INDIAN SIGN LANGUAGE NUMBERS RECOGNITION USING

INTEL REALSENSE CAMERA

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Sravani Mudduluru

May 2017

COMMITTEE MEMBERSHIP

TITLE: Indian Sign Language Numbers Recognition using Intel
                                RealSense Camera

AUTHOR: Sravani Mudduluru

DATE SUBMITTED: May 2017

COMMITTEE CHAIR: Franz Kurfess, Ph.D.

                                Professor of Computer Science

COMMITTEE MEMBER: Christopher Lupo, Ph.D.

                                Associate Professor of Computer Science

COMMITEE MEMBER: Hisham Assal, Ph.D.

                                Associate Professor of Computer Science

ABSTRACT

Indian Sign Language Numbers Recognition using Intel RealSense Camera

Sravani Mudduluru

The use of gesture based interaction with devices has been a significant area of research in the field of computer science since many years. The main idea of these kind of interactions is to ease the user experience by providing high degree of freedom and provide more interactive way of communication with the technology in a natural way. The significant areas of applications of gesture recognition are in video gaming, human computer interaction, virtual reality, smart home appliances, medical systems, robotics and several others. With the availability of the devices such as Kinect, Leap Motion and Intel RealSense cameras accessing the depth as well as color information has become available to the public with affordable costs.

The Intel RealSense camera is a USB powered controller that can be supported with few hardware requirements such as Windows 8 and above. This is one such camera that can be used to track the human body information similar to the Kinect and Leap Motion. It was designed specifically to provide more minute information about the different parts of the human body such as face, hand etc. This camera was designed to give users more natural and intuitive interactions with the smart devices by providing some features such as creating 3D avatars, high quality 3D prints, high-quality graphic gaming visuals, virtual reality and others.

The main aim of this study is to try to analyze hand tracking information and build a training model in order to decide if this camera is suitable for sign language. In this study, we have extracted the joint information of 22 joint labels per single hand .We trained the model to identify the Indian Sign Language(ISL) numbers from 0-9. Through this study we analyzed that multi-class SVM model showed higher accuracy of 93.5% when compared to the decision tree and KNN models.

ACKNOWLEDGMENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

With the advancement in technology the ways of user interaction with software has been changing from physically operating devices via keyboard to, interacting without any touch such as gestures, virtual reality etc. Gesture recognition has been main focus in recent times in the field of Computer Science in various areas. For example, Sign Language detection and translation, Robot control, medical systems, virtual environment, gaming, fitness etc. The main goal of gesture recognition is to build a software or a system that is capable of detecting and identifying particular human gestures in order to perform actions. This process is based on the mathematical analyzations of the gestures. Although much research work has been done since early 70's, there are a lot of improvements going on. Gesture recognition is a challenging task and it is very complex to understand because every human has their own way of expressing gestures .So this will result in same gesture being expressed in different ways. Different tools such as Data glove, Kinect sensor, Intel RealSense camera etc. have been developed in order to track the person's body movements to accurately detect the gestures. Traditional methods in computer vision are based on extracting the features by basically converting the 3D hand posture into a 2D image, and depend on template matching algorithms. With these methods there is lack of information that is needed for efficiently identifying the gestures. On the other hand, the information that is extracted directly from the camera in 3D is much more useful and efficient.

Sign language uses gestures to express the meaning by combining hand shapes, hand movements, facial expressions and lip patterns. Sign language is not only used by deaf people but also by people who cannot physically speak. Sign language translators

can help people to communicate with other people through their gestures. There is a wide number of sign languages all over the world. Out of all these, American Sign Language (ASL) is popularly known [2]. Every sign language has its own symbols, rules and grammar. Data gloves are used to detect hand symbols related to sign language with the help of sensors. The sensors present in data glove provide very accurate and high quality measurements about the hand movement. But they are not widely used because they are expensive. Moreover, they restrict the ease of user interaction because it needs the user to be physically connected with the device. Because of improved interactivity and user comfort, sign language recognition based on optical sensors with depth information has become more popular. Microsoft Kinect, Leap Motion, Intel RealSense are some of the devices that provide depth information and support hand gesture recognition.

Much work is being already done using the depth information from Microsoft's Kinect and Leap Motion cameras related to sign language. Intel RealSense is the latest technology and not much work has been done with it regarding the sign language recognition. Intel RealSense Camera, a USB controlled input device was introduced by Intel, with the intention of allowing the user to interact with the systems more naturally and intuitively. This camera is mainly popular for the features such as face gesture recognition, 3D printing and scanning, video conferencing (to change the background) etc. The Intel RealSense camera is equipped with one HD camera, one Infrared camera and one infrared projector [1] .The Intel RealSense camera specifically is able to track the hand movements accurately in order to detect hand gestures that can be used for natural interaction. It is able to track 22 joints per single hand. Intel RealSense is capable of providing each minute detail of the 22 joint labels detected per hand such as finger

2

foldedness values, radius of the fingers, orientation of joints in both two dimensions and three dimensions, etc. The Intel RealSense SDK is able to provide information of a hand in different modes such as depth, color, cursor mode, etc. Intel RealSense SDK has a set of tracking features with robust API calls inbuilt, such as facial recognition, HandsViewer module etc.

Although the Intel RealSense API is capable of detecting gestures such as a V sign, hand wave, pinch and few other gestures, a direct call in the API is not available to detect more custom signs. This , thesis will focus on experimenting with the 0 -9 digits from Indian Sign Language (ISL) in order to evaluate the performance and accuracy of the Intel RealSense camera with respect to hand gesture recognition. The software provided by Intel RealSense is capable of extracting the accurate data points that will be needed for the study, and this software will help us skip the phases such as preprocessing and other computer vision techniques that are needed to remove background noise in the image etc.

The experiments in this project are based on the information that is collected from the Intel RealSense camera when the user is performing a particular sign from ISL to generate training data. Training data is generated by carefully analyzing which features to include and which features to exclude that does not make any difference. This data is split into both training and testing data. This data samples are tested with the Multi-class SVM in order to evaluate the accuracy of each sign from sign language. We will be focusing on how accurate the camera can be used to identify the gestures in sign language and try to analyze the limitations of the camera, analyze the future improvements that are possible in these areas etc. The rest of the paper is organized into

background, related work, implementation, validation, conclusion and future work.

CHAPTER 2. BACKGROUND

This chapter provides information on topics that may be useful for understanding further sections of this thesis document. In short this section will discuss Gesture recognition, Sign language recognition, Machine learning, Matlab.

2.1 Sign Language Recognition

Sign language involves non- vocal communication with a combination of hand movements, lip patterns and facial expressions in order to identify as a meaningful expression.

History: In early 1500s people with impairments were neglected and nobody respected them, because they were unable to communicate with the world. In the 16th century an Italian physician Geronimo Cardano, declared that it is necessary to take care of the deaf community and they should be taught how to communicate with the world. Juan Pablo de Bonet created and published the first book on Sign language in 1620[2]. In 1755, Abbé Charles-Michel delÉpée created the first sign language school with no cost to students in France. Later they created finger spelling and gestures to recognize phrases and words. Later on, different sign languages have been created wherever deaf communities existed all over the world. There are different sign languages for example American Sign Language, British sign language, Chinese sign language and others. The American Sign Language (ASL) [9] is complete and widely used. ASL consists of 26 signs that are used to express words from the English language. Indian Sign Language (ISL) has its own grammar and rules [1].

2.2 Gesture Recognition

In gesture recognition technology a camera reads the movements of the human body to communicate with the device that uses gestures as input to control the functions of that device.

The basic steps involved in hand gesture recognition as shown in Figure 2.1 are:

1. Image Acquisition: User input is captured as a single image or sequence of frames with single camera or sometimes two cameras.

2. Preprocessing: Preprocessing is the second phase of hand gesture recognition. In this process, the input image is divided into regions separated by boundaries by eliminating the noise.

3. Feature Extraction: Relevant data from the input are extracted as feature vectors and these features can be used for classification instead of using the entire input data.

4. Classification: It is the process of separating the features collected into predefined categories. Proper selection of the feature set and recognition algorithms can result in good performance of the classifier (accuracy in terms of identifying a gesture).

Input/Image Acquisition → PreProcessing → Feature Extraction → Classification

Figure 2.1 Basic Steps Involved in Gesture Recognition.

