

- **Series-Parallel (Open-Loop):** In this architecture actual values of the output variable are fed to the network to estimate the next value as shown in Figure 4.6(a). Upon the availability of the targets, this architecture is efficient for training and can achieve more accurate results.
- **Parallel (Closed-Loop):** As can be noticed from Figure 4.6(b), the closed-loop architecture uses the predicted output values as a feedback to estimate the next target. Accordingly, it is used to predict unseen data points that have unknown targets.

To employ NARX for sign language recognition, we propose to use the numeric gesture labels as a response variable (y). The NARX predicted output is then rounded



(a) Series-Parallel Architecture (b) Parallel Architecture

Figure 4.6: Possible NARX Architectures [45]

to the closest integer. The gesture labels are numbered from 1 to 80. Different design parameters and algorithms used to test this approach are detailed in Chapter 5.

4.3 Evaluation Measures

A sentence is a set of sequential feature vectors where successive vectors of similar labels constitute a word belonging to that sentence. The outcome of the classifier is a series of predicted labels associated to a test sentence. In order to evaluate the performance of the proposed methodology, class, word, and sentence recognition rates are reported. As such, the corresponding actual labels of each predicted test sentence are known. Accordingly, the class recognition rate is defined as the ratio of correctly classified feature vectors to the total number of test feature vectors. An example of actual and predicted labels of one test sentence is given in Figure 4.7, where the x and y axes represent the position of a predicted label in a test sequence and a predicted word index respectively.

On the other hand, the word recognition rate is given by Equation 4.3, where D, I and S are the number of deleted, inserted, and substituted words in a predicted sentence, whereas N is the total number of words in the actual sentence or reference [41].

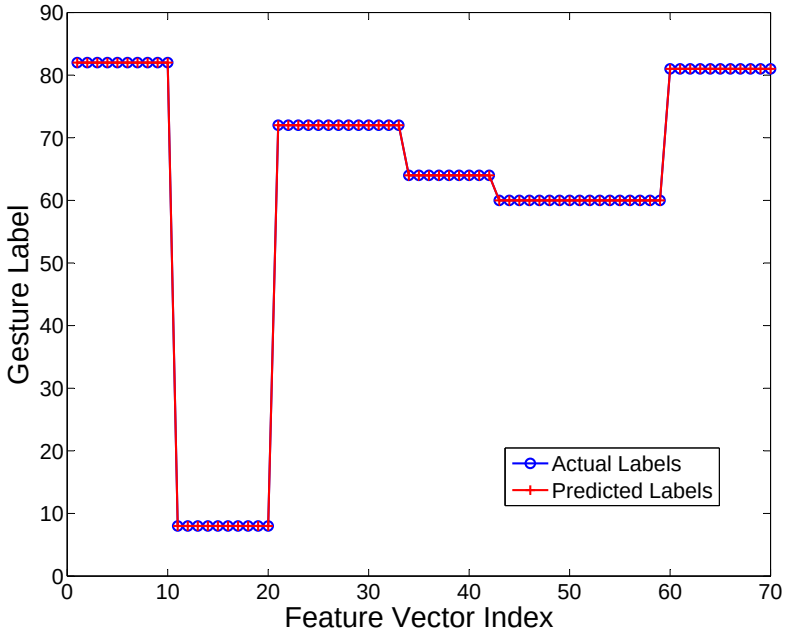


Figure 4.7: Illustration of a Correctly Classified Sentence

$$Word\ Recognition\ Rate = 1 - \frac{D + S + I}{N} \tag{4.3}$$

For further illustration, the following example gives an actual sentence and its predictions with a variation of deletion, insertion, and substitution errors:

- Actual : *This is a correct sentence*
- Predictions :
  - Deletion: *This is a sentence*.
  - Insertion: *This is **not** a correct sentence*.
  - Substitution: *This is a **wrong** sentence*.

Finally, sentence recognition rate is defined as the ratio of correctly classified sentences to the total number of test sentences. A sentence is correctly classified if all the words constituting it are correctly recognized with their original order. While Figure 4.7 demonstrates a correctly classified sentence, Figure 4.8 shows a wrong classification of another sentence even though some of its words were correctly predicted.

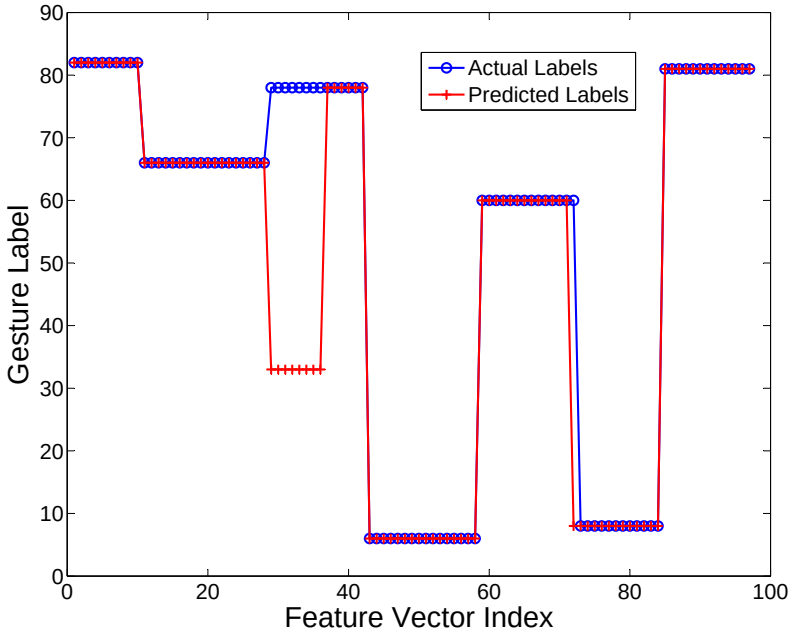


Figure 4.8: Illustration of a Misclassified Sentence

Chapter 5: Results and Discussion

In this chapter, experimental results of the proposed system are discussed. We start with the proposed sensor-based solution and report the experimental results using the modified KNN approach. Thereafter, we compare our results against the results obtained using NARX. We also compare our results against the vision-based system reported by Asslaeh at al. [13], which used HMMs.

5.1 Sensor-based Results

The sensor-based dataset, introduced in Chapter 4 above, was divided in a way that 70% of each sentence repetitions were used for training and the remaining 30% were used for testing. In order to ensure a fair estimate of the recognition rates, a Round Robin technique was used. For this system, three rounds were applied to generate three different sets of train and test feature vectors with approximately the same portion in each round. The average recognition rate is then reported.

Having the original feature vectors, two resampling factors ($Q = 2, 3$) were used, as detailed in section 3.1. Consequently, five types of feature vectors were obtained as follows:

- Original feature vectors without resampling
- Resampled with filtering using a ratio of 1:2
- Leave one out without filtering
- Resampled with filtering using a ratio of 1:3
- Leave two out without filtering

Afterwards, feature extraction techniques were applied to these types and their results were normalized. The resultant features were then used for the proposed system as presented in the following sections.

5.1.1 MKNN results. As an instance based classifier, the proposed MKNN compares unseen data points from the test set with the instances from its training set

using some distance function. The predicted labels are then validated with the actual test labels to determine the accuracy of the classifier.

