Abstract

We propose a system that would translate Pakistan Sign Language to text in real time. As a prototype, the system is limited to recognition of four pre-defined sentences. The system takes input from a Kinect Camera device as a person practices signs in front of it. The system generates the sentences on the screen being practiced in the sign. Our system uses depth information right from the Kinect device and computes the time-series of a sign being practiced. The system is trained on the four sentences using Random Forest. It evaluates the input sign on the model established after training and displays the sentence against the sign practiced on the screen as suggested (predicted) by model. Our tests report an accuracy of 85% using this approach. Our work can be extended to commercial scale based on the schema proposed by the project.

Chapter 1

Preliminaries and Introduction

1.1 Overview

The Project title states "Real time Video based Pakistan Sign Language Recognition". Real time gives the sense that system will be producing output just as it is given an input i.e. very small time laps. Real time performance is one of the most demanded quality of a machine learning based software product. The video based approach means that we will be using multiple videos of signs for training (tuning) the model in order to use this model for purpose of prediction of an unseen example (practiced sign). The segment "Pakistan Sign Language" conveys that the signs that will be predicted by the system are of Pakistan Sign Language.

The signer will be doing the sign in front of a Kinect Camera device. This is passed to the system. System evaluates the sign on model already built and outputs the text against the sign. Many such systems are developed and many more are in form of complete commercial products e.g. Motion Savvy and Microsoft Kinect Sign Language Recognition. This product does not intend to challenge the previous work rather this is an attempt to expand the cloud of this technology to Pakistan Sign Language Recognition.

1.2 Motivation

Around 360 million people around the world (5% of world's population) are suffering from some kind of hearing loss and many are those who cannot speak. This creates a sub-conscious barrier between such disabled persons and rest of the society. Such subjects use Sign languages to communicate. Many countries have developed technologies that could help such participants remove this barrier in order to get a healthy and transparent society. But sadly there's been very little work done in Pakistan Sign Language (PSL) technologies. PSL Recognition systems created till now are limited to numbers and other signs like directions, colors and body parts etc. Most of PSL technologies are static sign language recognition.
System proposed is a dynamic sign language recognition technology that takes live video of PSL signs and predicts its meanings in textual form. This can contribute in flourishing of PSL and disabled people.

1.3 Objective

The objective of this project is to come up with a technology that takes visual input (live video) of a sign being practiced in Pakistan Sign Language and predicts what it means in textual form.

1.4 Target Group

The intended subjects of this technology are the people who suffer from any kind of hearing loss and knows Pakistan Sign Language and those who want to communicate to a mute person and do not know Pakistan Sign Language.

1.5 Problem Statement

Hearing impaired or mute people are cornered due to inability of communication with other people. Currently in Pakistan, there is no technology that could help them with communication. There is work done on this technology but that is only limited to static sign inputs. There is needed a system that works on dynamic inputs and could serve the purpose of Sign to text translation in real time.

1.6 Solution proposed

A video based machine learning approach is proposed that takes in video in real time and predicts the text against a sign practiced in video. This system can be trained on some pre-specified set of signs and a model for classification is established. Later a sign video can be evaluated on this model and classified accordingly.

1.6.1 Motivation to use Kinect device for SLR

The conjunction of depth map and color image from Kinect sensor could produce great contribution for sign language recognition in three aspects. First, with the depth map, back-ground modeling becomes more simple and robust. We can easily and accurately extract human body part from color images. Second, in previous 2-D solutions, how to track hands is a difficult task. However, the skeleton information, developed from depth map, can be utilized to locate the positions of hands robustly and in real time. Third, beyond the traditional 2-D features, Kinect senor can provide some novel 3-D features, which are quite useful and hence and improve the performance of sign language recognition. These advantages will emerge in our following experimental results.

1.6.2 Motivation to use features from hand finger and skeleton

These actually gives a visualization of how a hand performs a sign in physical and how an interpreter can interpret it. As a finger is bent, its distance from the successive finger changes, angle of the defect (joint) is also effected and the distance between the joint and a hull around the hand also changes. So a single moves provides uniqueness in all three dimensions of main features. This justifies that these features would be good to be used for unique pattern identification.

Figure 1.1: Skeletnon of Hand

1.6.3 Motivation to use Random Forest Classifier

Random forest is actually an ensemble based learning in which we used multiple classifiers each a decision tree, and then predict the sign using all those trees. In the end, we assign probabilities to each class as with the ratio of predicted by trees and lastly we pick it based on the probability. Moreover, our data is non-categorical or numeric (ordinal). Random works really well when the features are known. For each of the tree, it would apply multiple splits based on minimum entropy and maximum information gain. Do in this case, Prediction is simpler i.e. just parse through the tree and predict the leaf class. Prediction does not take too much time. Prediction is ensemble so not a much chances to over fit on some class.

1.7 Scope:

This project can be installed to different scope domains like in hospital for training, communication and prescriptions. Following scenarios can use a furnished and modified form of this project.

- **Training:**

  There are patients in the hospital who have partial disability to speak, they use a bit of sign language to communicate. Such people can use the product. Also there are others who are deaf but they need to be trained how to speak i.e. how to twist the tongue, clump the lips and direct your voice. So they can be trained and communicated using this sign language machine by an ordinary person who does not know sign language. Moreover there are others who are complete deaf or mute, they also use sign language. For each of the above case, different sets of sentences could be selected for communication and make a custom model and use it.