2.3 Machine Learning

Machine learning is defined as the methodology that can "detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty" [3]. It has been a significant subfield of Computer Science derived from pattern recognition and learning methodologies in Artificial Intelligence. In this research work, we have generated the input dataset from different users by extracting the joint information with the help of the Intel RealSense camera. Signs from the digits 0-9 of Indian Sign Language have been focused for this study. Only the significant features that we considered would be beneficial were included in the feature vectors by careful analyzations on the features that need to be excluded. Each time the user has to wait until the camera is calibrated and can completely detect every single joint of a user's hand, before he starts performing a particular gesture. The data is provided by the Intel RealSense SDK from the API function calls. In this process a lot of data is written to the files from every frame, and we manually selected one particular frame for each gesture per person. Each feature set has the different properties about 22 different joints such as finger openness value, radius of the finger , Image position values and world coordinate values in x, y, z axes. Detailed explanation of all the features that are considered for this study will be included in the Methodology section of this document. After this process, we tagged the feature set with the label so as to perform supervised learning to expect good accuracies. Once this tagging is finished we had a labeled dataset
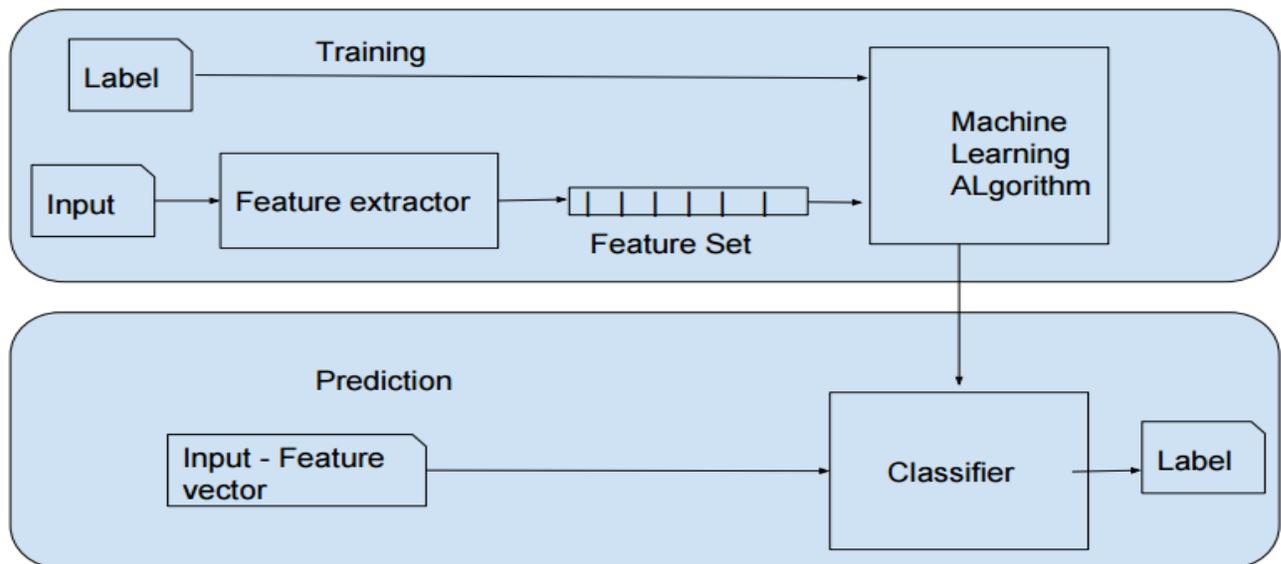
that is ready for the training phase.

Figure 2.2 Supervised Learning Model Workflow

A collection of feature sets should be extracted from the data in order to match the prediction process for the expected output. Selecting the dataset that is significant to the study out of huge amount of data is a critical task till date in Machine Learning. Once the feature set is selected the whole dataset is divided into different percentages of training and testing datasets. The training dataset is used to build the classification model to learn from the labeled data, and the classifier will be able to make predictions based on the given input samples. During the training phase, the classifier tries to learn from features as well as the labels and uses this learning in the testing phase to predict the samples based on the previous knowledge it has gained. These predictions are compared to the actual output to verify the accuracy.

2.4 Ten-fold Cross Validation

Cross validation is the process of partitioning the data set into necessary testing and training datasets. This is also known as rotational cross validation. In a k-fold cross

validation process, the data is divided into equal subsets of k size which are called folds [9]. One of these sets will be considered as the test set and the remaining will be the training set. Once the model is trained with the dataset and predictions are made, this test set will become part of the training set in the next iteration. This process is repeated until each fold has got a chance as a test set. For example in a ten-fold cross validation the first set will be the testing data and other 9 sets will be training sets. And in the next iteration the second set will be the testing set and the remaining will be considered for training. This procedure is repeated until every set has got its chance as the test set.
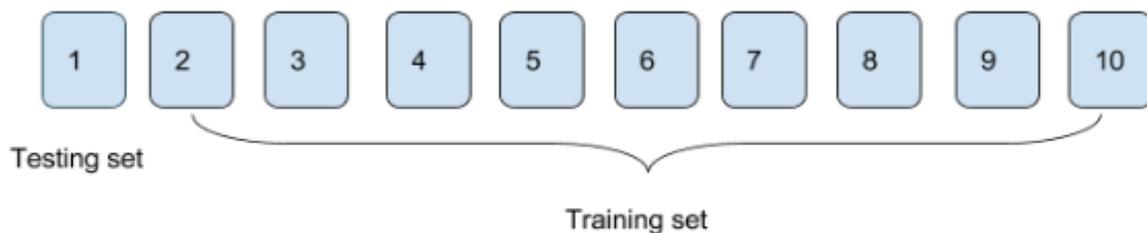
Figure 2.3 Ten- Fold Cross Validation

The main advantage with this process is that effective utilization of the dataset as both training and testing datasets and each dataset is used for validation once. This will help in building a more accurate prediction model. The more folds, the higher accuracy can be expected.

Ten-fold cross validation is the most used validation method by the researches, because of the proven results for having a good balance of producing good predictions given the reasonable size of data [6].

2.5 Classifiers

Classification is the process of categorizing samples into categories that they belong to. Classifiers are the algorithms that learn from the previous observations and use that knowledge to do further classification. Classification can be of two types: First, binary classification categorizes the samples into two classes, while multi-class classification categorizing the samples to multiple classes. Classifiers that we analyze for this thesis support Multi-class SVM, Decision trees and KNNs.

2.5.1. Multi-class SVM

Support vector machines were designed for binary classification but many problems deal with more than two classes. Multi-class SVM comes into the picture when there is a problem that needs more than "two class" classifications. Some of the problems where multi-class SVM is used are optical character recognition, speech recognition, bioinformatics and others [4]. Usually multi-class classification is achieved by binarizing one class versus all other classes.

Following are the commands to use SVM that are supported in Matlab for this project.

" G1vAll=(Train_label==u(k));

models(k) = svmtrain(Train,G1vAll);

svmclassify(models(k),Test); "

2.5.2 Decision Tree

A decision tree classifies the data based on the conditions that it forms from the root to the leaf nodes. It predicts the target values based on the tree that it forms during the training process. It can classify data into true or false, as well as multiple target

values. Decision nodes are the nodes in the tree from which the nodes are split based on a specific condition. One advantage of using decision trees is that they are non parametric, which means they have no assumptions about the distribution and the classifier structure.

2.6 Performance Measures

To validate the model, after the classification is done, the accuracy has to be calculated.

Some of the commonly used metrics in machine learning are accuracy, recall, precision, F-score. These values are calculated by considering different combinations of True Negatives, True Positives, False Negatives and False Positives. Positives are the observations that are selected by the classifier and negative are the observations that are not selected by the classifier.

Accuracy of a classifier is calculated by number of correct observations divided by total number of observations.

Accuracy=   (True Positives+True Negatives)_____

                (True Positives +False Positives+False Positives+False Negatives).

Precision is calculated by the total number of positives divided by the number of classifications that belong to the positive class [7].

Precision =   True Positives_____

                (True Positives +False Positives).

Recall is calculated by the total number of True Negatives divided by the total number of classifications that actually belong to the positive class [8].

Recall=   True Positives

               (True Positive+ False Negatives)

The F-Score is the harmonic mean of Recall and Precision and is considered as balance for both Recall and Precision.

F-Score=   2PR

               (P+R)

Where P stands for Precision and R stands for Recall.

Balanced F-Score: when either of Recall or Precision is very high, then a weight can be used to balance the F-Score result. This is called Balanced F-Score.

2.7 Matlab Machine Learning Toolkit

Matlab comes with the statistics and Machine Learning toolkit, which supports different learning models. The Matlab Coder can even generate C-Code for any given classification model.

Matlab has a feature called Classification Learner App which will easily allow the user to select from a variety of the classification models and experiment with the data. Some of the models that are included in this app are decision trees, regression, support vector machines and others. Matlab provides a nice UI for the user to select and experiment with the subsets of feature sets and visualize the results. It automatically trains the model given the data and can help users with best model to select. By default this classification learner app takes care of overfitting and also gives another option of holdout validation [5]. It also allows the users to use cross-validation with different folds to validate the model.

CHAPTER 3. RELATED WORK

Although there exists previous work in different sign languages in different countries with different devices such as Kinect and Leap Motion, not much work has been done using the Intel RealSense camera. But the main motivation for this paper comes from the following academic papers with the authors researching different experiments with devices in order to perform gesture recognition. Each of these systems are developed by implementing different algorithms, features and methodologies etc. This section discusses the implementation, results obtained and future scope of the works done by the researchers.

3.1. Real Time Letter Recognition for ASL: (Arabic Sign Language) using Leap Motion Controller and Kinect:

The authors in [11] developed a supervised learning model using two cameras, Leap Motion controller and Kinect. They tried to build the model by considering the two depth images from both cameras and extracted 3D world coordinate orientations of each joint in the hand. With this model the authors claim that users will be able to perform actions in front of the camera, and the corresponding letters in ASL are then displayed to the user.