There are several parameters in the classification and post-processing stages that affect the overall recognition rates. The following summarizes these parameters according to their order of the system structure:

- **MKNN classifier parameters**
  - Basic KNN classifier
    - Number of nearest neighbors (k)
    - Distance metric
  - Proposed statistical mode approach
    - Context window ($ModeW$)
    - Number of nearest labels in the context (m)
  - Proposed median filtering
    - Filtering window ($MedW$)
- **Post-processing parameters**
  - Word Threshold ($WordTh$)

Cityblock distance, a simple and effective pairwise distance metric, was used for KNN to classify any new test point based on its distance from the training data points. In addition, k was set to three and hence, the three nearest labels were reported for each test feature vector. Next, for the proposed statistical mode approach, the number of nearest neighbors in the context was fixed to two. That is, the k nearest neighbors of both the previous and the future readings were taken into account as explained in Figure 4.1. The three parameters, $k = 3$, *Cityblock* distance and $m = 2$, are fixed throughout the experiments.

Then, to compute the context and find its most frequent label, a range of window sizes was examined to determine the best one for each set of feature vectors. This is followed by a median filter, which makes use of a different window size. Finally,

different thresholding values for the minimum word duration were applied to find the optimum class, word, and sentence recognition rates.

In the beginning, raw feature vectors along with their resampled versions were tested without any feature extraction technique. As such, three optimization parameters, applied to the mode filter, median filter, and word threshold, were varied to examine their effect on the recognition rates. Figure 5.1 shows the effect of changing $ModeW$ duration on the sentence recognition rate by fixing $MedW$ and $WordTh$ parameters for each distinct type of input feature vectors separately.

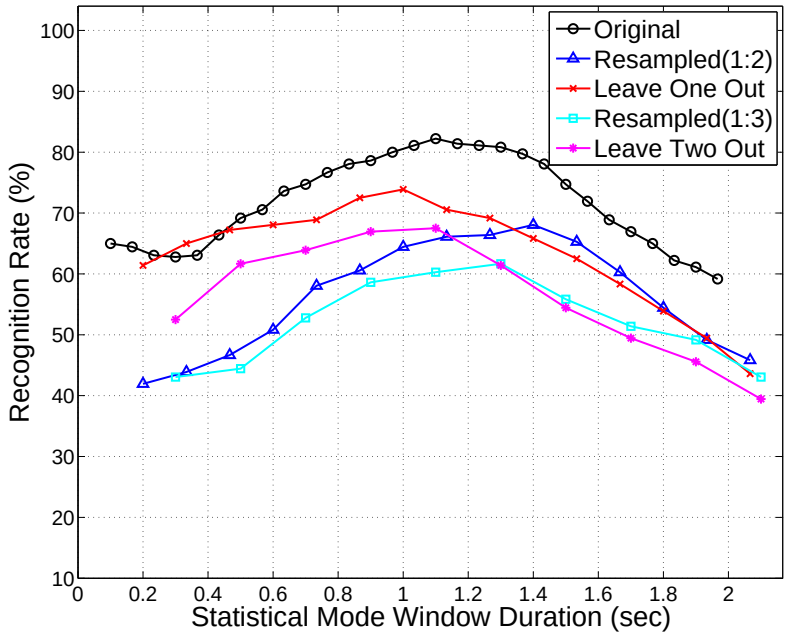


Figure 5.1: Sentence Recognition Rate vs. Mode Window (Raw Features)

It is observed that increasing the duration of the context window results in a higher recognition rate until it reaches a certain time limit where the accuracy starts to decline. This is due to the nature of the dataset's gestures, which do not have long durations. For instance, the sentence recognition rate for the original set of feature vectors is improved from 65% to 82% until the context window duration reaches 1.1 seconds. Then, it starts decreasing, all while considering more data points from the past

and future, which might result in it including irrelevant sensor readings that belong to previous and successive gestures.

Next, a range of median filter window sizes was applied to choose a proper refining window for the sequence of predicted labels. Figure 5.2 shows the effect of this filtering parameter on the sentence recognition rate.

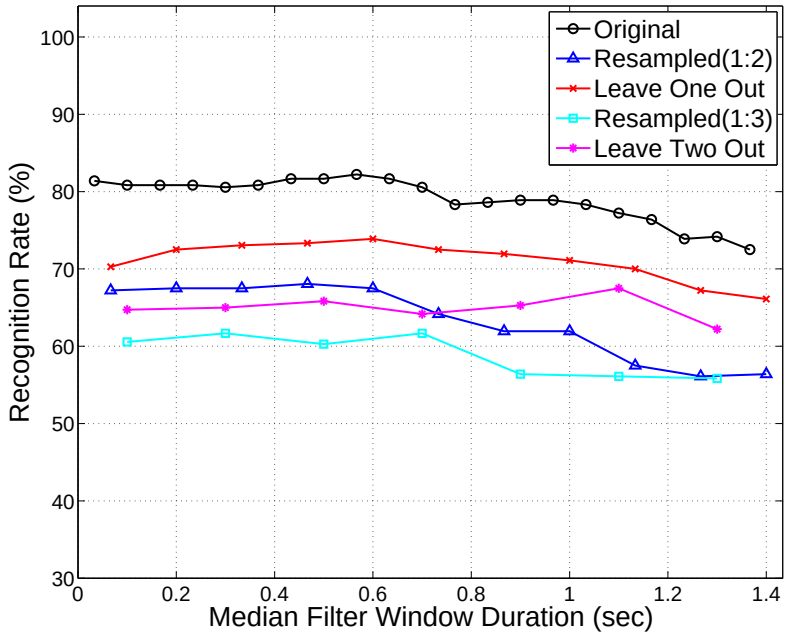


Figure 5.2: Sentence Recognition Rate vs. Median Window (Raw Features)

From the results shown, it seems that varying the window size for the post median filter is not too crucial, hence we can fix it as 0.6 seconds in this case.

The last step requires setting the minimum duration required to detect a word. The impact of this parameter ($WordTh$) is illustrated in Figure 5.3, where the x-axis represents its duration, while the y-axis shows the corresponding sentence recognition rate. Clearly, a small word threshold means that sub-words might be mistakenly recognized as gestures. Likewise, a large threshold might result in merging more than one word into one gesture.

A summary of the best obtained recognition rates with their optimization parameters is illustrated in Table 5.1. The parameters are presented as the number of predicted

labels instead of the time representation given in the figures. The maximum achieved sentence recognition rate was 82%, obtained from using the original set of feature vectors. This percentage accuracy is an average of three rounds with a standard deviation of 4.88.

Table 5.1: Raw Data Optimized Parameters and Recognition Rates

	Classifier Pars.		**Postproc. Par.**	**Recognition Rates**		
Input FVs	**ModeW**	**MedW**	**WordTh**	**Class (%)**	**Word (%)**	**Sentence (%)**
Original	32	17	10	87.17	85.49	82.22
Resampled (1:2)	20	7	5	81.59	73.46	68.06
Leave one out	14	9	7	85.42	80.13	73.89
Resampled (1:3)	12	3	4	80.37	66.60	61.67
Leave two out	10	11	4	83.12	74.25	67.50

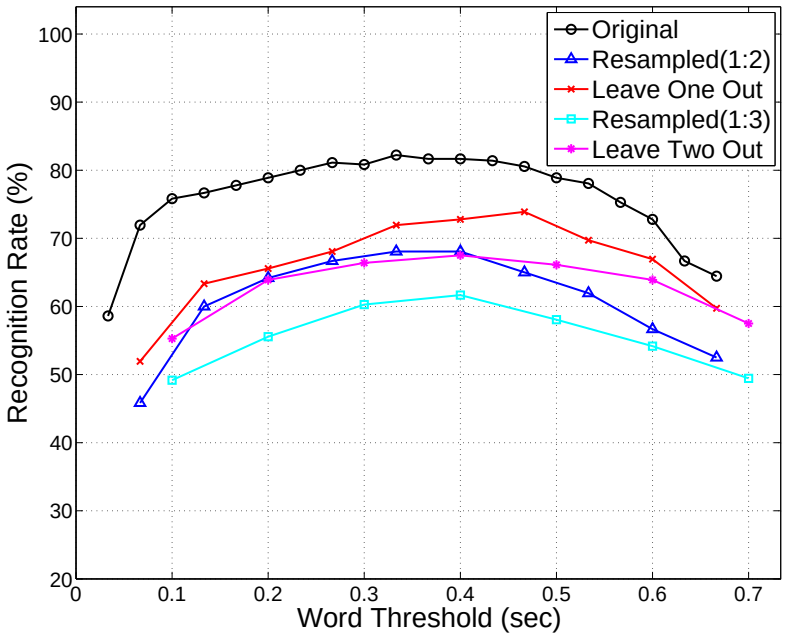


Figure 5.3: Sentence Recognition Rate vs. Word Threshold (Raw Features)

Since we did not use feature extraction, it can be noticed from Table 5.1 that using the original feature vectors without resampling gives the best recognition rates. However, experimental results indicate that this is not the case when the statistical feature extraction technique is used.