- **Communication with doctor:**

  A doctor who does not know sign language can be made able to understand the problems of the mute patient. We can also enable the doctor to do the prescription to a deaf person using this machine by adding feature to replace the words with sign itself.

1.8 Some related famous products:

Following are the examples of already available state of the art products that are available to general public. Their technology in some aspects to the technology of the concerned project.

- DICTA sign [1]
- Motion Savvy [2]
- MS Sign Language Translator [3]

1.9 SWOT Analysis

Strengths	**Weaknesses**
Performance : Now a days, performance is main concern. System under consideration performs well in a constrained environment as specified in weaknesses. **Time:** This system works in real time so no more time lapses. **Reliability:** Reliability is the best a promise a product can make to its users. The system assures reliability by assuring all software development measures and steps. **Luminance:** Almost all other systems serving the purpose are confined by luminance i.e. proper lightning is necessary for system to work properly. Which is not the case with the system being proposed.	**Supplements:** A Kinect Camera device is needed in order to get input as no ordinary camera can serve the purpose of depth frame capturing. **Portability:** A laptop is enough to get the system working but it needs a Kinect Camera to be available also. Which is not an every place utility so needed to be carried along. More else the system is going to need to be configured before use. Not an instant plug and play module. **Distance:** The system works only when the sign is being practiced at a constrained distance of # ft. Else way, the sign goes undetected and hence can't be processed.
Opportunities	**Threats:**
open source software is easily available to edit and extend. Their debugging is also easy, as it is open source so many people can extend it for their purposes and can debug.	System pose no physical or social threats to anybody. It will only benefit those who use it. But yes, in a way, this may prevent someone to learn PSL by themselves

Chapter 2

Review of Literature

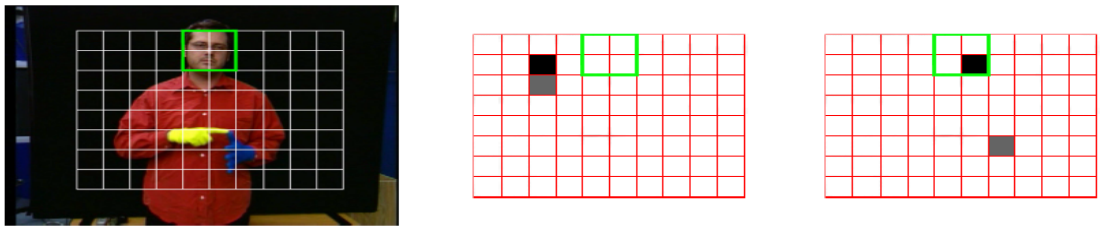
There can be many techniques and approaches to address a problem. Many solutions can be developed to solve a single a problem. We studied a bunch of different approaches in the literature of SLR and analyzed them so that to study what benefit an approach over another. These approaches involved sub-unit based approach to extract hands from background, static gabor filter based approach to background subtraction, ESF descriptor approach to extract features and many more. A few of the major approaches that helped us to get a deeper insight and derive a strategy are given in below sections.

2.1 Sub-unit based approach

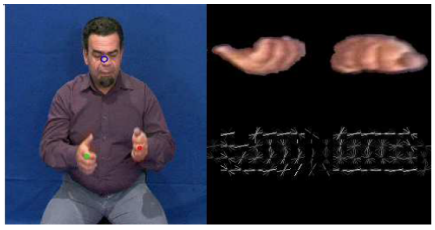
This is a sub-unit based Sign language recognition culminating in a real time Kinect demonstration system. This approach uses different types of sub-units e.g. Location, Motion and Hand-shape sub-units tracked and identified by both 2D and 3D tracking data. These sub-units are then combined using a sign level classifier. Further two approaches are used for time-series processing. First is Markov Models (in form of DTW i.e. Dynamic Time Warping) to encode temporal changes in sub-units. Second is uses Sequential Pattern Boosting to apply discriminative feature selection at the same time as encoding temporal information. This approach is more robust and performs well in signer independent tests and improves accuracy from 54% to 76% and then to 85.1% for signer

independent tests. [4]

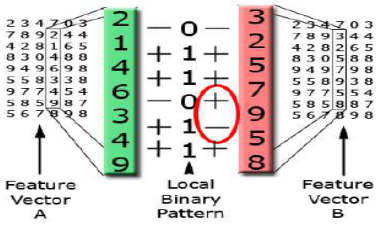
Figure 2.1: Sub-Unit Based Approach



(a) Location based sub-units



(b) HOG Hand Shape descriptor



(c) LBP Motion based descriptors

2.2 Static Gabor filter based approach

In this Research, Kinect Camera was used for input purposes keeping in mind that in this case input were images of static signs. Hand was segmented based on depth information right from the Kinect. Then hand shape features used are based on Gabor filtering of intensity and depth images. The purpose of using Gabor filter was that Gabor filter is apt at capturing contrast on image patch at different scales. Lately Random Forest was used for training and classification purposes on feature values obtained by Gabor filtering. A mean precision of 75% was obtained following the considered approach.[7]