The Kinect and Leap Motion controller were placed at distance of 50 centimeters apart from each other. The main features they considered for recognition are bone directions along with the features such as the fingertip, bone center point, bone from point, and bone to point.

TABLE I. SAMPLE OF "ᴗ" AND "ε" LETTERS SIGNED BY THREE PARTICIPANTS

Figure 3.1 Screenshot of Leap Motion Tracking ASL signs

The authors claim that, with their model they were able to detect 26 out of 28 signs with 100 % accuracy. One of the main advantages of this model is that it makes use of both cameras, the Kinect is used for tracking the joints and the Leap Motion is used to track more detailed information such as finger details. With the use of the Leap Motion they tried to overcome the limitation that they had with the Kinect in terms of finger detail information. The main drawback is the fact that the total number of samples during the training is very less.

3.2. Finger Spelling Recognition using Leap Motion Controller

Second is the research done on detecting and analyzing the finger spellings using the Leap Motion controller. In this paper, the authors proposed a finger spellings recognition model that can detect 16 finger spellings using their own sign language recognition model. The rate of recognition for finger spelling can be varied if the order of the representation in a decision tree is changed. To obtain the optimal solution they used generic algorithms and performed several experiments such as finding crossover probabilities, and finding appropriate mutation probabilities [12]. They obtained an

14

accuracy of 82.71% recognition rate. The main drawback of this model is that the decision tree is fixed and if the hand movements for the other two finger spellings are also included, the recognition rate at different conditional branches will change and affect the accuracy of the model.

3.3 Sign Language Recognition using Leap Motion and Kinect

In this study the authors proposed a model for manual American Sign Language alphabets by extracting the features based on finger positions and orientations of the fingers [13].In this study the authors compared the depth information from both Kinect and Leap Motion controllers and detailed how the combined data can be used to improve the recognition rate further. Features that they extracted from the Leap Motion include position of finger tips, palm center, hand orientation etc. Features extracted from the Kinect sensor are from both depth and color from the 3D points of the hand. Their database consisted of 10 different gestures performed by 14 users 10 times each for a total of 1400 samples. With the Leap Motion features they prove that, the three features when combined together have improved the accuracy to 81 %, than using them individually. Kinect provides more information but it is less accurate when compared to Leap Motion.

CHAPTER 4. METHODOLOGY

This chapter will explain in detail the methodology of how the features were extracted and the model was trained and evaluated. We have experimented the model with different supervised learning models such as Decision trees and KNNs.

System Setup: Hardware and Software Requirements:

The Intel RealSense SDK has particular hardware and software requirements to support proper functioning. It requires a USB 3 port to connect the Intel RealSense Camera, 8GB free disk space, Windows 8 or 10, and a 64 bit operating systems. We used Microsoft Visual Studio 2015 for this project.

4.1 Gesture Selection

The gestures that we selected for this study are the digits 0 to 9 from the Indian Sign Language. Every sign from 0 to 9 was declared clearly in Indian Sign Language grammar, so that there is not much scope for the camera to get confused due to ambiguity. All the signs were performed in front of the camera by the user's hand facing towards the camera. The methods that are used for this process are basically provided by the Intel RealSense API. We used the Hands Viewer API part for our experiment where there are methods that make the processing much easier. The Hand module consists of different modes like cursor mode extremities mode and full hand mode. For this study we extracted only the information from 1080p HD sensor camera of the device using full hand mode tracking. But the Intel RealSense camera also has IR sensors to track depth information

Cursor mode: The cursor mode is very accurate when compared with the other two but is limited to support only few gestures.

Extremity mode: This will track and shows the entire hand boundary along with the fingers but does not provide details about the joints on the fingers. It provides the extremities of the hand which include left most, right most, bottom most, top most, center point on the hand.

Full hand mode: It provides the 3 dimensional skeleton of the hand including 22 joints information and supports many gestures when compared with other modes.

This Hand module API tries to focus only on the hand part and normalizes all sizes of the hand parts, making the processing easier for further feature extraction.

Every user is asked to perform the signs from 0 to 9 in order to extract joint information about 22 joints that are present in a single hand. Every user will have to open their hands full in order to make sure that the camera is calibrated and then they can perform each sign individually. The distance of the user's hand from the camera is very important because the camera tries to calculate all the 22 joints, very close or very far can make the user's hand go out of focus and may result in wrong data.

The Intel RealSense camera is capable of detecting one or two hands, and the users were able to perform the gestures with either their left hand or their right hand. It points to the center of the joint and draws labeled different joint lines for every finger, and it is not difficult for the camera to give accurate values.

For querying the hand data, first we need to initialize and configure the Hand module which is done by initializing the Sense Manager. The Sense Manager is a simple interface that supports hand tracking, face detection, voice recognition and others. We can configure the Hand module to extract different hand configuration information. For example, hand segmentation images, joint tracking information, enable tracked joints, and

enable normalized joints etc.

But since our research is based on joint information we have configured the hand module to extract joint information, which uses the function enableTrackedJoints. This can provide the rotation and position data of all 22 joints that are detected per hand. We have used the following functions in order to extract the joint position and orientation of each joint in the x, y and z axes.

1. IsCalibrated: This will return true if the camera is able to detect all the joints in the hand for the user. Once the calibration is completed then an alert is issued. A valid calibration is essential for accurate information.

2. HasTrackedJoints: this function returns true if the tracked joint data exists otherwise it returns false.

3. QuerytrackedJoints: This function will help to retrieve the tracking information from the joint. This function uses joint label and joint data as parameters. Joint label is the enumerator that has variables such as JOINT_WRIST, JOINT_CENTER, and so on to the 22 joint labels per hand as shown in the Figure 1.
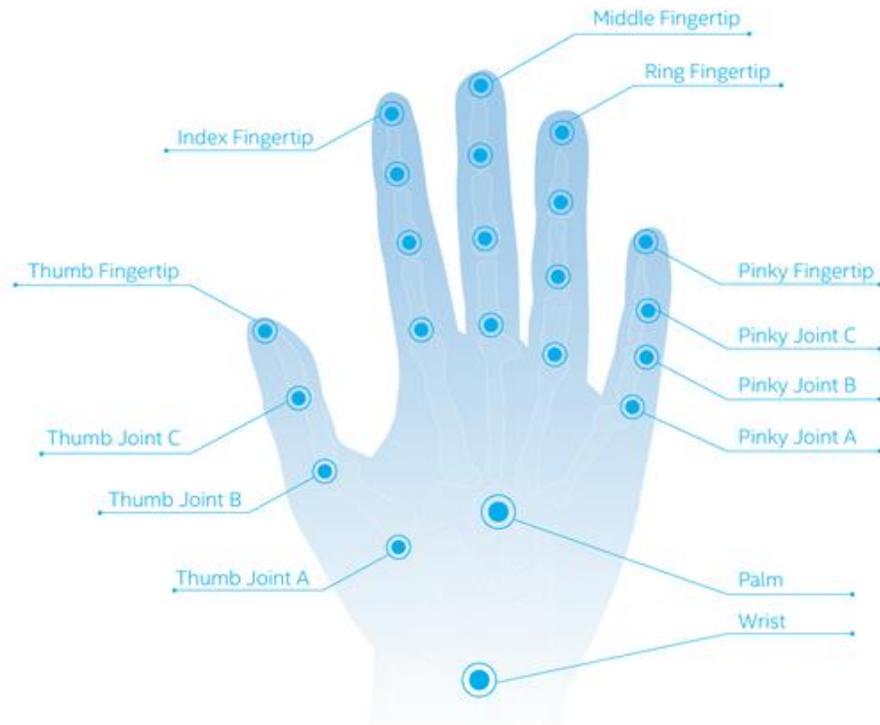
Figure 4.1 Joint Labels

And joint data structure will contain the parameters such as confidence, Position World, Position Image, local rotation, global orientation and speed.

4. HasNoramlizedJoints: This function returns true if the normalized joint data is present otherwise false.

5. QueryNormalizedJoints: It returns the data of the normalized joint form the tracked skeleton.

4.2 Feature Selection

For this project we have considered different features that were provided by the Intel RealSense SDK, Hands Viewer module.

Following are the collection of feature vectors that we considered for our feature

vectors.

1. Openness Property: This will range from 0 (all fingers completely closed) to 100(All fingers wide spread).

2. Finger Data: Finger data can provide two properties: foldedness value and radius of the finger [14].

   a. Foldedness: This will range from 0(the finger is completely closed towards the center of the palm) to 100(the finger is open).

Figure 4.2 Finger Foldedness

   b. Radius: This will return the radius of five finger tips of a user measured in pixels. It is an approximation of the fingertip mass.

Figure 4.3 Fingertip Radius

3. Position World: Geometric position of a joint in world coordinates in meters

4. Position Image: Geometric position of a joint in depth coordinates.

We collected 143 features per sign which are included in the feature vector.

