computers

MDPI

Article

Comparison of Four SVM Classifiers Used with Depth Sensors to Recognize Arabic Sign Language Words†

Miada A. Almasre * and Hana Al-Nuaim

Department of Computer Science, Faculty of Computing and Information Technology, King AbdulAziz University, Jeddah 21499, Saudi Arabia; hnuaim@kau.edu.sa

* Correspondence: malmasre@kau.edu.sa; Tel.: +966-505579332

† This paper is an extended version of our published paper: Almasre, M.A.; Al-Nuaim, H. Recognizing Arabic Sign Language gestures using depth sensors and a KSVM classifier. In Proceedings of the 8th Computer Science and Electronic Engineering Conference (CEEC 2016), Colchester, UK, 28–30 September 2016.

Academic Editor: Laith Al-Jobouri
Received: 14 April 2017; Accepted: 12 June 2017; Published: 15 June 2017

Abstract: The objective of this research was to recognize the hand gestures of Arabic Sign Language (ArSL) words using two depth sensors. The researchers developed a model to examine 143 signs gestured by 10 users for 5 ArSL words (the dataset). The sensors captured depth images of the upper human body, from which 235 angles (features) were extracted for each joint and between each pair of bones. The dataset was divided into a training set (109 observations) and a testing set (34 observations). The support vector machine (SVM) classifier was set using different parameters on the gestured words' dataset to produce four SVM models, with linear kernel (SVMLD and SVMLT) and radial kernel (SVMRD and SVMRT) functions. The overall identification accuracy for the corresponding words in the training set for the SVMLD, SVMLT, SVMRD, and SVMRT models was 88.92%, 88.92%, 90.88%, and 90.884%, respectively. The accuracy from the testing set for SVMLD, SVMLT, SVMRD, and SVMRT was 97.059%, 97.059%, 94.118%, and 97.059%, respectively. Therefore, since the two kernels in the models were close in performance, it is far more efficient to use the less complex model (linear kernel) set with a default parameter.

Keywords: depth sensor; gesture recognition; support vector machine; classification; linear; radial; SVM

1. Introduction

Arabic Sign language (ArSL) resembles other forms of sign language for different spoken languages. ArSL is a multi-dialect sign-language system that is used by a considerable number of the hearing impaired in the Arab world, and it facilitates communication using hand gestures.

Understanding the meaning of sign language is difficult, so having an interpreter to translate sign-language gestures is inevitable. Due to the limited number of available ArSL interpreters for the different dialects, an automatic hand-gesture-recognition system is needed that can translate the sign language into text or audio in real time.

The available literature shows that the American and British Sign Languages are the most researched by far, with the American variant being the most influential [1]. In contrast, ArSL is still lagging behind in the literature.

As is the case with all sign languages, the automatic recognition of ArSL poses some problems, because the letters of ArSL are signed using gestures that are not always detected easily. Sometimes, there are difficulties with reading the hand's joints, the palm, and the fingertips. Therefore, it is important to determine the values (features) that could be used to recognize the meaning of the gestures. Human skeleton bone direction and joint or hand positions for the body parts in general are valid features that give high recognition accuracy [2].

Figure 1 presents some ArSL words that have different meanings as well as recognition issues. For example, for words one and two in Figure 1 ("cruel" and "giant", respectively), recognition will give the same meaning if it relies on the finger poses, but the two words will be recognized as distinctive when considering the elbows' angle.

By comparison, the gestures for words three and four in Figure 1 ("plate" and "tower", respectively) can be recognized easily by the elbow's bone angle, even though the gestures have the same finger pose (all of the finger bones have the same orientation in both gestures). Therefore, the same finger pose could cause an incorrect recognition between each of the two words, but the directions of the other arm bones, like the shoulders or elbows, could make the recognition more accurate. In addition, word five in Figure 1 ("objection") has the fingers attached together, which are also attached with the palm, so it is difficult to recognize each finger, or even the hand.

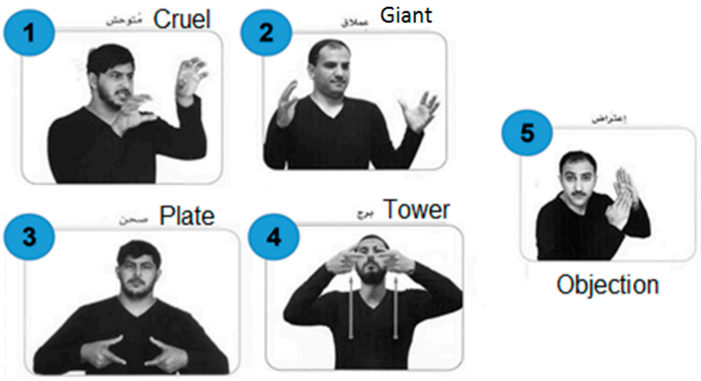


Figure 1. Five Arabic Sign language (ArSL) Words: 1—Cruel, 2—Giant, 3—Plate, 4—Tower, 5—Objection.

This makes the accurate recognition of imperceptible hand gestures a major research concern. It is assumed that digital gloves or marker-based techniques are possible solutions, but the unnatural quality of such interactions (people having to wear extra equipment on their hands) makes these solutions impractical to realize in real life [3]. However, it seems that opportunities for research in this field have increased with the introduction of cheap depth sensors.