2.3 ESF descriptors

Another approach used in previous computer scientists used ESF descriptors to unique identify different signs (static images). They used ToF (Time of Flight) cameras but techniques works perfectly fine with Kinect inputs just needs to be configured accordingly. ESF features were extracted from training samples and then trained with MLRF (Multi

layered Random Forest). This is a simple approach which sacrifices accuracy for quick results. An accuracy of 85% was observed practicing this approach.[6]

2.4 Skin Intensity based hand-feature extraction

This approach proposes a technique that first takes intensity of skin color of sign and then segment the face, hand and elbow based on these intensity values. There are water-marks on the interface. In the few starting frames, signer has to place his elbow and hands on the specified water-marks. With these frames, algorithm is capable of finding the range of intensities of skin pixels. Then segment each of the frame to get the hand, elbow and face-region. The flatness of hand-region, CG of hand w.r.t face, area of hand, the direction of hand motion (depicted in below diagram) and the direction of hand-region by elbow angle.[8]

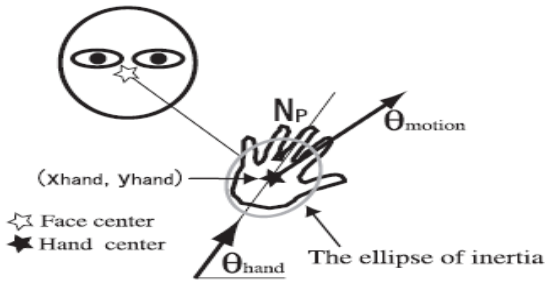


Figure 2.2: Skin Intensity based hand-feature extraction

Feature set is fed to HMM (Hidden Markov Model).Which is used for prediction of signs.

2.5 DTW based approach

This approach is the most similar one to the project being presented; in this the whole words are being translated from sign videos not gestures. For this purpose, system takes the input from the Kinect as a series of frames. Then for all these frames, joints of interest are extracted which are the joints of two fingers in the ideal case. The distance between these joints and the Cartesian coordinates of the joints of interest (taking the origin at the

center of the spine of signer's body) are taken as features. As these features are extracted now a time series is built for the whole video. Later the time series is trained using the DTW. DTW matches the frames in the two (trained time-series and test time-series) and then predicts to the class with which it has lowest error. DTW on the backend uses an HMM. The most prominent feature of this pilot study was that the signs are taken from Pakistan Sign Language. [5]

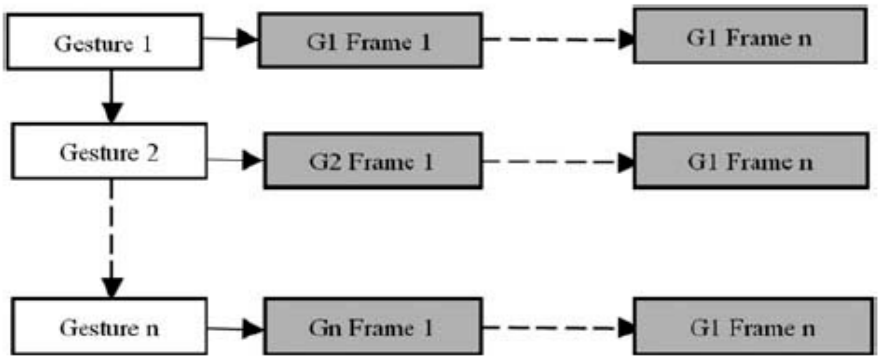


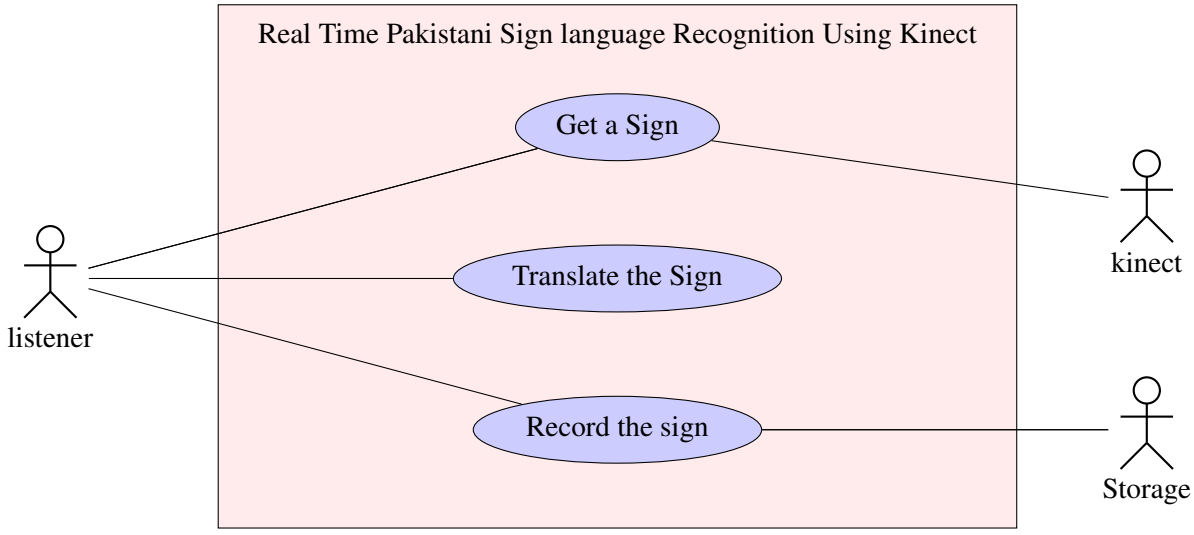
Figure 2.3: Different Gesture Format

The following picture demonstrates the dataset shape and organization

Chapter 3

System Analysis and Design

3.1 Use Case



Where the color spectrum from black to brighter gray represents the primary, secondary and outstanding actors