Afterwards, to investigate the validity of the proposed feature extraction techniques, $ModeW$, $MedW$, and $WordTh$ are also varied to evaluate the effect of augmenting dynamic and statistical features to their raw vectors.

Starting with (Raw|Velocity) features, the maximum obtained sentence recognition rate was 75% for the full set of feature vectors. Table 5.2 shows the best possible results obtained from this feature. Additionally, the sentence recognition rates of the five types of input feature vectors are presented in Figures 5.4 - 5.6 to show the effect of each parameter.

Table 5.2: Optimized Parameters and Recognition Rates of the Raw Data with Velocity

	Classifier Pars.		**Postproc. Par.**	**Recognition Rates**		
Input FVs	**ModeW**	**MedW**	**WordTh**	**Class (%)**	**Word (%)**	**Sentence (%)**
Original	36	15	6	84.63	77.97	75.00
Resampled (1:2)	12	15	6	80.30	64.51	58.61
Leave one out	18	3	5	81.17	70.20	63.06
Resampled (1:3)	10	3	4	78.99	61.44	56.11
Leave two out	10	11	4	79.04	63.66	56.11

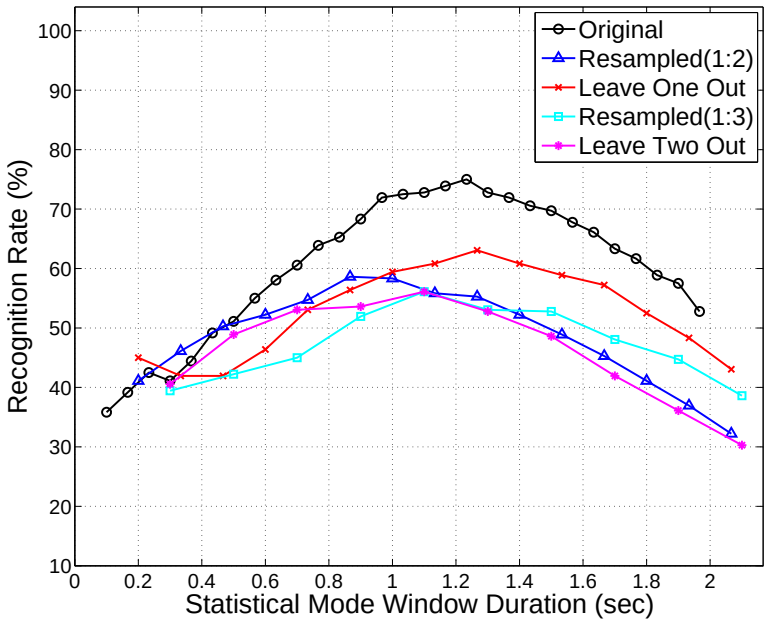


Figure 5.4: Sentence Recognition Rate vs. Mode Window (Raw|Velocity Features)

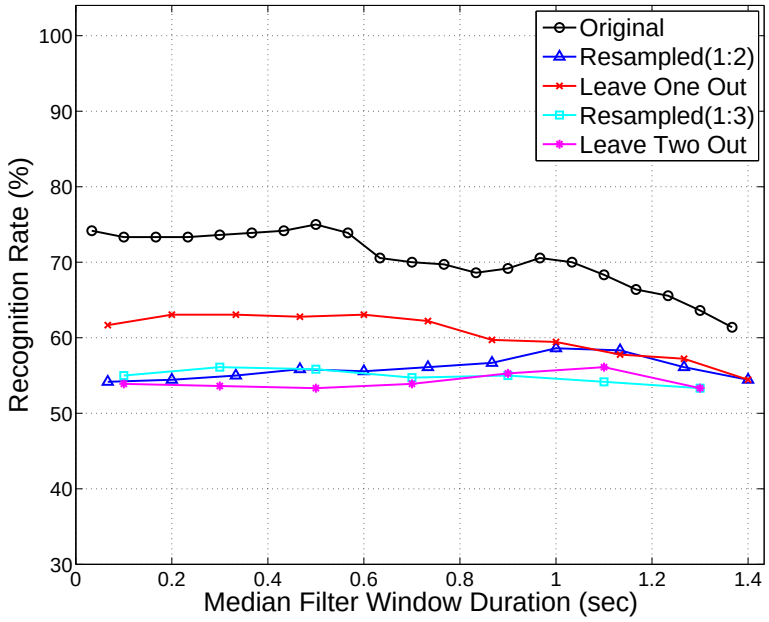


Figure 5.5: Sentence Recognition Rate vs. Median Window (Raw|Velocity Features)

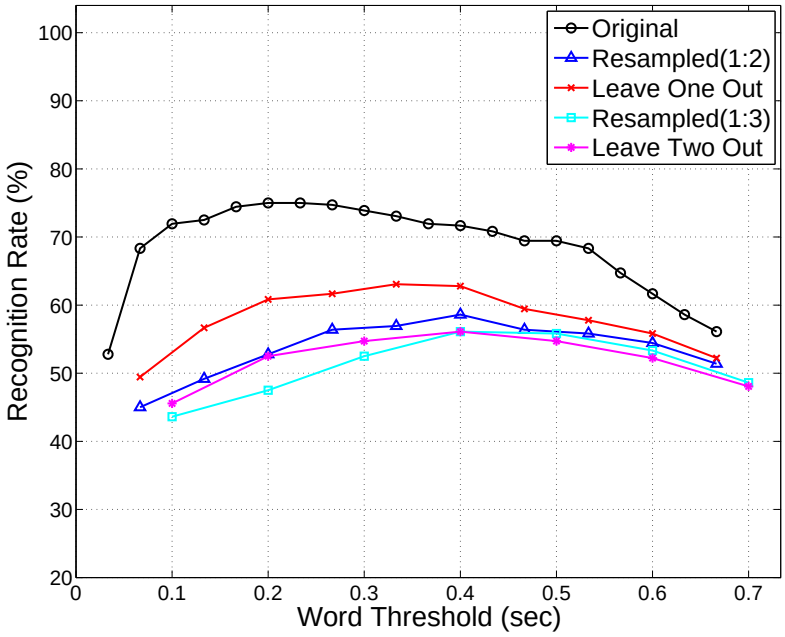


Figure 5.6: Sentence Recognition Rate vs. Word Threshold (Raw|Velocity Features)

Next, the second dynamic feature (acceleration) was assessed and again, the rate did not improve as shown in Table 5.3, where the maximum sentence recognition rate was 61% for the original feature vectors. However, the positive effect of using MKNN and post-processing parameters on the accuracies are obvious in the Figure 5.7.