Depth sensors provide essential data about each object or human in close proximity to the sensors, which will help extract a user's hand and body features. In real life, hand- and finger-recognition systems need to extract extra features and use complex procedures to recognize and interpret gestures accurately [3]. Among these devices are currently used ones like the Senz3D, Orbbec Persee, Microsoft Kinect, and Leap Motion Controller (LMC). Basically, these depth sensors project infrared beams, which bounce off a user standing in front of them. The infrared sensor uses time to measure the user's distance from the device [4].

Kinect, however, cannot accurately detect the movement and details of fingers and joints. To overcome such limitations, researchers use Kinect with a combination of image processing software, classifier algorithms, and software development kit (SDK) libraries, as in [5], or use Kinect with hardware such as electronic gloves, or integrate it with another sensor device to detect fingers and joints, as in [2].

Al-Masre and Al-Nuaim in [2] applied a supervised machine learning (a concept that matches through similarity) hand-gesturing model on a small dataset of 224 gestured letters to recognize the 28 letters of the Arabic alphabet [2]. Their research provided a true representation of ArSL when they increased the dataset sample from 224 to 1400 gestured letters [6]. Therefore, as the dataset was large, they used the kernel support vector machine (KSVM) as the supervised learning algorithm, with the radial kernel set with two parameters [6]. In addition, to overcome the time complexity of interpreting data for their model, the authors used the principle component analysis (PCA) algorithm to simplify the large dataset by reducing features and deleting redundant, irrelevant, or erroneous data due to noise [6].

In addition, Al-Masre and Al-Nuaim in [2–6] applied a KSVM algorithm as a supervised machine learning classifier to recognize the hand-gesturing movement and to classify it as the letter to which it belongs, which achieved 86% accuracy for recognizing ArSL letters [2–6].

For the purposes of this research, a new dataset for ArSL words—as opposed to letters—was collected using the same prototype as in [6]. However, to overcome the complexity of using a support vector machine (SVM) with only the radial kernel, the SVM classifier was used with different kernels and parameters to classify a five-word ArSL dataset, with each word considered a class (Figure 1).

Therefore, the objective of this paper is to compare the accuracy of the classifier with each parameter setting for each kernel.

2. Related Work

Sign language syntax can be defined as a combination of word order and non-manual markers, such as head tilting and shoulder raising, that add to the hand signs to create a meaning [7–11]. Hence, among the major problems a researcher encounters while developing gesture-recognition systems are those related to syntax. Similar problems exist in Arabic Sign Language, for which only a few linguistic studies have been conducted to account for its phonological, morphological, and syntactical aspects [8].

Technologies with new sensors and machine-learning approaches may help to create an auto-recognition sign-language system to compensate for the lack of a manual for Arabic Sign Language [8].

Machine learning is "an evolving branch of computational algorithms that are designed to emulate human intelligence by learning from the surrounding environment" [9,10]. Thus, machine learning is more than just calculating averages or performing data manipulation; it involves creating predictions about observations based on previous information [10].

Machine learning is critical to interpreting sign language and translating it into text. The concept of machine learning is divided into two main categories: supervised and unsupervised learning [11]. In supervised learning, the machine has prior knowledge of the characteristics of the dataset, which is usually divided into training and testing phases. In unsupervised learning, the machine does not have any prior information. Each learning style involves applying different algorithms to classify the datasets. However, the supervised method is more accurate, especially when dealing with larger datasets.

Machine learning for gesture recognition involves four steps: (1) choosing the appropriate sensors to collect the gestured letters; (2) analyzing and extracting features—which are values related to describing gestured letters—from the data; (3) classifying the data by recognizing and interpreting gestures using one or multiple algorithms; and (4) displaying the recognized gesture's name by text or audio [12].

The observed gestured letters are segmented into different classes based on the same or related values [13]. During classification, the collected data are segmented into two sets: a training and a testing set [13]. Thus, classification is the process of assigning a new observation to a specific class on the basis of training set values.

Generally, there are many classifier algorithms, such as the neural network, support vector machine (SVM), nearest neighbor (kNN), and random forest (RF) algorithms, with each having a different method to predict or choose the set to which a particular observation belongs [11].

By applying machine learning to classification, it has become easier—with the development of depth cameras and sensors—to make three-dimensional molding more accurate in identifying the individual body parts of a natural-looking human [14].

Mohandes, Deriche, and Liu in [15] used an LMC device in a system to recognize signs of the Arabic alphabet. A single volunteer gestured the 28 letters of the Arabic alphabet 10 times each, for a total of 280 (10 × 28) frames of data. The authors used two back-propagation algorithms to test the accuracy of the sign recognition. The first was a naive Bayes classifier with an accuracy of 98.3%, and the second was a multilayer perceptron with an accuracy of 99.1%. In their conclusion, they recommended the use of two sensors for more accuracy [15].

Ahmed et al. in [16] developed a system for sign-language recognition. A statistical analysis of the data from the collected images was provided as an input to an SVM. The results of that system for 10 letters reached an accuracy rate of 83%. They recommended further enhancements to construct a real-time static and dynamic gesture-recognition system [16].

Aujeszky and Eid in [17] introduced a framework for ArSL communication systems using Microsoft's Kinect. The proposed framework is a two-part real-time communication system with real-time feedback about the signer's performance via real-time avatar animation. The results had a 96% detection rate, and the average time to complete an Arabic sign was about 2.2 s [17].