3.2 Get the sign

Table 3.1: Get the Sign

Id	1
Title	Get Sign
Description	Start getting the sign to be translated from the Kinect device.
Actor	Listener
Pre-Condition	System must be 'On'.
Post Condition	Frames of sign are being received live from Kinect Camera by the software.
Success Scenario	Learning module receives frames from the Kinect Device.
Alternative Flow	Corresponding error displayed to user and program aborts.
Stakeholders	Listener(operator), Signer.
Special Requirements	Kinect must be powered, Kinect is successfully connected to the system.
Open Issues	The system when gets the sign then it stores the sign in an excel file. This file sometimes gets corrupted because of some internal file system error.

A user arrives on the interface terminal. System presents him with three buttons labelled as 'start', 'end' and 'Predict'. If user wants to get the sign recorded from the Kinect, he would press the 'start' button. The system has now started recording the sign. As the sign duration ends, the user has to press the 'end' button. As the 'end' button is pressed, sign is saved in the directory and can be used for further operations.

3.2.1 Translate the sign

Table 3.2: Translate the sign

Id	2
Title	Translate the sign
Description	Sign is being received in form of series of frames from the Kinect Camera. Now this sign needs to be translated if Listener opts to translate.
Actor	Listener
Pre-Condition	Program started receiving frames from the Kinect in form of series of frames.
Post Condition	Sign is evaluated and processed by the system and translated to a label or voice chunk.
Success Scenario	Sign is translated to a right label or voice chunk.
Alternative Flow	Sign is translated to a wrong label or voice chunk.
Stakeholders	Listener(operator).
Special Requirements	Kinect must be powered, Kinect is successfully connected to the system,Sign must be recorded and stored successfully.
Open Issues	A naïve sign if done would be predicted to the class closest in features. So validity is only ensured if a sign is done from the predefined sentences.

The signer has recorded the sign using the 'start' and 'end' button. Now he presses the 'predict' button. As he presses the button the system looks for the sign in the corresponding directory and then predicts it against the model. The 'label' is presented to the signer in the lower right text box.