Table 5.3: Optimized Parameters and Recognition Rates of the Raw Data with Acceleration

	Classifier Pars.		**Postproc. Par.**	**Recognition Rates**		
Input FVs	**ModeW**	**MedW**	**WordTh**	**Class (%)**	**Word (%)**	**Sentence (%)**
Original	24	25	13	81.12	67.25	61.39
Resampled (1:2)	16	5	6	76.93	58.63	50.28
Leave one out	16	7	4	75.04	53.14	48.06
Resampled (1:3)	10	1	4	74.90	54.64	46.11
Leave two out	8	7	5	73.29	50.46	43.06

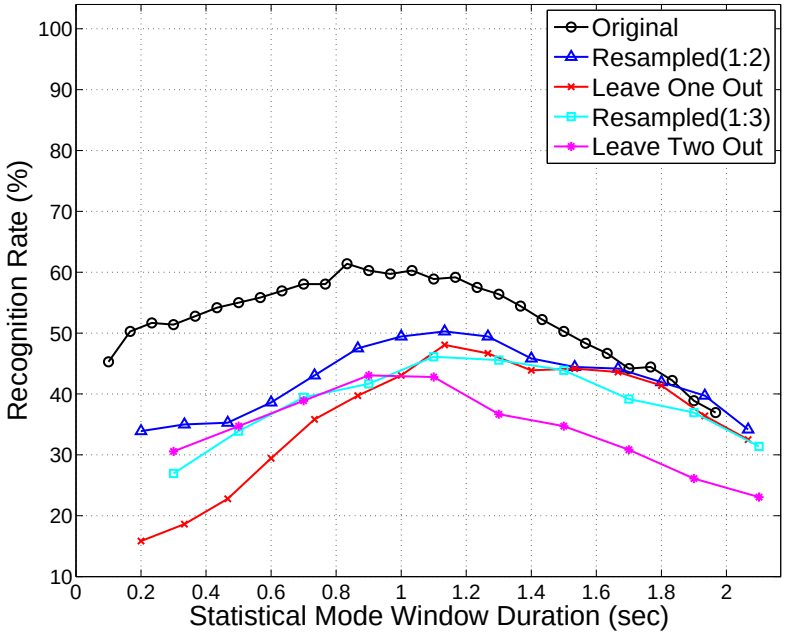


Figure 5.7: Sentence Recognition Rate vs. Mode Window (Raw|Accel. Features)

Although these features capture hand movement velocity and acceleration, it can be observed that they lower the final recognition accuracies due to the fact that they pass high frequency noise. Additionally, only one previous feature vector is considered while

computing these dynamic features. In the end, the obtained results from augmenting both velocity and acceleration with the raw features are presented in Table 5.4.

Table 5.4: Optimized Parameters and Recognition Rates of the Raw Data with Velocity and Acceleration

	Classifier Pars.		**Postproc. Par.**	**Recognition Rates**		
Input FVs	**ModeW**	**MedW**	**WordTh**	**Class (%)**	**Word (%)**	**Sentence (%)**
Original	20	29	13	80.33	66.41	60.00
Resampled (1:2)	12	17	6	74.73	51.44	44.44
Leave one out	16	5	6	74.21	54.64	47.50
Resampled (1:3)	14	7	3	71.94	52.61	43.33
Leave two out	10	9	5	72.79	51.18	40.83

The second feature extraction technique uses window-based sample mean and standard deviation (SD). Likewise, another parameter ($FVsW$) was introduced to determine a suitable window size that optimizes the overall system performance. This approach extracts the new features and augments them to the raw features. Consequently, experiments were carried out with different combinations of the four optimization parameters to study the impact of this approach on the final results.

The raw features were augmented first to the mean values and hence, (Raw|mean) features are evaluated. Then, (Raw|SD) and (Raw|mean|SD) were tested and the final results were discussed.

First, corresponding to the mean values, the obtained results indicated great improvements in the recognition rates. As such, the maximum sentence accuracy was 97% for resampled feature vectors as shown in Table 5.5. While fixing the classifier and post-processing parameters, the effect of $FVsW$ is illustrated in Figure 5.8, where the x-axis represents the number of feature vectors encapsulated in that window. On the other hand, the effect of the $ModeW$, $MedW$, and $WordTh$ parameters are shown in Figures 5.9 - 5.11.

Second, just as the mean feature, (Raw|SD) with MKNN resulted in high recognition rates as summarized in Table 5.6. The effect of changing the feature extraction parameter on the sentence recognition rate is shown in Figure 5.12. Other parameters follow approximately the same pattern as the (Raw|mean) features.

Table 5.5: Optimized Parameters and Recognition Rates of the Raw Data with Mean

	F.E. Par.	Classifier Pars.		Postproc. Par.	Recognition Rates		
Input FVs	**FVsW**	**ModeW**	**MedW**	**WordTh**	**Class (%)**	**Word (%)**	**Sentence (%)**
Original	47	22	29	10	93.29	95.56	95.56
Resampled (1:2)	43	14	1	6	93.36	97.97	97.22
Leave one out	45	14	3	7	91.93	95.82	94.17
Resampled (1:3)	35	10	9	2	93.20	97.25	96.67
Leave two out	37	10	9	3	91.04	93.73	92.22

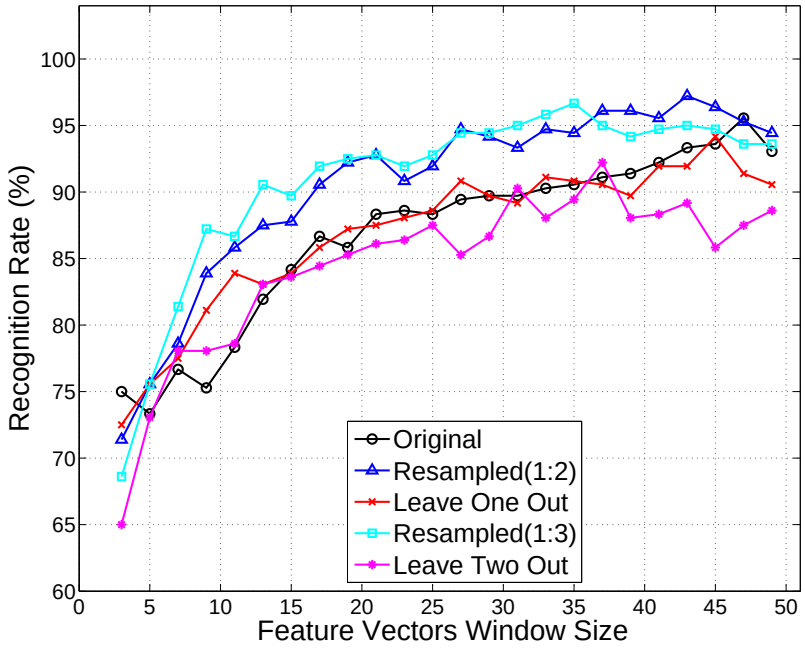


Figure 5.8: Sentence Recognition Rate vs. Feature Extraction Window (Raw|mean)

Table 5.6: Optimized Parameters and Recognition Rates of the Raw Data with SD

	F.E. Par.	Classifier Pars.		Postproc. Par.	Recognition Rates		
Input FVs	**FVsW**	**ModeW**	**MedW**	**WordTh**	**Class (%)**	**Word (%)**	**Sentence (%)**
Original	43	34	19	10	92.79	95.88	95.00
Resampled (1:2)	37	14	1	6	93.18	96.34	95.83
Leave one out	39	10	15	5	93.51	96.99	96.94
Resampled (1:3)	43	10	1	4	92.98	96.86	95.56
Leave two out	35	12	3	3	92.24	95.69	94.72