Marin, Dominio, and Zanuttigh in [18] explained how to exploit two types of sensors for accurate real-time gesture recognition. They used LMC for hand gesture recognition, which provides only a limited set of relevant points, and a depth camera to obtain a complete three-dimensional description. Their captured feature sets were fed to a multi-class SVM classifier. The accuracy recognition results were 80.9% when they used LMC only, 96.35% when they used only the depth camera, and 96.5% when they combined the two [18].

3. Gesture Recognition Pipeline

The process of the gesture recognition pipeline starts by inputting unclassified data into the devices and ends with output information about the data class to which the data belong [11].

3.1. Input (via Sensors)

In this research, Kinect™ and Leap Motion Controller (LMC) sensors were used to create a model to recognize ArSL gestures. Microsoft Kinect Version 2.0 has a depth camera, voice recognition, face-tracking capabilities, and provides access to the raw sensor records [11]. Kinect has an open-source SDK for developing applications; it also has a wider sensing range that can track a complete skeleton, with joint points. Figure 2 shows the upper human joint points that Kinect can detect and track [4].

LMC Version 2.0 provides a skeletal-tracking algorithm, which offers information about hands and fingers as well as overall hand-tracking data, even if the hands cross over each other (Figure 3). LMC uses an application programming interface (API) that measures different units of physical quantities, such as measuring distance using millimeters, time using microseconds, and an angle using radians [19].

Like Kinect, LMC has an open-source library (SDK Leap Version 2), which provides APIs that retrieve many details about the hands, such as the direction of each finger or bone.

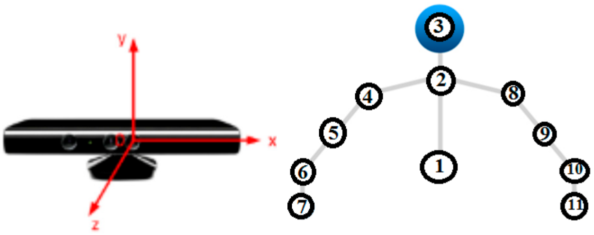


Figure 2. The upper human body joint points that Kinect detects. 1—Spin, 2—Shoulder Center, 3—Head, 4—Right Shoulder, 5—Right Elbow, 6—Right Wrist, 7—Right Hand, 8—Left Shoulder, 9—Left Elbow, 10—Left Wrist, 11—Hand Left.

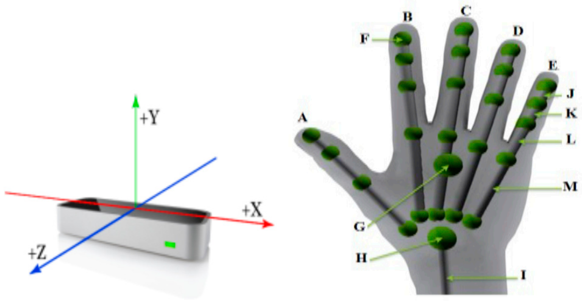


Figure 3. Hand skeleton points that Leap Motion Controller (LMC) retrieves based on Cartesian coordinate system. A—Thumb, B—Index, C—Middle, D—Ring, E—Pinky, F—Tip Position, G—Palm Position, H—Wrist Position, I—Arm, J—Distal Bone, K—Intermediate Bone, L—Proximal Bone, M—Metacarpal Bone.

3.2. Feature Extraction

A feature in computer vision means a piece of information that is related to explaining a certain part of data or computational tasks that are connected to a specific part in an image [20]. Therefore, in machine learning and pattern recognition, the preprocessing generally starts by extracting a vector of features that help the system recognize the image [20]. Features could be points, edges, and directions of certain points in an object, such as the direction of hand bones [20]. Feature extraction is an important process that requires certain classification algorithms, especially when the input data are too large. Thus, the analysis of complex data is one of the main difficulties, due to the number of variables involved. Consequently, feature extraction is used to reduce the representation of features in an image, instead of in the complete initial data set. The extracted features are predictable, to cover all of the relevant information from the input data [20]. As mentioned above, the Kinect and LMC sensors retrieve depth data of a human body's skeleton, which yield precise features that can be used to enhance any gesture-recognition system [21].

3.3. Classification Using Support Vector Machine (SVM)

In machine learning, the classification process provides the classifier with accurate data, in which the right answers for each class are produced; it checks whether this classifier can predict the class of new data [22]. In other words, the classification process divides the collected dataset into two subsets: training and testing sets. Then, it implements the following steps:

The training step: based on the training set observations, for which each input is mapped to the output, the researchers used classification to estimate a predictive model (classifier) that can map or generate output to an arbitrary input.

The testing step: the classifier can then label the unseen observations in the testing set into a class. Therefore, the testing step assesses the classifier's (the predictive model's) ability to predict the class name of the unseen data correctly.

There are many classifiers, and each one uses different parameters to predict and choose the right class. The SVM performs classification by finding the hyperplane that maximizes the margin between the two classes. The vectors (cases) that define the hyperplane are the support vectors [23]. The SVM algorithm is used to classify data by drawing a clear line between observation data, which are actually points on a plane. The margin space around the line should be as wide as possible to avoid misclassified values in the testing set [24]. With a multi-dimensional space, the line becomes a plane or a hyper plane [24]. In addition, the SVMs can efficiently perform nonlinear classification using what is called the kernel function (the linear and radial functions are the popular kernels used) to implicitly map its inputs into high-dimensional feature spaces [25]. The kernel function transforms the data into a higher-dimensional feature space to make it possible to perform the linear separation [24]. A kernel basis function can be defined mathematically as follows:

If there are two observations x and x', represented as feature vectors in some input space, the radial basis kernel function is defined as:

$$K(x, x') = \exp\,(-\frac{(x - x')^2}{2\partial^2}) \tag{1}$$

where $(x - x')^2$ represents the squared Euclidean distance between the two features, and ∂ is the variance [26].