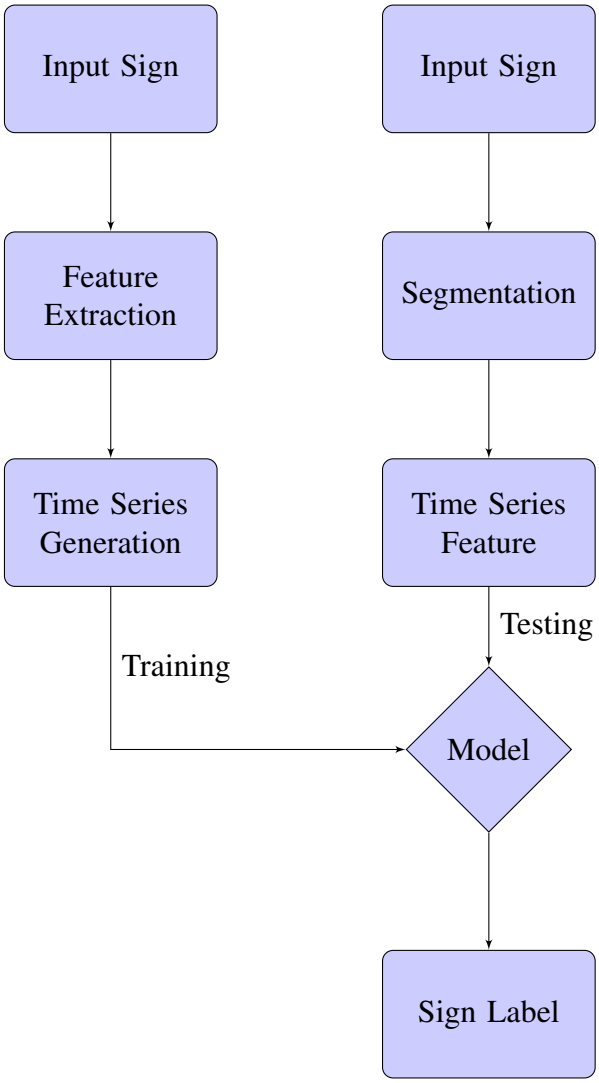
3.2.2 Save the Sign

Table 3.3: Save the Sign

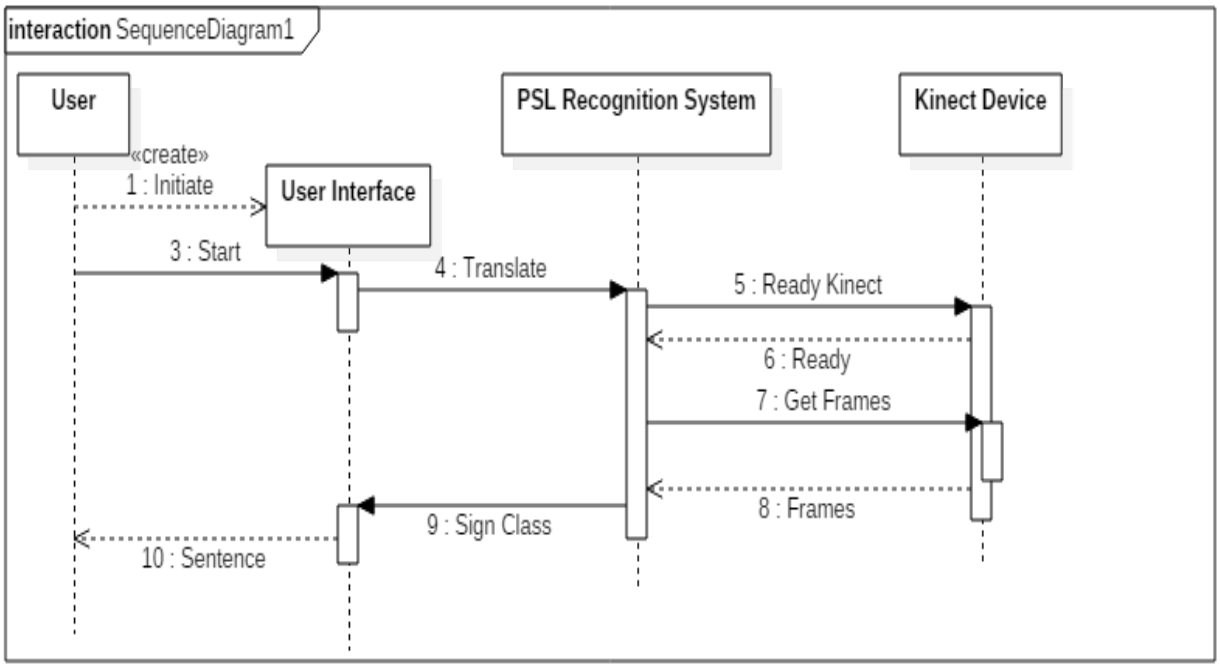
Id	3
Title	Save the Sign
Description	Sign can be saved to some repository and can be translated later if wanted.
Actor	Listener
Pre-Condition	System must be 'On'.
Post Condition	A sign is saved in form of a video to a repository.
Success Scenario	Sign was saved to repository.
Alternative Flow	Corresponding error to user and system rolled back to previous safe point.
Stakeholders	Listener(operator)
Special Requirements	Kinect must be powered, Kinect is successfully connected to the system, All the libraries are successfully imported.
Open Issues	Note that only one sign can be saved for the future work not more than one.

The operator will record the sign using the 'start' and 'end' button. Now the operator presses the 'start' button the data will be start storing when the operator clicks on 'end' button, the storage of the sign will be stopped.

3.3 Flow Diagram



3.4 System Sequence Diagram



Chapter 4

Dataset

4.1 Dimensions and Size

In this section, we will explore the size and dimension of the dataset. How it is stored in the storage. Moreover, we are going to know, how dataset is built. How many instances are there? What does an instance look like? And much more.

4.1.1 Size:

The dataset consists of 4 sentences (initially) to reflect the working and performance of the system being developed. The four sentences of Urdu are:

1. Khush-Aamdeed.
2. Ap kese hain ?
3. Apka naam kia hai ?
4. Allah Hafiz

So dataset contains samples from each sentence. To create a balanced dataset, we've taken 22 instances of each class (sentence). Each instance is a video of sign being performed in front of Kinect camera. Each video is stored in form of an Excel file with each record

representing a frame of the video. Length of videos is variable i.e. a sign may long 2 seconds and some other may long for 3 seconds. For a class, there may also be variable video lengths. Each file consists a number of rows representing frames in the video.

4.1.2 Basic unit

Each video contains 'N' number of frames. 'N' varies from video to video and further varies from sign to sign. Each frame record contains 'M' number of features from the corresponding frame. These features of frames are the basic units of which an instance is composed of. 'M' varies from frame to frame. All the features from the frames of the videos are numeric and quantifiable.

4.2 Analysis:

In this section, we will analyze the significance of dataset for building some model. We will see how well the dataset instances generalize a sign? How missing data occurs? What potential effects it can pose to the machine learning? How missing values are handled? All this analysis will be done in the following section.

4.2.1 Generalization

For the generalization purpose, we've taken the signs from 3 different signers so that system does not over fits a single person. Each signer has performed 8 signs. Moreover, the video length is also variable that introduces some degree of generalization to dataset.

4.2.2 Missing Values

As discussed, the feature length for a frame is variable. So for a file, there are frames stacked to each other in the xlsx file. So for a feature 'X' in a feature vector of frame 'A', 'X' may not be present in another frame 'B' causing a missing value (NA). Our dataset in contains a bundle of missing values and dataset is very noisy. Moreover, due to the

variance in the length of the feature vector, it becomes very difficult to handle and use this data to make a model due to difficulty in data imputation.