Finally, the combination of both mean and standard deviation features outperformed other techniques and provided accurate results as presented in Table 5.7, where resampled feature vectors achieved the best sentence recognition rate of 98.9%. The

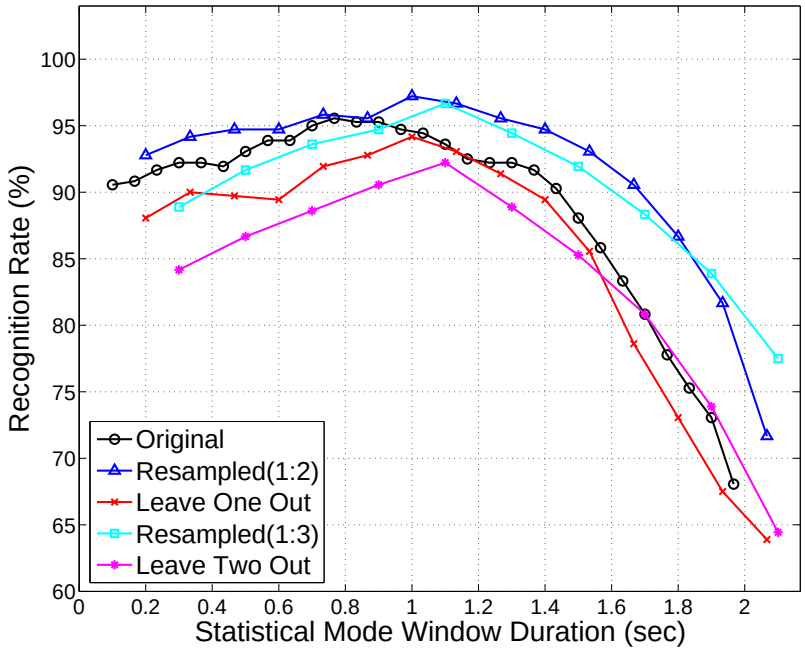


Figure 5.9: Sentence Recognition Rate vs. Mode Window (Raw|mean)

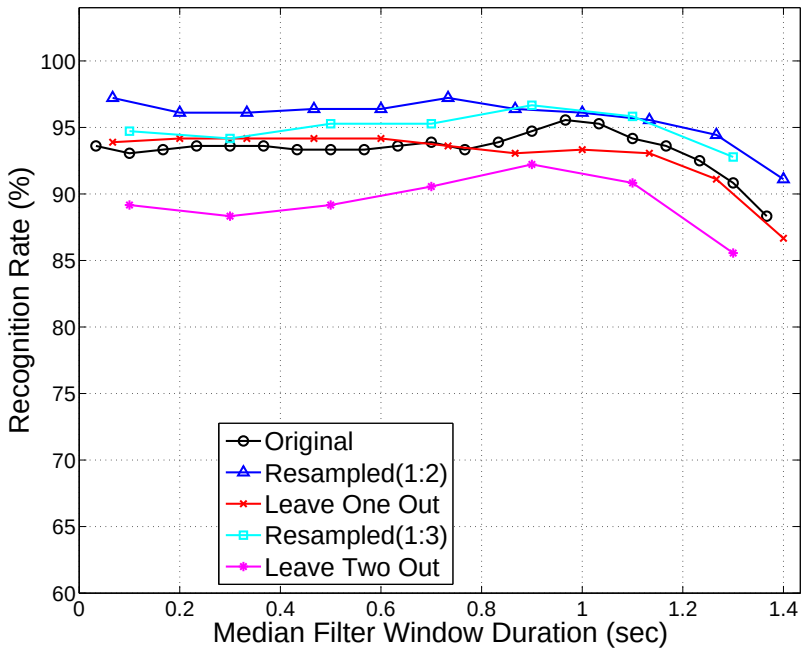


Figure 5.10: Sentence Recognition Rate vs. Median Window (Raw|mean)

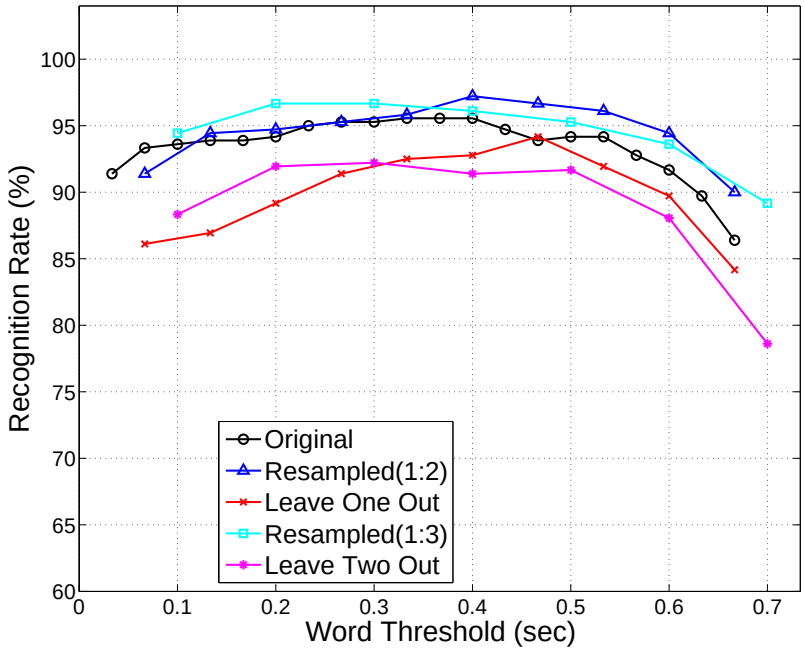


Figure 5.11: Sentence Recognition Rate vs. Word Threshold (Raw|mean)

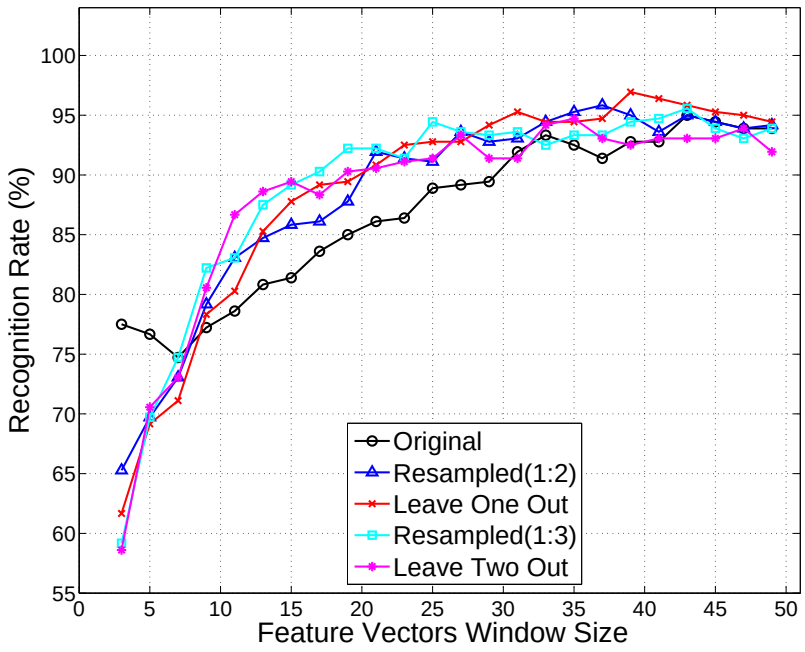


Figure 5.12: Sentence Recognition Rate vs. Feature Extraction Window (Raw|SD)

standard deviation of this averaged accuracy over three rounds is 1.27, which indicates that the optimization parameters are not over fitting a specific training set. The resultant sentence recognition rates for different $FVsW$ sizes are demonstrated in Figure 5.13.