Moreover, a kernel method helps learning algorithms to learn a nonlinear function or decision boundary, but it does not require the explicit mapping to be computed. For all x and $x0$ in the input space X, there are functions $k(x, x0)$ that represent the inner product in another space V. The function k: X × X → V is called a kernel function and can be defined mathematically as follows:

$$k(x, x0) = \langle \varphi(x), \varphi(x0) \rangle_{\eta}. \tag{2}$$

An explicit representation for φ is not required where the choice for the function φ is arbitrary, as long as it satisfies data accuracy. Popular choices of φ include radial, polynomial, linear, and Gaussian functions [26].

Moreover, predicting the values and setting the parameters with correct values are the main objectives of the SVM learning algorithm. Many statistical packages set those parameters to give the best prediction, such as the R studio statistical package [27].

Additionally, using an SVM requires not only choosing the kernel function, but also the parameter C (cost function) or a penalty term [26]. This parameter is used because an SVM relies on predictions to make a decision about the best boundary that could cause an error [26]. If the value of C is very large, then the decision boundary will be close to the data points nearest to the support vectors. This means that the misclassification probability increases as the value of C decreases [27].

A confusion matrix (CM) algorithm was used to evaluate the classifiers' performance [23]. The CM shows the number of accurate and inaccurate predictions made by the classification model compared to the actual outcomes (actual value) in the testing set [23].

4. Proposed Model

Sign language relies on different body parts, which necessitates the use of two sensors. This research used Kinect to recognize the user's whole skeleton without finger details (because Kinect cannot capture fingers details [28]), and the LMC to recognize the user's hand bones only, as it provides no details on the whole arm or the human body.

The proposed model's prototype used the following hardware and software:

- Kinect Version 2.0 with voice-recognition and face-tracking capabilities, and an RGB depth camera.
- The two sensors (Kinect and LMC) were connected to a personal computer running a 64-bit Windows 10 operating system, with an Intel® Core (TM) i7 2.5-GHz processor, and 16 GB RAM.
- Visual Studio 2013 with C# to calibrate the two sensors and capture the dataset. Figure 3 presents the 12 points of hand joints retrieved via LMC, while Figure 4 presents the 11 joints retrieved via Kinect.
- SDK Version 2 of Kinect and LMC with Windows Media3D to present a three-dimensional model of a human body's skeleton, which provided three-dimensional object transformation.
- SQL Server Management Studio to create a relational database to save the data and depth values captured by the two sensors. The two feature types in the database were:
    - Type one—denoted as "H" in the database—has three angles for each hand bone, which are angles between the bone and the three axes of the coordinate system (X, Y, Z).
    - Type two—denoted as "A" in the database—has one angle between each pair of bones, as shown in Figure 4. These angles are the main factor for a comparison between two gestures.

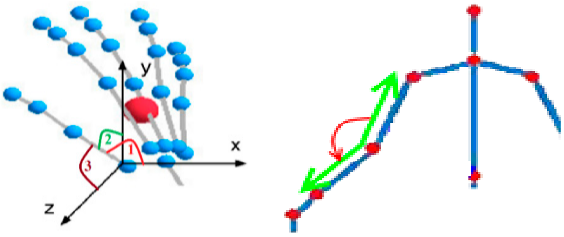


Figure 4. Example of three angles for one joint and one angle between two bones.

4.1. Feature Representation As Histograms

A histogram is a bar graph in which the data are organized into equal intervals. The intervals are represented as columns with the same width and varying lengths based on the frequency of the data value in each specific interval [29]. Statistical analyses with histograms allow the real distribution of data to be visualized, such as in the histogram of an image [30]. A histogram's x-axis and y-axis values could have any type of points. In the proposed model, the joints' points were presented using two types of features as a histogram.

The first histogram (3-angles for each hand joint) has 180 points on the x-axis, with each 3-angle representing one joint, and 200 points on the y-axis, which have angle values ranging between −100 and +100. For each joint point, a line is drawn between each corresponding point (vector). Each vector has a corresponding direction and an angle, and each angle was measured based on the corresponding x, y, and z angles. Thus, the calculated 3-angles (Ax, Ay, Az) represent each joint point (Figure 5). These angles are the main factor for a comparison between two gestures.

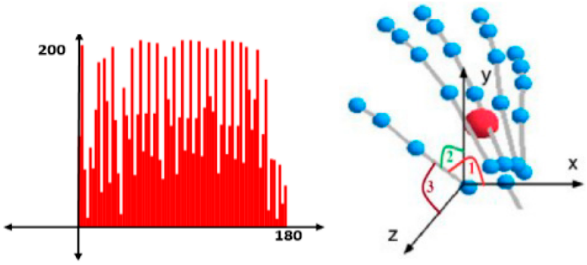


Figure 5. Histogram of three angles for one joint (3-angles).

The second histogram (1-angle) has 60 points on the x-axis, representing absolute angles between each pair of bones in the hand (vector), as shown in Figure 6.