4.3 Data Collection

A total of 40 users, both CalPoly students and outsiders, were asked to perform

the user experiments with the camera and the information was collected for all the signs from 0 to 9. The user was able to see how the hand is being tracked with the help of the visualizer provided by the Intel RealSense SDK. This helped the users to see how their hand was tracked in real time and how the camera is able to detect the joints and details from their hands. A total of 40x10 samples were collected from the participants. The users were encouraged to perform the gestures at different angles and with both left and right hands.
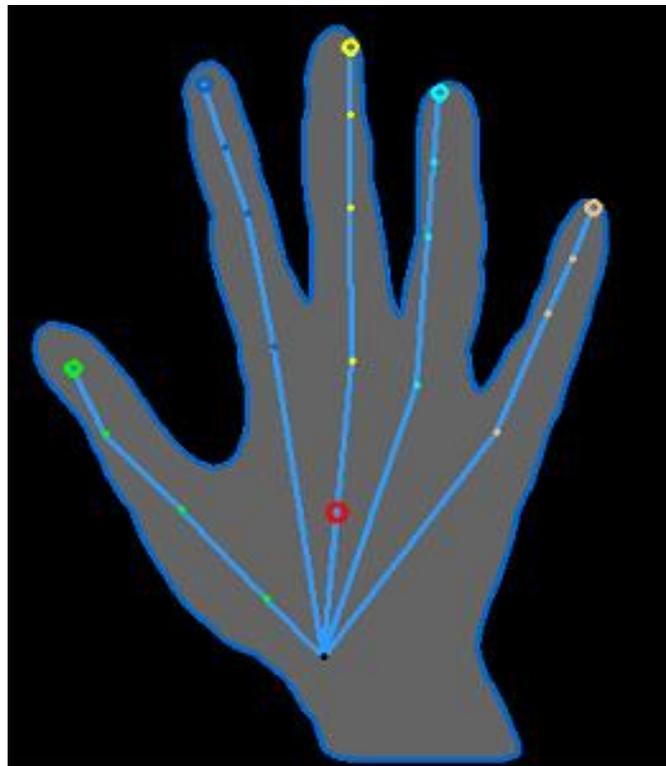

Figure 4.4 User hand after Calibration

Due to the hardware limitations of the camera a few gestures were tracked completely wrong. For example the camera had difficulty with the sign 9 when compared with any other sign. One reason for that is due to the portion of the hand that is not visible

to the camera. Because of this, the camera assumes and tries to approximate the invisible fingers and results in incorrect information about the joints.

4.4 Labeling

For this study, the users were allowed to perform in front of the camera to extract the features from frames. All the serialized and normalized arrays of values from the frames are written to a file. And one set of values is picked manually by observing the set carefully. I.e. selecting the set of features that were recorded after the calibration process is finished. One limitation of the camera that caused difficulty to the users is that, once the user's hand is very close or far from the camera, it goes out of the boundary and the user cannot return to the gesture directly. Instead the user will have to do the calibration process again and then perform the gesture to obtain accurate readings.

We created two separate data files: one consisted of an array of feature vectors, and the other file consisted of the corresponding class names of the feature vector. Since the data has a large of number of frames we had to select the feature vectors for each symbol for every user and do the labeling.

4.5 Extraction

Selecting the features was a difficult and important task in this project. It was difficult for us to decide on the features because of the fact that the Intel RealSense API can provide a lot of information about a hand. So we analyzed what are the feature points that may differ for the 0 to 9 signs that can significantly help the classification process. We have tried to experiment with two methods. One way was to collect a dataset by capturing the images that contain depth and RGB information, but this method was not

feasible for our study because of the limitation to obtain a huge training dataset of images for deep learning. Previous research conducted by other researchers had shown that deep learning will require a huge dataset and cannot guarantee expected results. So, we decided to proceed with creating features sets with the numerical joint values. We decided to consider radius of finger, finger foldedness value, and global and local positions of the joints in the x, y, and z dimensions. We did some data analytics and removed the values from the feature vectors that did not contribute to the classification process. We have extracted 121 features per sign per user in the feature vector that we used for training.

4.6 Training

For training purposes we chose supervised learning model multi-class SVM. A total of 40x10 samples were manually labeled for training the classification model. We considered different splits of the training data to compare the performance with respect to training size. We have written code in Matlab which will help us change the training and testing samples in different proportions and it will also allow us to choose different subsets of feature vectors.

Matlab has a classification learner app as the feature in the tool that will allow us to create a training model easily with different algorithms. With this inbuilt feature we will be able to consider different machine learning algorithms and experiment to see the variation in the results. But for this project, we had considered multi-class SVM ,decision trees, and KNN learning models.

4.7 Evaluation

Our algorithm in Matlab is written in such a way that the dataset is divided in the different portions for both training and testing samples. But for testing purposes we provided the model with unlabeled data to measure the performance of the model. The predictions that are made by the classifier are compared to the actual label for that particular gesture and verified. All the performance measures are recorded to compare the results.

In addition to calculating the performance based on accuracy we measured the accuracy in terms of training time and testing time for the two classifiers and compared the results. We calculated accuracy based on the number of True Positives, True Negatives, False Positive and False Negatives. The training time will be much higher if we consider a real time incremental model which would add all the frames to the training set that are being recorded, which will need much more processing power and requires extra disk space.

CHAPTER 5. RESULTS

This study generated a significant amount of analyzations to understand the better performance of the model as well as the camera. We tried to make different splits of the training samples in order to analyze which percentage of training split showed better performance. We experimented with different features to analyze which features significantly increased the performance as well as to identify which features might not be that suitable to consider for this recognition model. We generated confusion matrix for the decision tree model with 10 fold validation in order to analyze which class had more difficulty in classification.

Through this study we found that features such as hand openness values, radius of fingertip, foldedness values of the fingertips showed higher accuracy when compared to the joint orientation information. But overall the model showed good performance in terms of accuracy when all the 121 features were considered. The following tables represent different splits of training sizes with different features sets.

Multi-class SVM:

Table 5.1 Performance when 1-11 features are considered (openness value, radius of five finger tips and foldedness values of five finger).

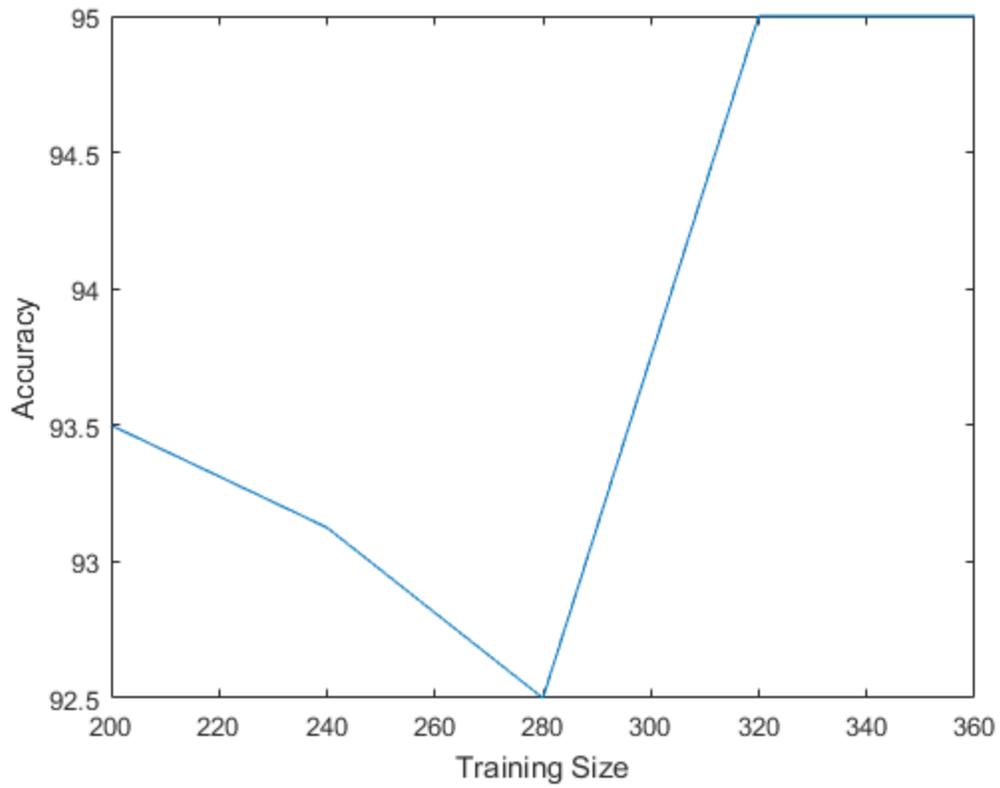| Training Size	Accuracy
200	99
240	99.37
280	100
320	100