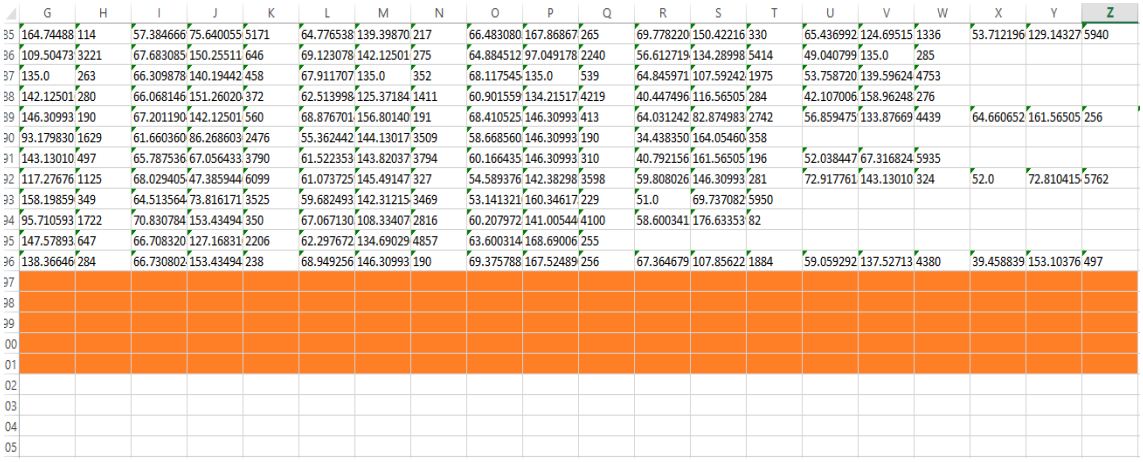
137.72631	355	42.720018	78.084852	1988	39.012818	101.39213	1405	31.400636	164.87599	181	32.649655	148.21908	682	42.047592	135.0	324			
136.33221	522	42.720018	122.28886	1387	31.780497	135.0	413	33.120990	146.30993	784	42.047592	146.30993	281						
164.05460	271	33.615472	126.41637	912	44.045431	144.46232	300	44.011362	122.66091	1370	31.953090	142.12501	380	32.756678	143.13010	781	40.012498	143.13010	229
116.70722	1792	40.049968	123.08878	1587	32.249030	143.13010	781	40.049968	135.0	343									
142.12501	485	48.548944	99.520021	1792	41.109609	128.41805	1602	32.249030	143.13010	781	40.049968	157.16634	229						
131.24144	1416	30.610455	173.65980	56	32.249030	149.98986	625												
90.0	1907	40.049968	137.94252	1485	32.202484	162.32894	568												
160.55996	181	46.043457	135.0	229	43.462627	133.66778	1213	31.320919	146.30993	343	31.240998	128.65980	328	32.202484	149.03624	755			
171.86989	57	44.384682	130.13548	1387	31.320919	141.34019	343	31.064449	129.80557	333	31.622776	155.55604	512	41.048751	146.30993	190			
108.43494	691	44.283179	142.76516	1110	31.304951	116.56505	284	31.064449	143.84181	427	32.202484	159.48668	563	41.012193	149.03624	324			
126.86989	320	42.379240	124.50852	1345	30.479501	166.98564	384	41.048751	160.34617	168									
121.67546	1379	43.863424	149.03624	229	46.872166	171.86989	57	43.462627	146.94417	1026	32.202484	161.56505	102	31.240998	168.69006	133	32.202484	156.80140	400	41.01
149.03624	286	35.227829	159.44395	351	33.105890	130.60129	337	31.890437	160.34617	512	39.012818	164.74488	114						
135.0	229	43.081318	117.89727	1250	30.528675	147.09475	305	30.805843	142.12501	280	31.780497	155.55604	576						
122.28886	1387	31.764760	147.09475	305	31.400636	135.0	783												
117.34987	1236	46.010868	116.56505	284	47.010637	133.15238	458	44.384682	134.47436	1046	31.304951	135.0	307	33.105890	144.46232	677			
140.90614	387	44.011362	134.51030	1162	32.015621	153.43494	229	31.384709	147.09475	305	33.105890	147.42594	661	41.0	156.80140	191			
135.0	229	47.010637	135.0	413	40.853396	126.86989	810	30.886890	153.43494	337	31.240998	159.22774	536	41.0	164.74488	114			
119.74488	593	44.011362	129.61068	1273	31.764760	135.0	307	32.015621	150.25511	262	33.105890	160.55996	238						
125.99149	1259	35.057096	144.46232	390	32.893768	149.03624	229	32.756678	171.86989	66	32.649655	169.99202	173	36.674241	138.17983	461			
153.43494	278	31.384709	171.86989	66	33.526109	138.36646	568												
149.03624	286	29.410882	146.52372	688															

Figure 4.1: Missing values in Dastsets

Above figure shows the missing values in as blank records.

4.2.3 Missing Records

As video length is variable, so a video 'A' may have 'N' number of frames and another video 'B' may have 'P' number of frames. The difference between 'P' and 'N' is the missing frames length in the corresponding video.