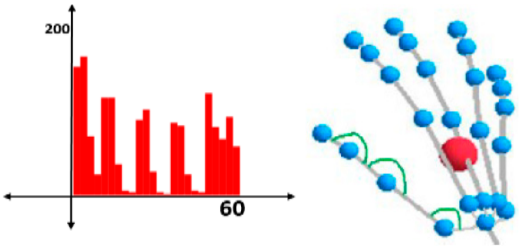
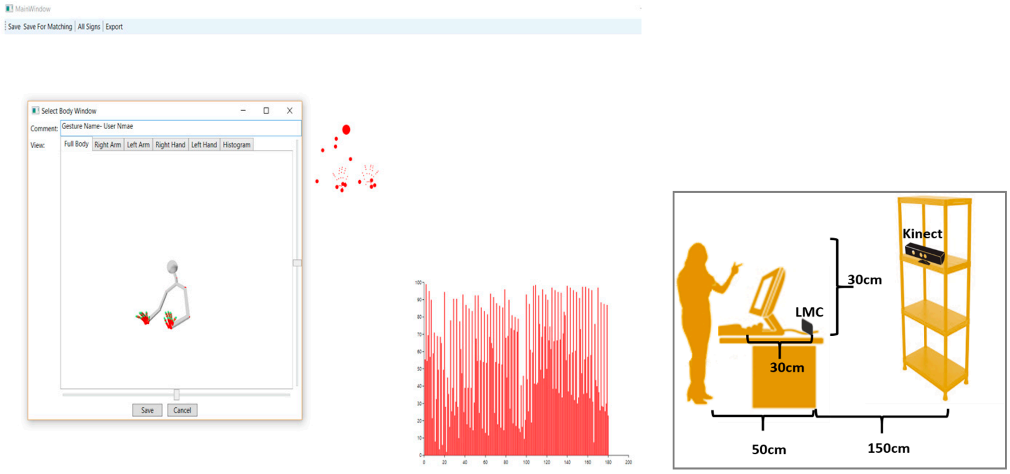


Figure 6. Histogram of one angle between two vectors (1-angle).

Therefore, the prototype was considered ready for use in the main window of the experimental environment (Figure 7a). The ten volunteer participants, who were females in their twenties, were knowledgeable in Arabic sign language; however, they were not deaf. Each participant was asked to gesture five words, repeating each gesture three times. Each participant stood in front of the devices—which were connected to a personal computer—and made 5 to 10 word gestures (Figure 7b).


MainWindow
Save Save For Matching | All Signs | Export
Select Body Window
Comment: Gesture Name- User Nmae
View: Full Body | Right Arm | Left Arm | Right Hand | Left Hand | Histogram
Save
Cancel
Kinect
30cm
LMC
30cm
50cm
150cm

(**a**) (**b**)

Figure 7. The prototype settings, (**a**) The Prototype Main Window, (**b**) Experimental Environment with Dimensions.

A total of 143 dataset observations were made, representing the five different ArSL words. Table 1 shows the number of observations and the proportion of each word (class) in the dataset.

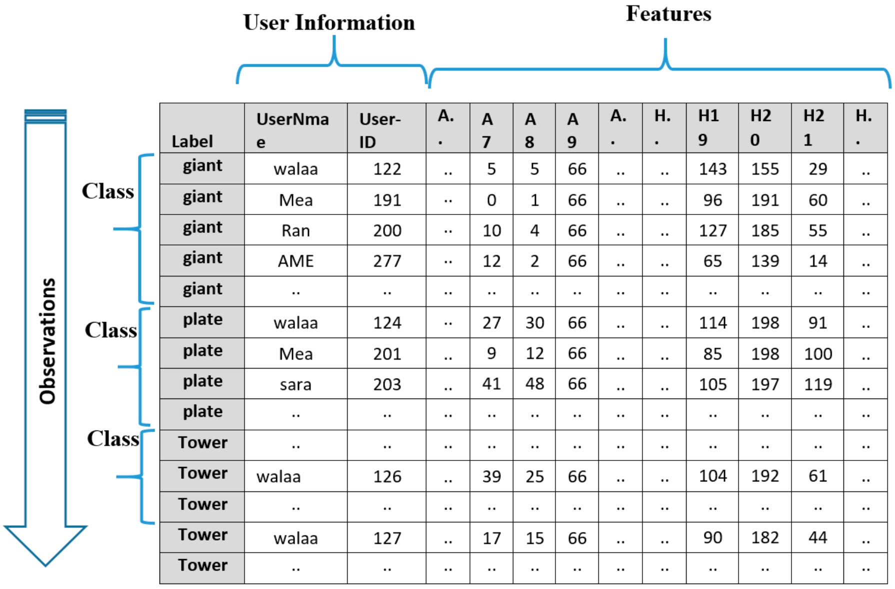
Table 1. Observation numbers.

Class Name (Words)	# of Observation	Class Proportion
Objection	9	6%
Tower	18	13%
Cruel	25	17%
plate	41	29%
giant	50	35%
Total	143	100%

4.2. Dataset Structure

The dataset observations are presented in Figure 8 as rows. Each observation was considered a word from a specific participant and contained many features.

The collected dataset has 235 features presented in Figure 8 as columns: the values H0 to H180 are from type one, and the values A1 to A54 are from type two. The dataset was reduced by selecting the body parts on which each gesture relies, while removing all of the values that will not affect the interpretation of the ArSL words, such as feature A9 (Figure 8).