360	100
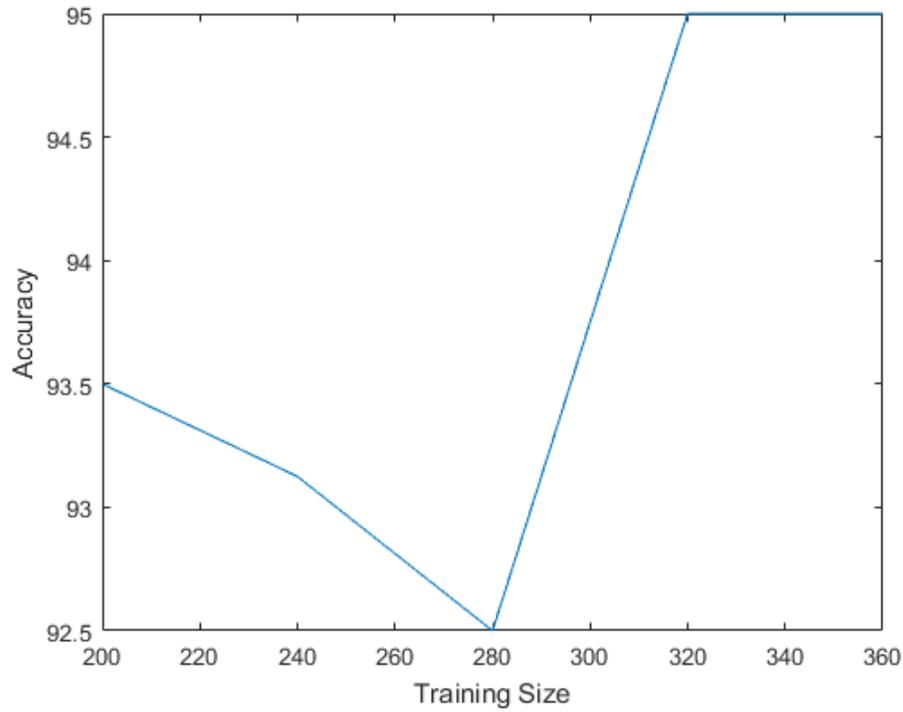
Graph 1 Accuracy vs Training size for features such as radius, openness, finger

foldedness values

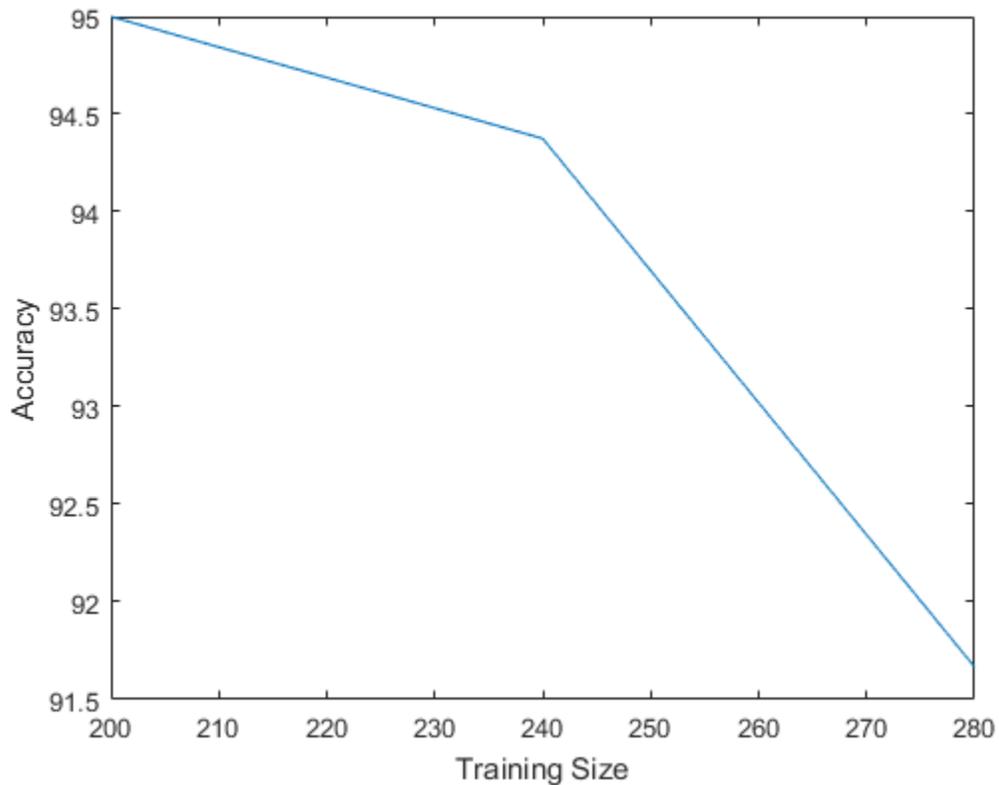Table 5.2 Performance when 121 features are considered

Training size   Accuracy

200	93.5
240	93.1
280	92.5
320	95.0

Graph 2 Accuracy vs Training size when whole set of features are considered.

Table 5.3 Performance when only position values of joints are considered.

Training Size	Accuracy
200	95
240	94.3
280	91.6

Graph 3 Plot that depicts Accuracy Vs Training size when position values of joints are considered.

We have found an important observation while experimenting with just the position values of the joints, which says "the training data reached maximum number of iterations and did not reach convergence". This means that these features taken alone cannot contribute for further classification, and that these particular features can only be considered into entire feature sets for classification.

We have found that, features such as hand openness value, radius of fingertips, foldedness values of fingers have shown higher accuracy when compared to the results with position values of the joints. But overall, we found that when all the features considered the model showed an average accuracy of 93.82.

We used classification learner app available in Matlab in order to build different models such as decision trees and KNNs and calculated the accuracies. Following table shows the results for KNNs and Decision tress with different validation folds.

Table 5.4 Results from classification learner app

Number of folds in validation

Classifier	5	20	30	50	holdout
Tree Complex	84.3%	84.8%	84.8%	84.5%	83%
Tree Medium	84.3%	86.3%	85.3%	86%	83%
KNN Weighted	51.5%	54.3%	55.1%	54.5%	56%
KNN Medium	45.5%	49.3%	48.8%	48.0%	49%
KNN Fine	51.7%	52.8%	53.5%	52.5%	48%

As we can see from the above table, decision trees had shown an average accuracy of 84.89% accuracy when compared to KNNs with different validation splits.

Table 5.5 Confusion matrix for decision tree model
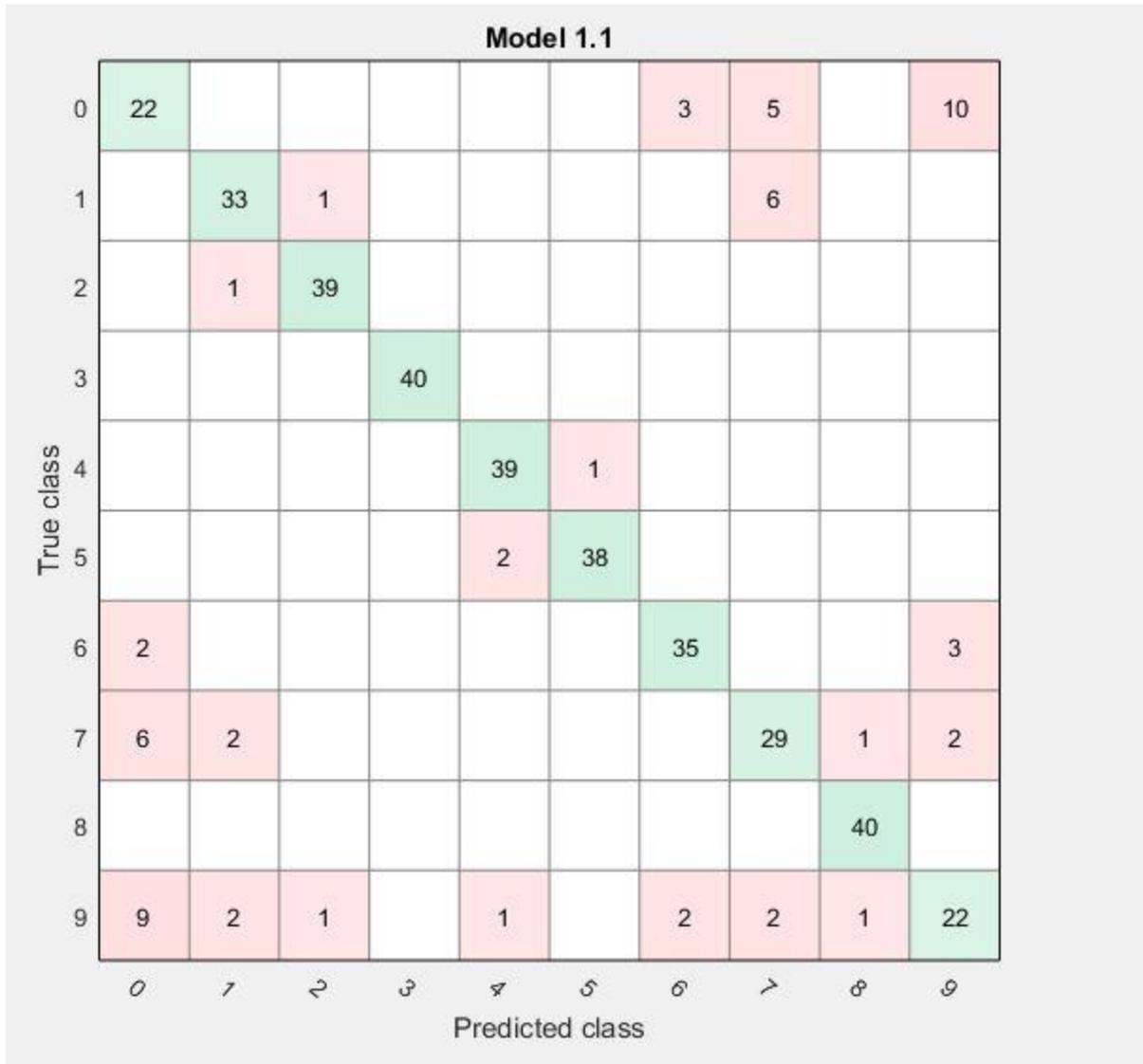
We plotted the confusion matrix for the decision tree with 10 fold cross validation model. From the above figure, we can analyze that Class '0' and Class '9' had the least number of true calculations when compared with the other classes. One reason for this

is that the features for these symbols are similar. Classes' 3' and '8' have distinct features with 100% accuracy.