Table 5.7: Optimized Parameters and Recognition Rates of the Raw Data with Mean and SD

	F.E. Par.	Classifier Pars.		Postproc. Par.	Recognition Rates		
Input FVs	**FVsW**	**ModeW**	**MedW**	**WordTh**	**Class (%)**	**Word (%)**	**Sentence (%)**
Original	43	26	33	14	93.80	98.43	97.78
Resampled (1:2)	43	14	15	1	93.81	98.82	98.89
Leave one out	29	14	15	4	93.58	97.58	97.78
Resampled (1:3)	37	6	3	5	93.50	98.76	98.61
Leave two out	37	8	9	2	92.79	97.32	97.22

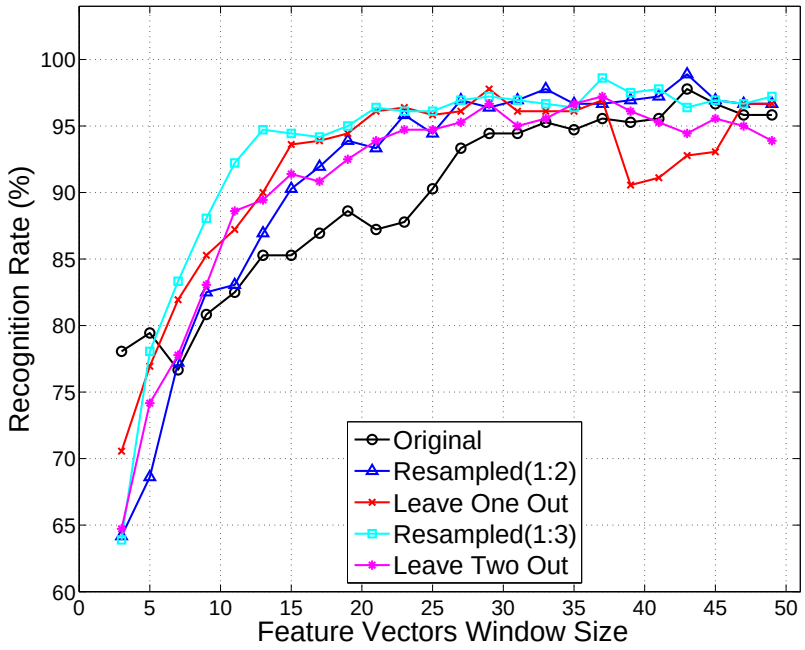


Figure 5.13: Sentence Recognition Rate vs. Feature Extraction Window (Raw|mean|SD)

5.1.2 NARX results. The nonlinear autoregressive network with exogenous inputs (NARX) is a time-series modeling neural network. Therefore, we examined its

performance on the sensor-based dataset, which has a temporal nature. Using MATLAB, a series-parallel NARX architecture was employed. First, the dataset was divided into three parts: 60% for training, 10% to validate the network and prevent overfitting, and the remaining 30% for testing. For this purpose, the "`divideind`" function was used to specify the indices of the data divisions implicitly. In addition, one hidden feedforward neural network layer was used to approximate the model function using a log sigmoid as a transfer function. The optimum number of neurons in this layer was chosen empirically. Then, the resilient backpropagation [46] "`trainrp`" was set as the training function since its time and memory requirements are more suitable for the large number of data presented to this model.

With these settings, the prediction of the future values of the targets in the test set is based on one past value from the targets (y) and another from the input feature vectors (u). This is due to the value of the delay, $n_u = n_y = 1$ as in Equation 4.2, which was chosen experimentally to achieve the maximum sentence recognition rate. Consequently, with 90 neurons, this network was modeled with many combinations of the feature vectors and feature extraction techniques. The best sentence recognition rate was found to be 82.5% for resampled raw feature vectors with a ratio of 1:3.

In order to compare the proposed MKNN with NARX in terms of the computational complexity, the set of (Raw|mean|SD) resampled feature vectors with a ratio of 1:3 was selected since it achieved the highest sentence accuracy in both methods.

Testing NARX with some of the feature extraction techniques, the maximum achieved sentence recognition rate was 81.6% with (Raw|mean|SD) resampled feature vectors with a ratio of 1:3; as compared with MKNN that showed a much higher set of accurate results reaching to 98.6% with the same set of feature vectors. Besides, for this case MKNN requires only 12 seconds for the training and classification compared with 418 seconds consumed by NARX to build and test its model. Due to the complexity NARX and its heavy computations, some limitations were imposed such as the number of neurons, the choice of the training function, and the size of the dataset. It is clear that MKNN outperforms NARX in terms of accuracy and computational complexity.

5.2 Vision-based Results

In this section, MKNN is used to classify the reviewed vision-based Arabic sign language dataset. Thereafter, the recognition rates are compared against those obtained by HMMs in [13].

To present the same portion of data to the classifier as the one used for HMMs, 70% of the dataset was used for the training and 30% for testing. Then, different combinations of the context, filtering, and word threshold parameters were examined. Besides, the variable k was set to three while *Cityblock* distance was selected as the distance metric. In addition, two nearest neighbors were considered from each point in the context ($m = 2$). The highest sentence recognition rate was found to be 71% with $ModeW = 6$, $MedW = 13$, and $WordTh = 7$. The results of changing these three parameters on the sentence recognition rate are shown in Figures 5.14 - 5.16.

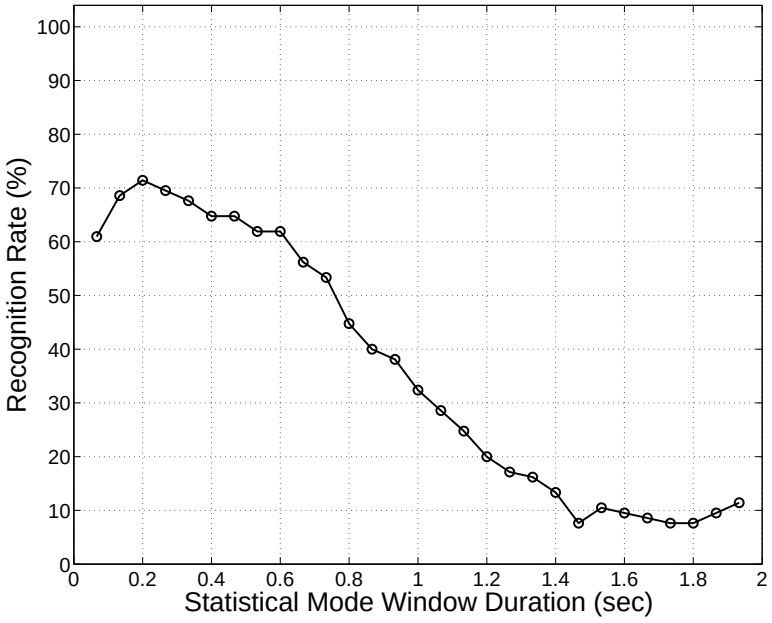


Figure 5.14: Vision-based Sentence Recognition Rate vs. Mode Window

As is noticed from these figures, shorter durations are more suitable for this data since its sampling rate is lower than that for sensor-based data. This is so because the