Label	UserNmae	User-ID	A..	A7	A8	A9	A..	H..	H19	H20	H21	H..
giant	walaa	122	..	5	5	66	..	..	143	155	29	..
giant	Mea	191	..	0	1	66	..	..	96	191	60	..
giant	Ran	200	..	10	4	66	..	..	127	185	55	..
giant	AME	277	..	12	2	66	..	..	65	139	14	..
giant	..	..	..	..	..	..	..	..	..	..	..	..
plate	walaa	124	..	27	30	66	..	..	114	198	91	..
plate	Mea	201	..	9	12	66	..	..	85	198	100	..
plate	sara	203	..	41	48	66	..	..	105	197	119	..
plate	..	..	..	..	..	..	..	..	..	..	..	..
Tower	..	..	..	..	..	..	..	..	..	..	..	..
Tower	walaa	126	..	39	25	66	..	..	104	192	61	..
Tower	..	..	..	..	..	..	..	..	..	..	..	..
Tower	walaa	127	..	17	15	66	..	..	90	182	44	..
Tower	..	..	..	..	..	..	..	..	..	..	..	..

Figure 8. Dataset structure.

4.3. Classification Implementation

A preprocessing step was applied to remove all of the null values and any features with zero variance from the dataset. The dataset was divided into a training set with 109 observations and a testing set with 34 observations, with a splitting ratio of 75:25. The SVM parameters in the training step were set using either default values from the *R* package, or adjusted values (tuning). Tuning the SVM parameters means the process in which one or more parameters of the SVM model are adjusted up or down to achieve high-accuracy results [10].

Consequently, four SVM models were produced by setting the SVM with two kernel functions (linear and radial). For each kernel, the parameters were set by using the default R package values or by tuning the parameters with different values. Therefore, the model was trained by allowing the *R* package to choose the SVM parameters *C* and sigma. Then, custom values for *C* and sigma were chosen, to search for the best parameters with the highest recognition accuracy rate when training the model.

5. Results

The SVM classifier was applied using the linear and radial kernel functions. The accuracy performance for the four SVM models—SVMLD, SVMLT, SVMRD, and SVMRT—differed between the training set and the testing set:

- The accuracy performance for the training set differed among the two kernels. The highest accuracy among the four models for the training set was 90.884%, when the radial kernel was set with default parameters or tuned parameters. However, the accuracy was higher—88.92%—when the linear kernel was set with either default parameters or tuned parameters.
- The accuracy performance for the testing set also differed among the two kernels. The highest accuracy among the four models for the testing set was 97.059%, when the linear kernel was set with default parameters or tuned parameters. However, the accuracy was higher—at 88.92%—when the linear kernel was set with either default parameters or tuned parameters. The overall accuracy in recognizing the corresponding words in the testing set for the SVMLD, SVMLT, SVMRD, and SVMRT models was 97.059%, 97.059%, 94.118%, and 97.059%, respectively.

In this research, the SVM with a "linear" kernel performed better than the "radial" kernel for the testing dataset. The reason could be that the data had a large number of features (235), making the model complex.

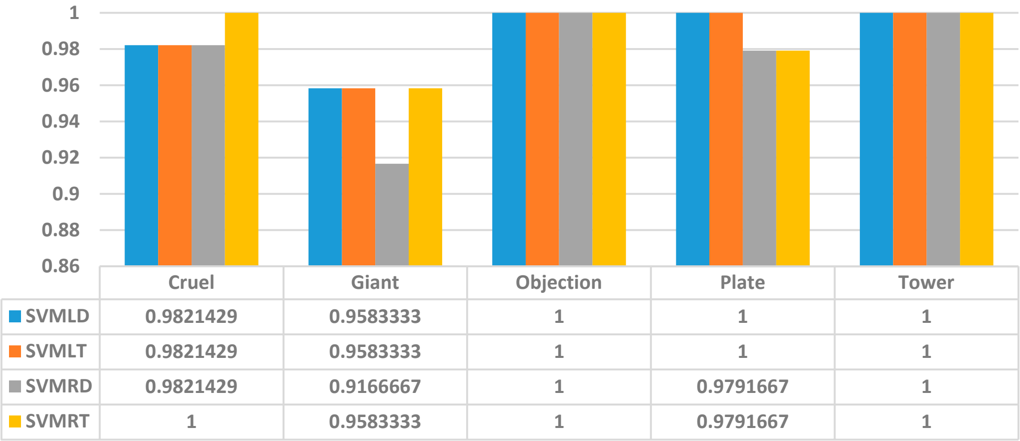
Increasing the model complexity by using a radial kernel could cause overfitting (the model was trained well yet cannot be generalized to a new dataset). Therefore, it would be better to balance the high degree of complexity due to the large number of features with the simplicity of using the linear kernel. However, both kernels were applied with the following parameters and kernel results in the training step:

- SVM with default parameters and linear kernel (SVMLD): an SVM with a linear kernel has only one parameter that needs to be set: the cost parameter. The default value in the *R* package is the cost parameter, $C = 1$, which gave an overall accuracy of 88.92% in the training step.
- SVM with tuned parameters and linear kernel (SVMLT): For an SVM with a linear kernel and tuning for C, trying different cost-parameter values resulted in almost the same training accuracy. When C was set to 0.001, 0.006, 0.011, 0.016, 0.021, 0.026, 0.031, 0.036, 0.041, and 0.046, the accuracy was 78.186%, 88.092%, 89.832%, 88.922%, 88.922%, 88.922%, 88.922%, 88.922%, 88.922%, and 88.922%, respectively. Thus, tuning the cost parameter did not actually increase accuracy, and in many cases the default parameter from the *R* package was a good choice, especially since the dataset has a large number of features, so the fine-tuning of hyper-parameters in this case was not advantageous.
- SVM with default parameters and radial kernel (SVMRD): The radial kernel needs two parameters to be set (cost and sigma). The default value in the R package for sigma was held constant at a value of 0.002964685, and $C = 4$, which gave an overall accuracy of 90.884% in the training step.
- SVM with tuned parameters and radial kernel (SVMRT): For an SVM with a radial kernel and the cost and sigma parameters tuned, Table 2 shows the accuracy in the training step for each pair of cost and sigma parameters. High accuracy was achieved when sigma was set to 0.003 and C was set to 3, 3.5, 4, 4.5, and 5.