CHAPTER 6. FUTURE WORK

The main aim of this study was to determine if the Intel RealSense camera is suitable to use for recognizing sign language gestures. In this process we tried to extract information provided by the Intel RealSense camera and tried to manually select a set of feature vectors from the joint information. This information was used to build training models using multi-class SVM, decision trees and KNNS. The entire training and testing process was done offline which means we extracted the data from the user first and then tried to build the training model in Matlab. One area for future work, is that we can train a model in real time while the user is performing experiments in front of the camera. But the limitations with this process is that, it needs lot of disk space (as in real time it writes a lot of frames) and high processing power (in order to maintain high performance).

Another scope of improvement with this trained model is, it can be used to build applications that will help users to interact and perform some action based on the recognized gesture. This can be expanded to recognize alphabets and other simple grammar in sign language. This kind of model can be used to build an application that can predict and autofill the words that the user is trying to communicate based on the previously interpreted signs by the model.

In this study we tried to manually select only one feature set per sign from each user, but it can be made more efficient by making this feature selection process automatic. Previously we have researched different ways and methods that showed proven results from different researchers but we have decided to go forward with the joint information. Deep learning usually has shown good performance in terms of identifying even complex features that are difficult for the humans to identify. This method will be successful if we

have a very huge amount of training data. Due to time limitations and the difficulty of obtaining huge training set, we have decided not to move forward with deep learning.

One area of research can be to expand the sign language grammar with the help of Intel RealSense camera as well as using other sensors such as Kinect or Leap Motion. This can help in having more accurate model, and can be further expanded to different modules that can identify different sign languages from all around the world.

CHAPTER 7. CONCLUSION

This study sought to examine and analyze if the Intel RealSense camera is able to identify gestures accurately and efficiently. We developed a simple system to store the data that we extracted from different participants and used that to train and test our machine learning model. We experimented with Multi-class SVM , Decision trees and KNNs to compare which classifier is best suitable.

Through our study we were able to analyze that since we considered only numbers recognition, the classifier did not have much problems during classification because there were no two signs that have similar representations. But if the model is expanded to identify the alphabets also, then there might be a chance of confusion between the signs that have same representations. The finger spellings of alphabets from ISL are included in Appendix C.

Some of the examples that fall into similar representation category of signs are:

1. number '0' and alphabet 'O'.

2. number '2' and alphabet 'V'

Previously shown studies have experimented with devices such as Kinect, Leap Motion, Data glove etc. But, Intel RealSense is latest technology that supports gesture recognition. So, we wanted to evaluate the device to see if it can be used for sign language recognition. Through this study we found that our model showed a good average accuracy of 93.5% and the camera was able to detect important information. Our study was limited to just identifying Indian Sign Language(ISL) numbers from 0-9.This study will hopefully serve as a foundation for expanding the improvements in sign language recognition using this device. Through this study we found out that multi-class

SVM classifier has shown better performance when compared with other models.

BIBLIOGRAPHY

[1] http://www.intel.in/content/www/in/en/architecture-and-technology/realsense-

  overview.html

[2] Chen, Justin K., et al. "Sign language gesture recognition with unsupervised feature

  learning." *(IJCSE) International Journal on Computer Science and*

  Engineering 2.3 (2010): 560-565.

[3] Murphy, Kevin P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[4] Wang, Zhe, and Xiangyang Xue. "Multi-class support vector machine." *Support*

  Vector Machines Applications. Springer International Publishing, 2014. 23-48.

[5] https://www.mathworks.com/solutions/machine-learning.html

[6] McLachlan, Geoffrey, Kim-Anh Do, and Christophe Ambroise. *Analyzing microarray*

  gene expression data. Vol. 422. John Wiley & Sons, 2005.

[7] Karita, S., Nakamura, K., Kono, K., Ito, Y., & Babaguchi, N. (2015, June). Owner

  authentication for mobile devices using motion gestures based on multi-owner

  template update. In Multimedia & Expo Workshops (ICMEW), 2015 IEEE

  International Conference on (pp. 1-6). IEEE.

[8] Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze.

Introduction to information retrieval. Vol. 1. No. 1. Cambridge: Cambridge university

  press, 2008.

[9] https://www.openml.org/a/estimation-procedures/1

[10] https://www.mathworks.com/solutions/machine-learning.html

[11] Almasre, Miada A., and Hana Al-Nuaim. "A Real-Time Letter Recognition Model for Arabic Sign Language Using Kinect and Leap Motion Controller v2." *IJAEMS Open Access Int. J. Infogain Publ.* 2.5 (2016).

[12] Funasaka, Makiko, et al. "Sign Language Recognition using Leap Motion Controller." *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015.

[13] Marin, Giulio, Fabio Dominio, and Pietro Zanuttigh. "Hand gesture recognition with leap motion and kinect devices." *Image Processing (ICIP), 2014 IEEE International Conference on*. IEEE, 2014.

[14] https://software.intel.com/sites/landingpage/realsense/camera-sdk/v1.1/documentation/html/index.html?fingerdata_pxchanddata.html

[15]. https://www.mathworks.com/help/stats/kmeans.html#zmw57dd0e439495

[16] Dong, Cao, Ming Leu, and Zhaozheng Yin. "American Sign Language Alphabet Recognition Using Microsoft Kinect." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2015.

[17] Shangeetha, R. K., V. Valliammai, and S. Padmavathi. "Computer vision based approach for Indian Sign Language character recognition." *Machine Vision and Image Processing (MVIP), 2012 International Conference on*. IEEE, 2012.

[18] Chen, Justin K., Debabrata Sengupta, and Rukmani Ravi Sundaram. "CS229 Project Final Report Sign Language Gesture Recognition with Unsupervised Feature Learning."

[19] Wang, Guan-Wei, Chunxia Zhang, and Jian Zhuang. "An application of classifier combination methods in hand gesture recognition." *Mathematical Problems in Engineering* 2012 (2012).

[20] Ibraheem, Noor Adnan, and Rafiqul Zaman Khan. "Survey on various gesture recognition technologies and techniques." *International journal of computer applications* 50.7 (2012).

[21] Osman, Ghazali, Muhammad Suzuri Hitam, and Mohd Nasir Ismail. "Enhanced skin colour classifier using RGB ratio model." *arXiv preprint arXiv:1212.2692* (2012).

[22] Ravikiran, J., et al. "Finger detection for sign language recognition."*Proceedings of the International MultiConference of Engineers and Computer Scientists*. Vol. 1. 2009.

[23] Rumyantsev, Oleg, Matt Merati, and Vasant Ramachandran. "Hand sign recognition through palm gesture and movement." *Image Processing* (2012).

[24] Funasaka, Makiko, et al. "Sign Language Recognition using Leap Motion Controller." *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015.

[25] Kumar, Piyush, Jyoti Verma, and Shitala Prasad. "Hand data glove: a wearable real-time device for human-computer interaction." *International Journal of Advanced Science and Technology* 43 (2012).

[26] Ghotkar, Archana S., and Gajanan K. Kharate. "Study of vision based hand gesture recognition using Indian sign language." *Computer* 55 (2014): 56.

[27] Chourasia, Neha S., and Sampa Barman. "Hand Gesture Spotting Using Sign Language through Computer Interfacing."

[28] Sigalas, Markos, Haris Baltzakis, and Panos Trahanias. "Gesture recognition based on arm tracking for human-robot interaction." *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010.

[29] Chen, Chia-Ping, et al. "Real-time hand tracking on depth images." *Visual Communications and Image Processing (VCIP), 2011 IEEE*. IEEE, 2011.