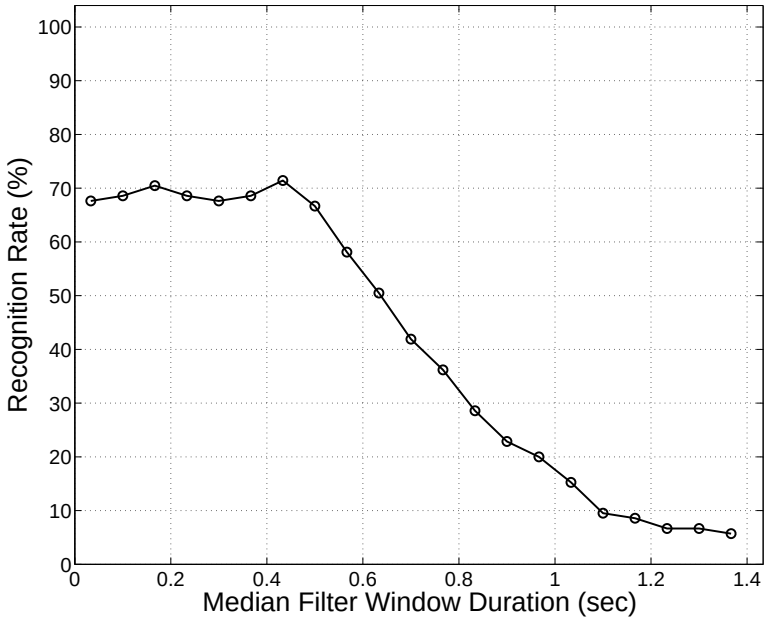


Figure 5.15: Vision-based Sentence Recognition Rate vs. Median Window

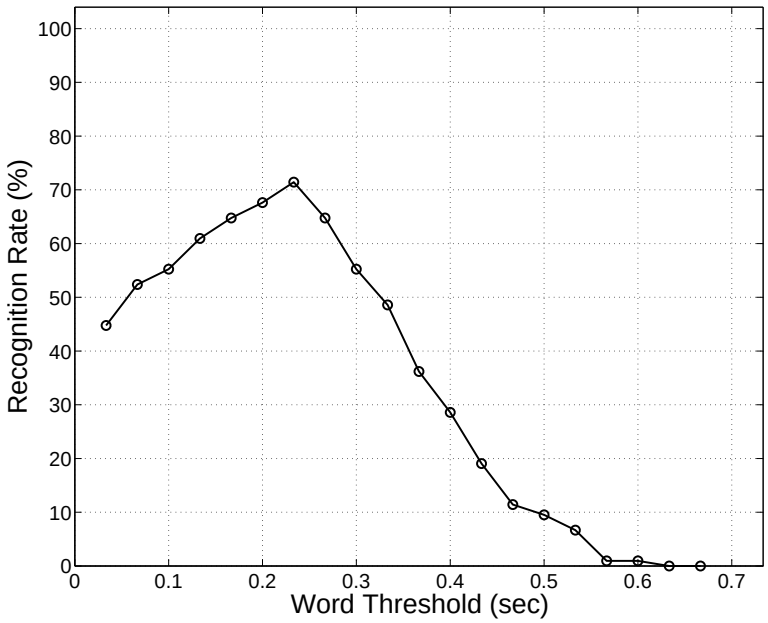


Figure 5.16: Vision-based Sentence Recognition Rate vs. Word Threshold

vision-based data was extracted using a window-based accumulated differences solution as explained in section 3.3 above. Hence, further windowing and grouping of data is not needed.

The resultant accuracy is comparable to the one reported in [13] using HMMs, where an average of 75% sentence recognition was obtained. However, MKNN has a low computational complexity and requires a simple setup, unlike HMM that requires several runs to set up its parameters, in addition to, its complex algorithms and tools.

As a final point, the sensor-based approach to sign language recognition using the proposed solution resulted in much higher accuracy than the vision-based counterpart. This is true even if HMMs were used in the modeling and recognition of the vision-based sign language. It is known that vision-based systems have limitations in terms of image segmentation, distance from the camera, scene variation, luminance variations, and so forth. All of these limitations do not apply to the proposed senor-based solution.

Chapter 6: Conclusion and Future Work

This thesis proposed a framework for continuous Arabic sign language recognition based on the data acquired from two data gloves. The ultimate goal is to facilitate a way of communication between hearing-impaired individuals and other community members. Equally important, we proposed a low-complexity classifier that can serve as an alternative for commonly used sequence-based classifiers such as hidden Markov models (HMMs), which require heavy computations and are usually difficult to setup. Before training the classifier, two DG5-VHand 2.0 gloves were used to generate the data. Next, manual labeling was carried out to complete the dataset. Then, some preprocessing and feature extraction techniques were employed. Finally, the resultant feature vectors were used to train the proposed classifier and report the obtained recognition rates.

This work investigated dynamic and statistical feature extraction techniques and showed that the latter has better impact on the overall recognition rate, since it generates more representative features. The proposed framework results indicate that our Modified k-Nearest Neighbor classifier (MKNN) outperformes other sequence-based models such as NARX, the nonlinear autoregressive network with exogenous input, and HMMs.

In summary, for the sensor-based dataset, MKNN achieved a high sentence recognition rate close to 99% as compared with 82% obtained from NARX model. As well, in terms of time complexity, the MKNN model performed much faster and efficiently than NARX model.

Moreover, we conducted an evaluation of MKNN with the vision-based dataset introduced by Assaleh et al. in [13]. The achieved sentence accuracy was close to that reported by HMMs; however, MKNN is simpler and faster.

Lastly, we intend to make the sensor-based continuous ArSL dataset and its labels publicly available to the research community.

To conclude, the sensor-based sign language recognition approach using the proposed framework outperforms the vision-based counterpart even when modeled with HMMs. This is due to the limitations of vision-based systems such as image segmentation, distance from the camera, as well as background and brightness variations. These limitations do not apply our proposed system.

In future work, it would be useful to extend the dataset by including data from more than one user and enlarging its vocabulary. Moreover, imposing some grammatical rules of Arabic sign language on the resultant predictions can improve the overall system's accuracy. It is worth mentioning that manual data labeling is time consuming; therefore, it would be desirable to develop an automatic labeling mechanism that can be essential for larger datasets.

As well, more feature extraction techniques should be investigated in an attempt to generate more representative features. Finally, it would be beneficial to devise an online recognition system that translates signs into text or speech.

References

[1] M. Mohandes, M. Deriche, and J. Liu, "Image-based and sensor-based approaches to Arabic sign language recognition," *IEEE Transactions on Human-Machine Systems*, vol. PP, no. 99, pp. 1–7, 2014.

[2] S. Ong and S. Ranganath, "Automatic sign language analysis: a survey and the future beyond lexical meaning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 873–891, 2005.

[3] O. Al-Jarrah and A. Halawani, "Recognition of gestures in Arabic sign language using neuro-fuzzy systems," *Artificial Intelligence*, vol. 133, no. 12, pp. 117–138, 2001.

[4] K. Assaleh and M. Al-Rousan, "Recognition of Arabic sign language alphabet using polynomial classifiers," *EURASIP Journal on Applied Signal Processing*, vol. 2005, pp. 2136–2145, Jan. 2005.

[5] O. Al-Jarrah and F. A. Al-Omari, "Improving gesture recognition in the Arabic sign language using texture analysis," *Applied Artificial Intelligence*, vol. 21, no. 1, pp. 11 – 33, 2007.

[6] M. Abul-Ela, A. Samir, and M. Tolba, "Neutralizing lighting non-homogeneity and background size in PCNN image signature," in *Proceedings of the International Conference on Computer Engineering Systems*, ICCES 2011, pp. 47–52, Nov. 2011.

[7] I. Elhenawy and A. Khamiss, "The design and implementation of mobile Arabic fingerspelling recognition system," *International Journal of Computer Science and Network Security*, vol. 14, no. 2, pp. 149–155, 2014.