	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
35	164.74488	114	57.384666	75.640055	5171	64.776538	139.39870	217	66.483080	167.86867	265	69.778220	150.42216	330	65.436992	124.69515	1336	53.712196	129.14327	5940
36	109.50473	3221	67.683085	150.25511	646	69.123078	142.12501	275	64.884512	97.049178	2240	56.612719	134.28998	5414	49.040799	135.0	285			
37	135.0	263	66.309878	140.19442	458	67.911707	135.0	352	68.117545	135.0	539	64.845971	107.59242	1975	53.758720	139.59624	4753			
38	142.12501	280	66.068146	151.26020	372	62.513998	125.37184	1411	60.901559	134.21517	4219	40.447496	116.56505	284	42.107006	158.96248	276			
39	146.30993	190	67.201190	142.12501	560	68.876701	156.80140	191	68.410525	146.30993	413	64.031242	82.874983	2742	56.859475	133.87669	4439	64.660652	161.56505	256
90	93.179830	1629	61.660360	86.268603	2476	55.362442	144.13017	3509	58.668560	146.30993	190	34.438350	164.05460	358						
91	143.13010	497	65.787536	67.056433	3790	61.522353	143.82037	3794	60.166435	146.30993	310	40.792156	161.56505	196	52.038447	67.316824	5935			
92	117.27676	1125	68.029405	47.385944	6099	61.073725	145.49147	327	54.589376	142.38298	3598	59.808026	146.30993	281	72.917761	143.13010	324	52.0	72.810415	5762
93	158.19859	349	64.513564	73.816171	3525	59.682493	142.31215	3469	53.141321	160.34617	229	51.0	69.737082	5950						
94	95.710593	1722	70.830784	153.43494	350	67.067130	108.33407	2816	60.207972	141.00544	4100	58.600341	176.63353	82						
95	147.57893	647	66.708320	127.16831	2206	62.297672	134.69029	4857	63.600314	168.69006	255									
96	138.36646	284	66.730802	153.43494	238	68.949256	146.30993	190	69.375788	167.52489	256	67.364679	107.85622	1884	59.059292	137.52713	4380	39.458839	153.10376	497
97																				
98																				
99																				
00																				
01																				
02																				
03																				
04																				
05																				

Figure 4.2: Missing records in Dastsets

4.3 Padding

In order to do machine learning on the dataset we need each instance to be in the same shape and size. To achieve this purpose, we pad our data instances with zero.

4.3.1 Feature Padding

We first find the maximum length feature vector. Then we pad each of the record with length less than the maximum length with zero to make compatible to the maximum length feature.

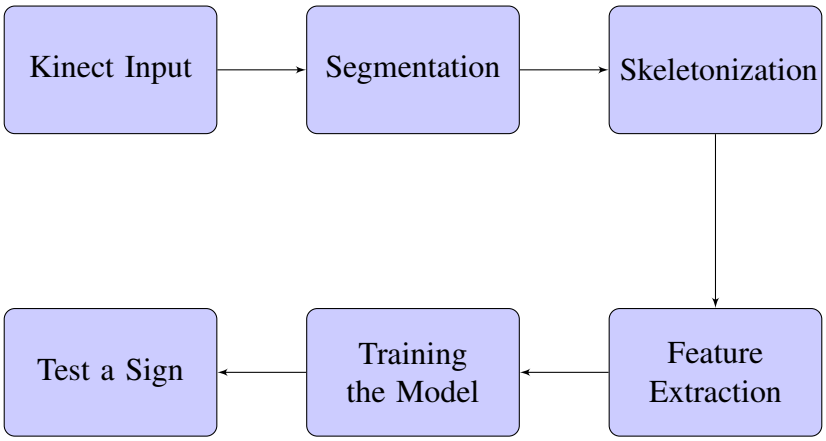
4.3.2 Frame padding

And lastly, we pad each video with zero vectors so that to make it equal to the length of maximum length video.

Chapter 5

PROJECT IMPLEMENTATION

5.1 Flow Diagram



5.2 Implementation

We take input from Kinect using the python library pykinect. We capture the depth frame from the event handler depth frame. This depth frame tells us the different to the object in each pixel. This distance is a 16-bit value out of which one bit is redundant and 3 bits are for player index. So each distance is a 12-bit value.

This 12-bit frame is then normalized to 8-bits by bit-by-bit shifting. This 8 bit frame can

be seen as an 8-bit image of gray scale. In which the nearer objects are darker than the far objects. So objects at minimum distance would be the darkest.

Figure 5.1: Gray Image Formation

Note: Any object beyond the range of the Kinect's optimum range is the darkest.
Obtained 8-bit gray image is then segmented based on the minimum distance objects from the Kinect. Keep in mind that our project bears a constraint that the signing hand is the only object that is at the minimum distance from the Kinect. So by segmentation based on the minimum distance we get the hand region and using it as a threshold to binary image we will get a binary image with the only signing hand. A segmented image of the above

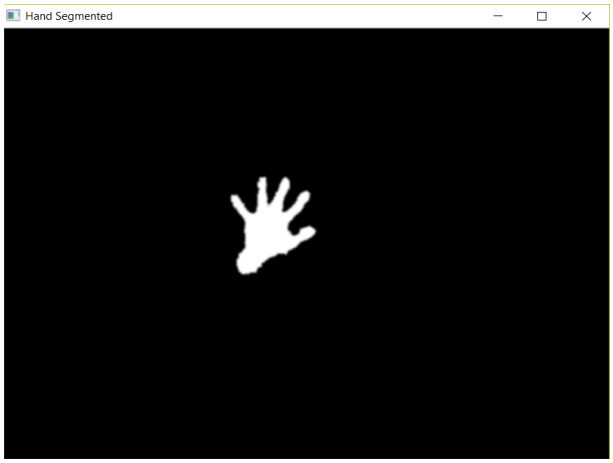


Figure 5.2: Segmented Image

shown gray image is shown.