[30] http://www.ijarcce.com/upload/2015/may-15/IJARCCE%2093.pdf

Appendix A

This section will provide the details of the representations of the signs, the code that we used in Matlab and the sample training file. These images were obtained from Intel RealSense SDK interface while the users were performing gestures.
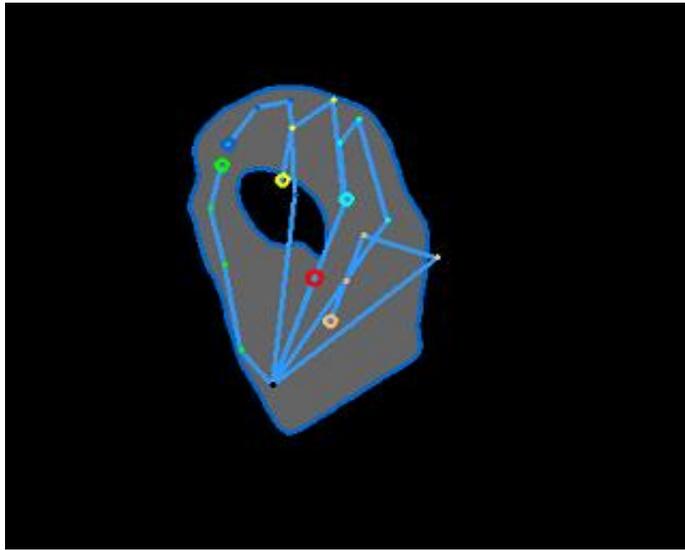
Figure A.1 Sign 0 in ISL
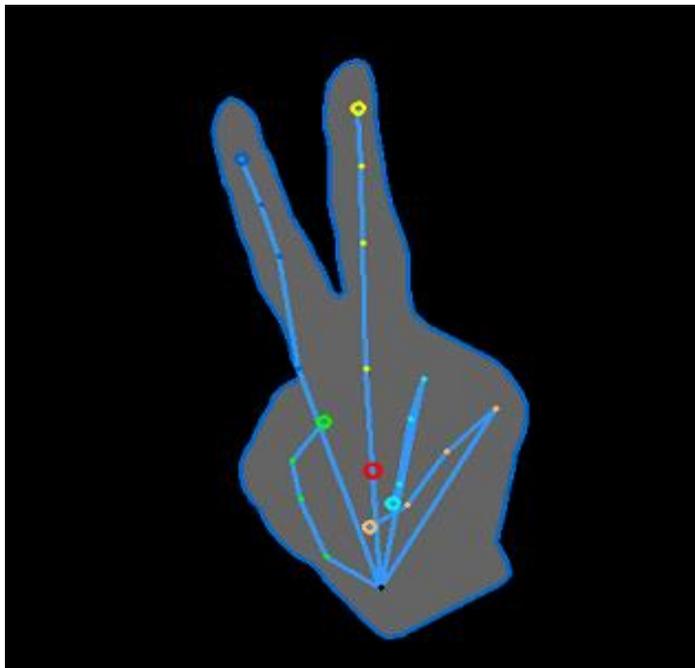

Figure A.2 Sign 1 in ISL
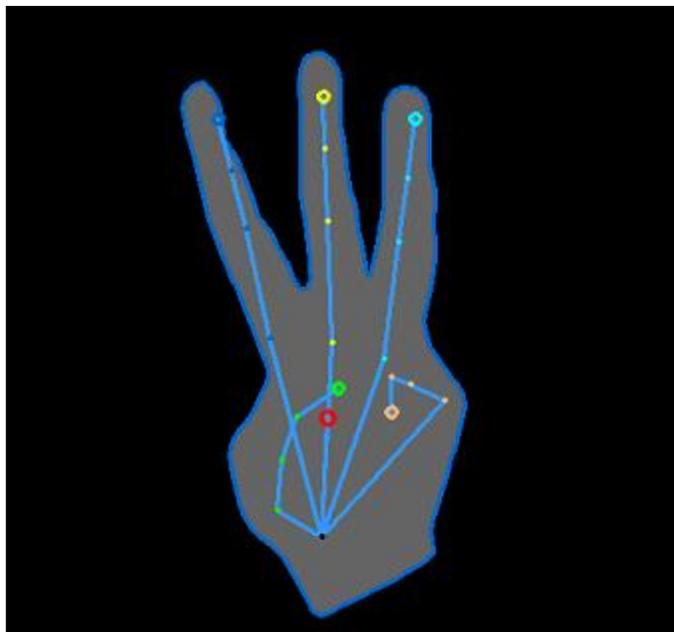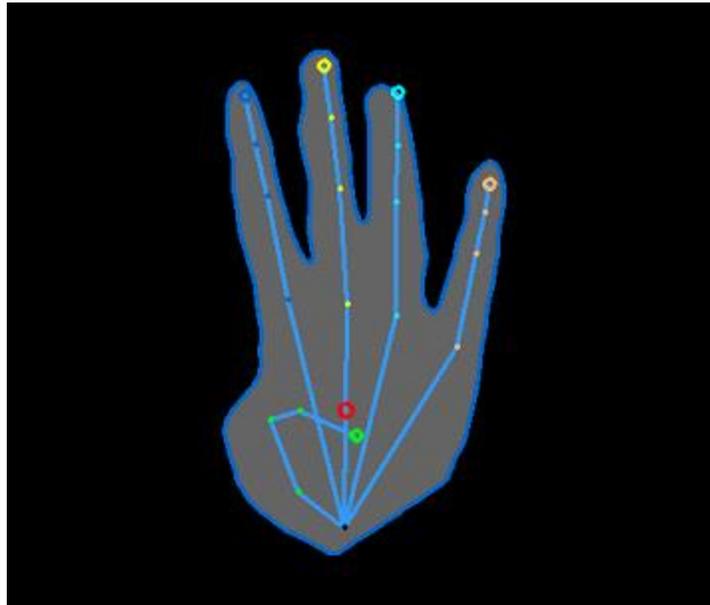
Figure A.3 Sign 2 in ISL
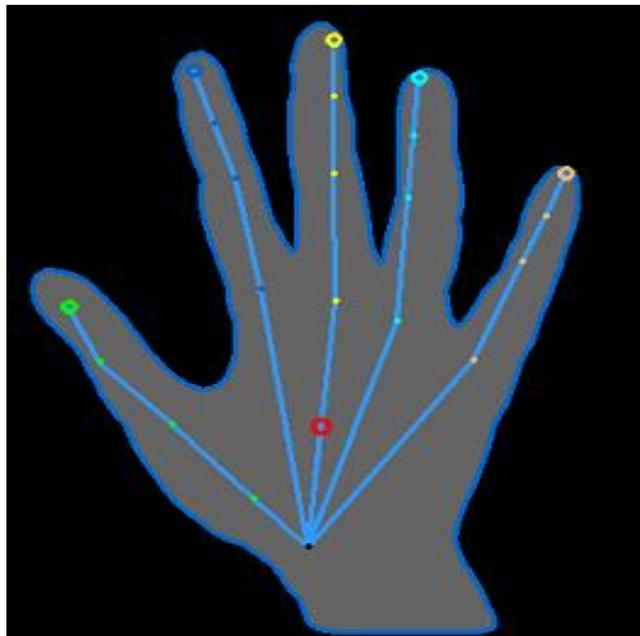

Figure A.4 Sign 3 in ISL
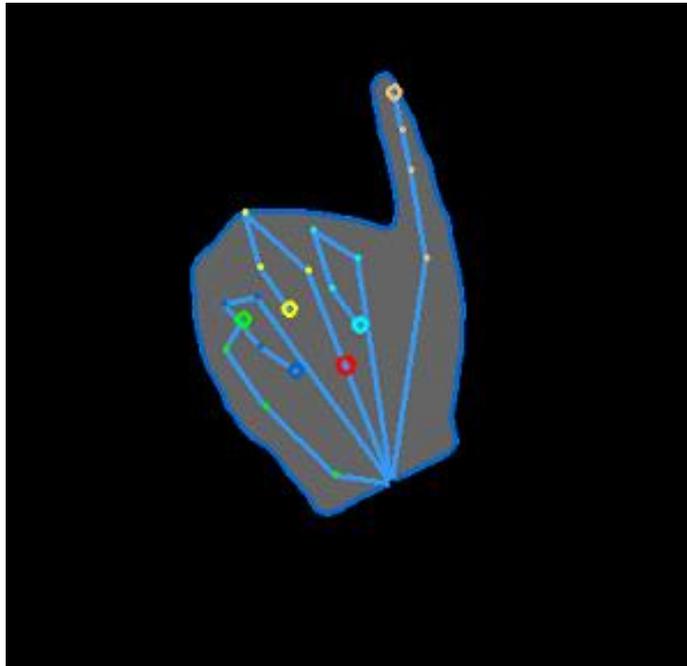
Figure A.5 Sign 4 in ISL
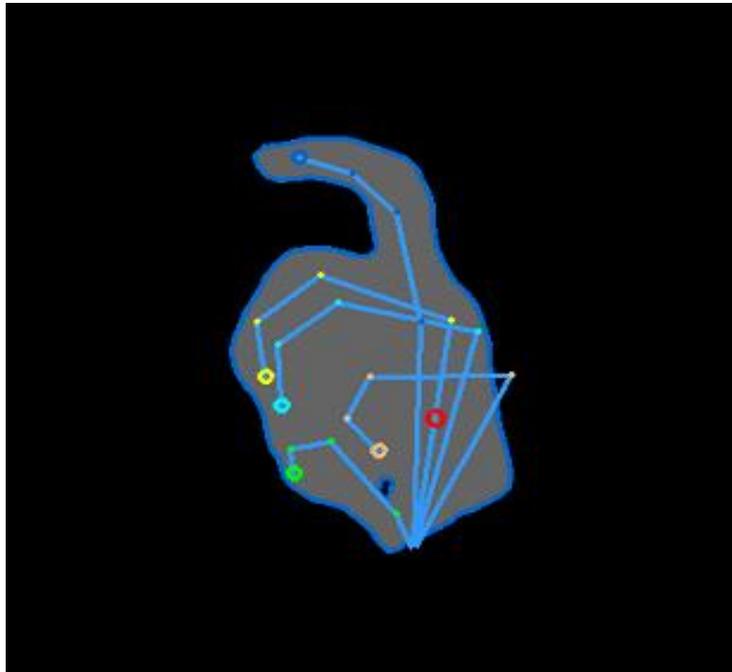