Table 2. Tuning parameters.

C	Sigma	Accuracy	C	Sigma	Accuracy	C	Sigma	Accuracy
1	0.001	0.790951	2.5	0.001	0.827564	4	0.001	0.871833
1	0.002	0.801477	2.5	0.002	0.880528	4	0.002	0.898315
1	0.003	0.838486	2.5	0.003	0.900146	4	0.003	0.908841
1.5	0.001	0.790951	3	0.001	0.854046	4.5	0.001	0.889619
1.5	0.002	0.836655	3	0.002	0.889619	4.5	0.002	0.898315
1.5	0.003	0.874059	3	0.003	0.908841	4.5	0.003	0.908841
2	0.001	0.810173	3.5	0.001	0.871833	5	0.001	0.898315
2	0.002	0.871833	3.5	0.002	0.898315	5	0.002	0.898315
2	0.003	0.900146	3.5	0.003	0.908841	5	0.003	0.908841

The testing step evaluated the model's ability to predict the class label from the five classes for unseen data. The testing results for the SVMLD, SVMLT, SVMRD, and SVMRT models achieved 97.059%, 97.059%, 94.118%, and 97.059% accuracy, respectively (Figure 9).



	Cruel	Giant	Objection	Plate	Tower
🟦 SVMLD	0.9821429	0.9583333	1	1	1
🟧 SVMLT	0.9821429	0.9583333	1	1	1
⬜ SVMRD	0.9821429	0.9166667	1	0.9791667	1
🟨 SVMRT	1	0.9583333	1	0.9791667	1

Figure 9. Accuracy for each class during the testing step.

6. Conclusions and Future Work

The Arab hearing impaired face numerous hardships when it comes to interacting with others, including the lack of ArSL interpreters and the excessive variation among different ArSL dialects. A platform incorporating Kinect and LMC could certainly offer a better chance for a more interactive role in society among this population.

Moreover, less complexity is needed to enhance the recognition accuracy of ArSL when using a supervised machine-learning approach, whether by reducing the dataset features or by setting fewer parameters for the classifier. This research started off with the recognition of all ArSL letters, and was then extended to recognizing a few ArSL words to test the most efficient classifier algorithm.

This research combined two sensor devices to improve the recognition accuracy rate for ArSL by capturing not only the hand's skeleton, but all upper joints upon which most sign-language gestures rely. The depth values of the human skeleton captured by the two sensors extracted 235 angles for all of the upper human skeleton bones.

The significance of this research is that it compares the results of using an SVM classifier with different parameters. This paper presents four SVM models to examine 143 signs for five ArSL words gestured by 10 users. These gestured signs were considered a dataset, and were divided into a training set with 109 observations and a testing set with 34 observations.

The difference in the performance between the linear and radial kernels using default values was not significant: both had equal accuracy rates with the models using tuned values. Therefore, if the two kernels in the models perform equally, it is more efficient to use the less complex model (the linear kernel), which only needs to be set with only one parameter (C) to overcome the time complexity needed to interpret data for this model.

In conclusion, with equal accuracy performance, using a linear kernel set with only one parameter (C) would be less complex than using KSVM, as in [5], by setting the radial kernel of the SVM with two parameters (C and sigma).

The dataset used in this research is considered small; therefore, the results could be different between the linear and radial kernels with a large dataset.

Future work will involve attempts to recognize ArSL phrases using depth sensors with a supervised machine-learning approach, taking into consideration the limited workspace the LMC has for the user's movement. Moreover, Kinect and LMC both need enhancements in speed to capture and recognize gestures with higher accuracy and in real time.

Acknowledgments: This research is part of a PhD thesis by two faculty members. Only personal funding was used.