[8] M. AL-Rousan, K. Assaleh, and A. Talaa, "Video-based signer-independent Arabic sign language recognition using hidden Markov models," *Applied Soft Computing*, vol. 9, no. 3, pp. 990–999, 2009.

[9] T. Shanableh, K. Assaleh, and M. Al-Rousan, "Spatio-temporal feature-extraction techniques for isolated gesture recognition in Arabic sign language," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 3, pp. 641–650, 2007.

[10] T. Shanableh and K. Assaleh, "User-independent recognition of Arabic sign language for facilitating communication with the deaf community," *Digital Signal Processing*, vol. 21, pp. 535–542, July 2011.

[11] T. Shanableh and K. Assaleh, "Two tier feature extractions for recognition of isolated Arabic sign language using Fisher's linear discriminants," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2, pp. II-501–II-504, 2007.

[12] T. Shanableh and K. Assaleh, "Telescopic vector composition and polar accumulated motion residuals for feature extraction in Arabic sign language recognition," *Journal on Image and Video Processing*, vol. 2007, no. 2, pp. 9–9, 2007.

[13] K. Assaleh, T. Shanableh, M. Fanaswala, F. Amin, and H. Bajaj, "Continuous Arabic sign language recognition in user dependent mode," *Journal of Intelligent Learning Systems and Applications*, vol. 2, no. 1, pp. 19–27, 2010.

[14] Q. Munib, M. Habeeb, B. Takruri, and H. A. Al-Malik, "American sign language (ASL) recognition based on Hough transform and neural networks," *Expert Systems with Applications*, vol. 32, no. 1, pp. 24–37, 2007.

[15] Z. Zafrulla, H. Brashear, T. Starner, H. Hamilton, and P. Presti, "American sign language recognition with the kinect," in *Proceedings of the 13th international conference on multimodal interfaces*, ICMI '11, pp. 279–286, ACM, 2011.

[16] C. Sun, T. Zhang, B.-K. Bao, C. Xu, and T. Mei, "Discriminative exemplar coding for sign language recognition with kinect," *IEEE Transactions on Cybernetics*, vol. 43, pp. 1418–1428, Oct 2013.

[17] A. Karami, B. Zanj, and A. K. Sarkaleh, "Persian sign language (PSL) recognition using wavelet transform and neural networks," *Expert Systems with Applications*, vol. 38, no. 3, pp. 2661–2667, 2011.

[18] B. Bauer, H. Hienz, and K. F. Kraiss, "Video-based continuous sign language recognition using statistical methods," in *Proceedings of the 15th International Conference on Pattern Recognition*, vol. 2, pp. 463–466 vol.2, 2000.

[19] M. Oszust and M. Wysocki, "Polish sign language words recognition with kinect," in *Proceedings of the 6th International Conference on Human System Interaction*, pp. 219–226, Jun. 2013.

[20] T. Bui and L. Nguyen, "Recognizing postures in Vietnamese sign language with MEMS accelerometers," *IEEE Sensors Journal*, vol. 7, no. 5, pp. 707–712, 2007.

[21] M. Mohandes, S. A-Buraiky, T. Halawani, and S. Al-Baiyat, "Automation of the Arabic sign language recognition," in *Proceedings of the International Conference on Information and Communication Technologies: From Theory to Applications*, pp. 479–480, 2004.

[22] M. Mohandes and S. Buraiky, "Automation of the Arabic sign language recognition using the Power Glove," *ICGST International Journal on Artificial Intelligence and Machine Learning*, vol. 7, no. 1, pp. 41–46, 2007.

[23] M. Mohandes and M. Deriche, "Arabic sign language recognition by decisions fusion using Dempster-Shafer theory of evidence," in *Proceedings of the Computing, Communications and IT Applications Conference*, pp. 90–94, Apr. 2013.

[24] M. A. Mohandes, "Recognition of two-handed Arabic signs using the CyberGlove," *Arabian Journal for Science and Engineering*, vol. 38, no. 3, pp. 669–677, 2013.

[25] K. Assaleh, T. Shanableh, and M. Zourob, "Low complexity classification system for glove-based Arabic sign language recognition," in *Proceedings of the 19th International Conference on Neural Information Processing*, ICONIP'12, pp. 262–268, 2012.

[26] W. Kong and S. Ranganath, "Towards subject independent continuous sign language recognition: A segment and merge approach," *Pattern Recognition*, vol. 47, no. 3, pp. 1294–1308, 2014.

[27] W. Gao, G. Fang, D. Zhao, and Y. Chen, "A Chinese sign language recognition system based on SOFM/SRN/HMM," *Pattern Recognition*, vol. 37, no. 12, pp. 2389–2402, 2004.

[28] X. Zhang, X. Chen, Y. Li, V. Lantz, K. Wang, and J. Yang, "A framework for hand gesture recognition based on accelerometer and EMG sensors," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 41, no. 6, pp. 1064–1076, 2011.

[29] P. Vamplew and A. Adams, "Recognition of sign language gestures using neural networks," *Australian Journal of Intelligent Information Processing Systems*, vol. 5, no. 2, pp. 94–102, 1998.

[30] J.-S. Kim, W. Jang, and Z. Bien, "A dynamic gesture recognition system for the Korean sign language (KSL)," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, pp. 354–359, Apr. 1996.

[31] R.-H. Liang and M. Ouhyoung, "A real-time continuous gesture recognition system for sign language," in *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 558–567, Apr. 1998.

[32] DGTech Engineering Solutions, "DG5 VHand 3.0 OEM Technical Datasheet," October 2013.

[33] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. New York: Wiley, 2001.

[34] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*. Amsterdam: Morgan Kaufmann, 3 ed., 2011.

[35] P. A. Flach, *Machine learning: the art and science of algorithms that make sense of data*. Cambridge, UK: Cambridge University Press, 2012.

[36] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37, no. 3, pp. 311–324, 2007.

[37] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, pp. 257–286, 1989.

[38] L. Rabiner and B.-H. Juang, *Fundamentals of speech recognition*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[39] DGTech Engineering Solutions, "DG5 VHand 2.0 OEM Technical Datasheet," November 2007.

[40] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*. San Diego, CA, USA: Academic Press Professional, Inc., 1990.

[41] T. Westeyn, H. Brashear, A. Atrash, and T. Starner, "Georgia Tech Gesture Toolkit: Supporting experiments in gesture recognition," in *Proceedings of the 5th International Conference on Multimodal Interfaces*, ICMI '03, pp. 85–92, ACM, 2003.

[42] L. C. Jain and L. R. Medsker, *Recurrent Neural Networks: Design and Applications*. Boca Raton, FL, USA: CRC Press, Inc., 1st ed., 1999.

[43] O. De Jesus, J. Horn, and M. Hagan, "Analysis of recurrent network training and suggestions for improvements," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 4, pp. 2632–2637, 2001.

[44] K. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, pp. 4–27, Mar 1990.

[45] The MathWorks Inc., "Neural Network Toolbox™ 6 Users Guide," 2009.

[46] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the RPROP algorithm," in *IEEE International Conference on Neural Networks*, pp. 586–591 vol.1, 1993.

Vita

Noor Tubaiz was born on December 14, 1989, in Baghdad, Iraq. In 2008, she moved to the United Arab Emirates and enrolled in the University of Sharjah, from which she received her Bachelors degree in Computer Engineering with honors. Then, she received a graduate teaching assistantship from the Department of Computer Science and Engineering at the American University of Sharjah and joined the Computer Engineering Masters Program in 2012. Her research interests include pattern recognition, data mining, computer security, and integrated circuit design.