Figure A.6 Sign 5 in ISL
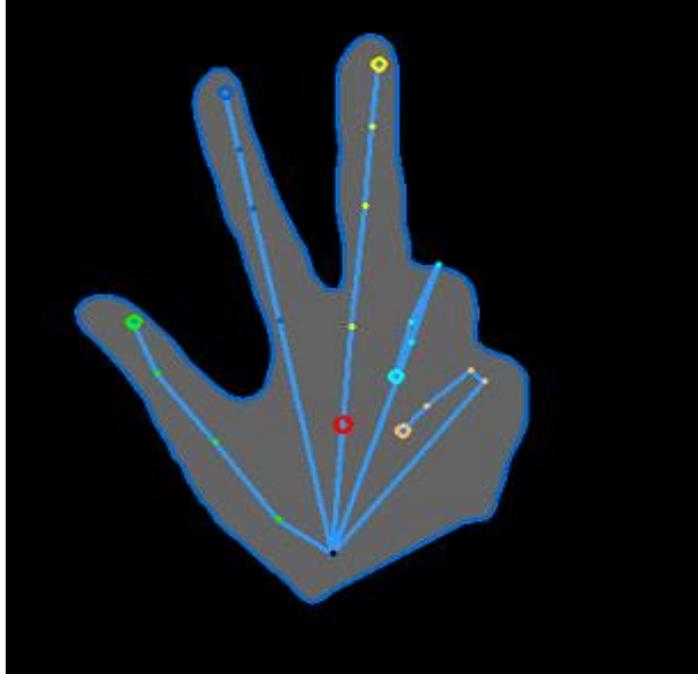
Figure A.7 Sign 6 in ISL
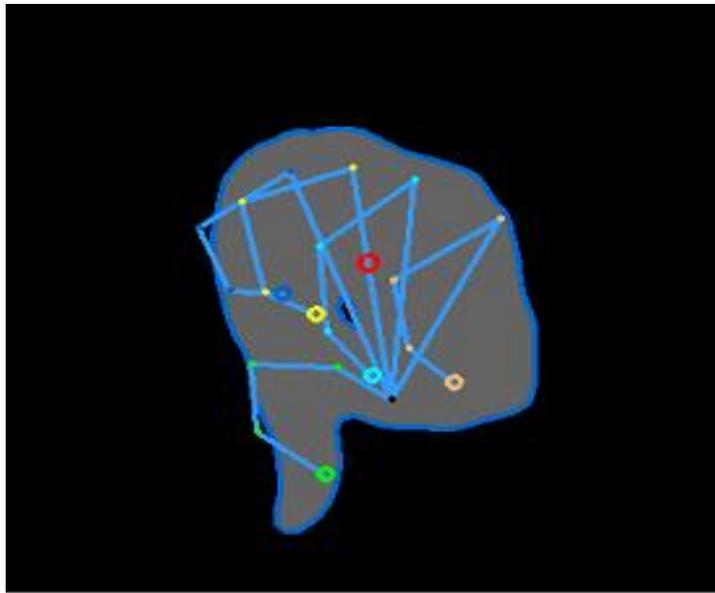

Figure A.8 Sign 7 in ISL

Figure A.9 Sign 8 in ISL

Figure A.10 Sign 9 in ISL

Figure A.11 Decision tree model for 1 to 11 features from the feature vector with 10 fold cross validation.

Op: Openness value
F1_R: Radius of thumb finger
F1_F: Foldedness value of thumb finger
F2_R: Radius of index finger
F2_F: foldedness value of index finger
F3_R: Radius of Middle finger
F3_F: foldedness value of Middle finger
F4_R: Radius of Ring finger
F4_F: foldedness value of Ring finger
F5_R: Radius of Pinky finger
F5_F: foldedness value of Pinky finger

Appendix B

Sample Training Data set of a User

0 - 9, 0.00841647, 28, 0.00841647, 17, 0.00841647, 7, 0.00841647, 1,   0.00841647, 0

1 - 64, 0.00845668, 25, 0.00845668, 100, 0.00845668, 75, 0.00845668, 8, 0.00845668, 36

2- 49, 0.0086112, 0, 0.0086112, 100, 0.0086112, 100, 0.0086112, 0, 0.0086112, 0

3- 90, 0.00847481, 0, 0.00847481, 100, 0.00847481, 100, 0.00847481, 94, 0.00847481, 23

4- 100, 0.00825778, 0, 0.00825778, 100, 0.00825778, 100, 0.00825778, 100, 0.00825778, 100

5- 100, 0.00826447, 100, 0.00826447, 100, 0.00826447, 100, 0.00826447, 100, 0.00826447, 95
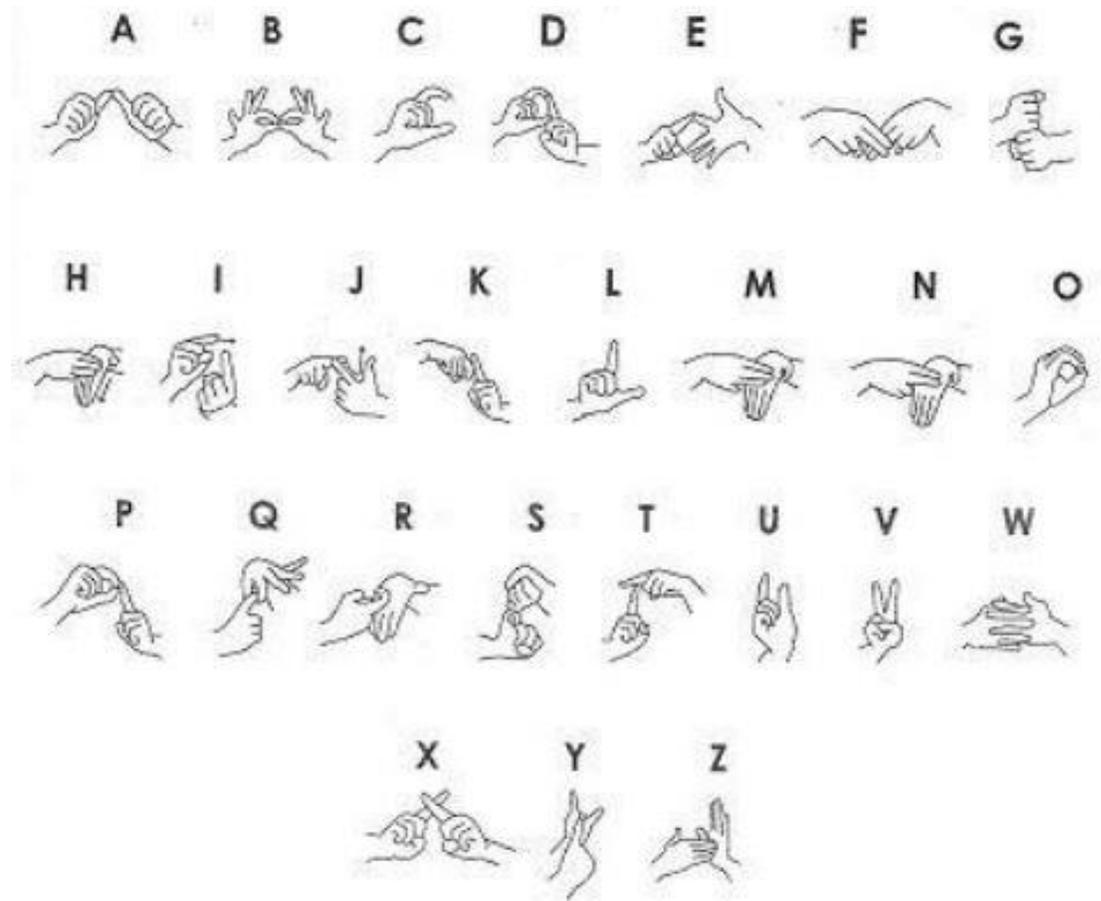
6- 14, 0.00836978, 90, 0.00836978, 0, 0.00836978, 5, 0.00836978, 0, 0.00836978, 100

7- 20, 0.00844361, 0, 0.00844361, 59, 0.00844361, 0, 0.00844361, 0, 0.00844361, 10,

8- 51, 0.00853083, 100, 0.00853083, 100, 0.00853083, 95, 0.00853083, 0, 0.00853083, 0

9- 0, 0.00854994, 0, 0.00854994, 0, 0.00854994, 0, 0.00854994, 0, 0.00854994, 17

Appendix C

A.12 ISL Alphabets [30].