Author Contributions: M.A. Almasre collected the data for the experiment, while both authors designed the research, analyzed the data; and wrote the paper. Both authors have read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Capilla, D.M. *Sign Language Translator Using Microsoft Kinect XBOX 360 TM*; University of Tennesse: Knoxville, TN, USA, 2012.
2. Almasre, M.A.; Al-Nuaim, H. A Real-Time Letter Recognition Model for Arabic Sign Language Using Kinect and Leap Motion Controller v2. *Int. J. Adv. Eng. Manag. Sci.* **2016**, *2*, 514–523.
3. Liang, H.; Yuan, J. Hand Parsing and Gesture Recognition with a Commodity Depth Camera. In *Computer Vision and Machine Learning with RGB-D Sensors*; Shao, L., Han, J., Kohli, P., Zhang, Z., Eds.; Springer International Publishing: Cham, Germany, 2014; pp. 239–265.
4. Han, J.; Shao, L.; Xu, D.; Shotton, J. Enhanced Computer Vision with Microsoft Kinect Sensor: A Review. *IEEE Trans. Cybern.* **2013**, *43*, 1318–1334. [PubMed]
5. Almasre, M.; Al-Nuaim, H. Using the Hausdorff Algorithm to Enhance Kinect's Recognition of Arabic Sign Language Gestures. *Int. J. Exp. Algorithms IJEA* **2017**, *7*, 18.
6. Almasre, M.A.; Al-Nuaim, H. Recognizing Arabic Sign Language gestures using depth sensors and a KSVM classifier. In Proceedings of the 2016 8th Computer Science and Electronic Engineering (CEEC), Colchester, UK, 28–30 September 2016; pp. 146–151.
7. Metaxas, D.N.; Liu, B.; Yang, F.; Yang, P.; Michael, N.; Neidle, C. Recognition of Nonmanual Markers in American Sign Language (ASL) Using Non-Parametric Adaptive 2D-3D Face Tracking. In Proceedings of the LREC, Istanbul, Turkey, 21–27 May 2012; pp. 2414–2420.
8. Abdel-Fattah, M.A. Arabic Sign Language: A Perspective. *J. Deaf Stud. Deaf Educ.* **2005**, *10*, 212–221. [CrossRef] [PubMed]
9. Naqa, I.E.; Murphy, M.J. What Is Machine Learning? In *Machine Learning in Radiation Oncology*; Naqa, I.E., Li, R., Murphy, M.J., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 3–11.
10. Munoz, A. Machine Learning and Optimization. Available online: https://www.cims.nyu.edu/~munoz/files/ml_optimization.pdf (accessed on 14 June 2017).
11. Bengio, Y.; Courville, A.; Vincent, P. Representation Learning: A Review and New Perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1798–1828. [CrossRef] [PubMed]
12. Pisharady, P.K.; Saerbeck, M. Recent methods and databases in vision-based hand gesture recognition: A review. *Comput. Vis. Image Underst.* **2015**, *141*, 152–165. [CrossRef]
13. Erol, A.; Bebis, G.; Nicolescu, M.; Boyle, R.D.; Twombly, X. Vision-based hand pose estimation: A review. *Comput. Vis. Image Underst.* **2007**, *108*, 52–73. [CrossRef]
14. Ionescu, D.; Suse, V.; Gadea, C.; Solomon, B.; Ionescu, B.; Islam, S. An infrared-based depth camera for gesture-based control of virtual environments. In Proceedings of the 2013 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), Milan, Italy, 15–17 July 2013; pp. 13–18.
15. Mohandes, M.; Deriche, M.; Liu, J. Image-Based and Sensor-Based Approaches to Arabic Sign-language recognition. *IEEE Trans. Hum.-Mach. Syst.* **2014**, *44*, 551–557. [CrossRef]
16. Ahmed, H.; Gilani, S.O.; Jamil, M.; Ayaz, Y.; Shah, S.I.A. Monocular Vision-based Signer-Independent Pakistani Sign-language recognition System using Supervised Learning. *Indian J. Sci. Technol.* **2016**, *9*. [CrossRef]
17. Aujeszky, T.; Eid, M. A gesture recognition architecture for Arabic sign language communication system. *Multimed. Tools Appl.* **2016**, *75*, 8493–8511. [CrossRef]
18. Marin, G.; Dominio, F.; Zanuttigh, P. Hand gesture recognition with jointly calibrated Leap Motion and depth sensor. *Multimed. Tools Appl.* **2016**, *75*, 14991–15015.
19. Spiegelmock, M. *Leap Motion Development Essentials*; Packt Publishing: Birmingham, UK, 2013.
20. Gavrila, D.M. The Visual Analysis of Human Movement: A Survey. *Comput. Vis. Image Underst.* **1999**, *73*, 82–98. [CrossRef]
21. Srivastava, R. *Research Developments in Computer Vision and Image Processing: Methodologies and Applications: Methodologies and Applications*; IGI Global: Dauphin County, PA, USA, 2013.

22. Begg, R.; Kamruzzaman, J. A machine learning approach for automated recognition of movement patterns using basic, kinetic and kinematic gait data. *J. Biomech.* **2005**, *38*, 401–408. [CrossRef] [PubMed]
23. Sayad, S. Model Evaluation. An Introduction to Data Mining. Available online: http://www.saedsayad.com/model_evaluation_c.htm (accessed on 14 August 2016).
24. Burges, C.J. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.* **1998**, *2*, 121–167. [CrossRef]
25. Ben-Hur, A.; Horn, D.; Siegelmann, H.T.; Vapnik, V. Support vector clustering. *J. Mach. Learn. Res.* **2002**, *2*, 125–137. [CrossRef]
26. Strickland, J. *Predictive Analytics Using R*; Lulu, Inc.: Raleigh, NC, USA, 2015.
27. Learning Kernels SVM. Available online: https://www.r-bloggers.com/learning-kernels-svm/ (accessed on 14 June 2017).
28. Jana, A. *Kinect for Windows SDK Programming Guide: Build Motion-Sensing Applications with Microsoft's Kinect for Windows SDK Quickly and Easily*; Packt Publishing: Birmingham, UK, 2012.
29. Gravetter, F.J.; Wallnau, L.B. *Statistics for the Behavioral Sciences*; Cengage Learning: Boston, MA, USA, 2016.
30. Silverman, B.W. *Density Estimation for Statistics and Data Analysis*; CRC Press: Boca Raton, FL, USA, 1986.