We then need to identify all the fingers and the center of the so that these points could be used to generate the feature vectors. For that, we first find the contours (boundary) of the hand. We also find the convex hull in the of the hand shape. Now try and find the points that are common to both the convex hull and the contour of the hand. Both the Convex hull and contours are extracted using OpenCV python. These are the points to be identified as finger tips and corners of the hand. Finger are identified and now these can be used to extract features. Now we find the sum of x-coordinates and also the sum of

Figure 5.3: Skeleton Image

y-coordinate of all the white pixels (hand-region). Now we divide both x-sum and y-sum by the total number of hand pixels. The obtained point i.e. (x, y) pair is the centroid of the hand.

Knowing all the main points of the hand region in the frame, we now extract features from the frame to make a feature vector. There are following four main features used:

- **Consecutive fingers gap :**

  We take the gap of the two consecutive fingers (our choice whether counting from left to right or right to left). This gap is calculated in the following way: Say the two fingertips were 'A' and 'B' points.

$$\sqrt{[(A.x-B.x)^2+(A.y-B.y)^2]} = Distancevalue(numberofpixels) \qquad (5.1)$$

- **Angles of the defects :**

  By using the convexity defects (extracted by convex hull and contour of the hand), we get to find the deepest point of the defect which is the considered to be the joint between the two fingers. Now for two consecutive finger tips 'A' and 'B', we form two vector:

  1. From the fingertip 'A' to the middle joint of 'A' and 'B'.
  2. From the fingertip 'B' to the middle joint of 'A' and 'B'.

  Now for the two tips 'A' and 'B', we get vectors say 'a' and 'b'. Now we apply the following formula.

  $$a.b = |a|.|b|cos\theta \tag{5.2}$$

  From here, if we find the magnitude of 'a' and 'b' using the distance formula from point 'A' to joint and point 'B' to joint. And finally, we get angle from the above formula. Now we do that for every defect and use these angles in the feature vector.

- **Area of the hand :**

  Area of the hand is number of pixel in the hand region in the respective frame.

- **Normal distance to convex hull :**

  Normal distance from each of the joint to the convex hull. The distance is calculate for number of unit pixel and using the simple distance formula.

We concatenate all these triplets and area of the hand to form the feature record for the frame. We do that for all the frames and then take them record by record and make a whole file where each record represents a frame in the video.
We combine all these videos of signs and then place all same signs in the same directory and same for all the other classes. As presented in the previous portion (4.3), we pad each video to the length of the maximum size video so that each instance be of the same size and be feasible to train. For padding purpose, we use the Keras python.
As all the dataset is in its final form so let's move in to build the machine learning model for the prediction of signs. A random forest was trained on this dataset with 50 number of trees each of depth 24. Class label was assigned based on majority votes of the tree.

Dataset is partitioned to two sets of 75% part of the whole dataset for training purpose and the rest 25% part of the dataset for evaluation of the model. Sklearn in python is used to build the model.

The interface is built in tkinter python. It contains a window that shows the frame coming right from Kinect. Another field that present you with the label of sign being performed in front of the Kinect. There are also two more buttons labelled as 'start' and 'Predict'. 'Start' serves the purpose to allow all the users perform the sign in front of the Kinect. And 'Predict' button can be used to terminate the sign duration. As the signer is done with his sign (presses the 'predict' button), the sign would be converted into the feature vector and evaluated against the model. In the result of evaluation, a label corresponding to the sign is extracted. This would be presented as the label (textual meaning of sign in Urdu) of the sign performed on the interface screen.

For prediction purpose, Random forest is used to classify the signs into sentences. Random forest is a simple but elegant ensemble machine learning technique. It is an ensemble of multiple simple machine learning classifier known as decision trees. It tries to predict class for the sign using different architecture trees i.e. different splits and bootstrapping on the train set. Then takes votes from these trees. And provides the results with majority votes. As far as Decision trees are concerned, these are simple trees with each node of the tree is a split criteria for a feature. Splits are chosen based on maximum information gain and minimum entropy. And at the leaf there is the class label. An instance when parsed through this tree satisfying different split reaches to its class label.

5.3 Baseline Results

As discussed and explained in the chapter-4, the complexity and noise of the dataset is huge. Moreover, to make it feasible, we padded the dataset; this also introduces some noise and bias in the dataset. Due to all those reasons, each frame carries some noise with itself and then later this noise convolves with noise of the whole video instance. Presence of all these noises in our dataset leads to a weak model. So the results (predictions) from this model are not fully reliable.

The accuracy achieved is 83% as observed on the 25% test dataset. Moreover, F1 measure is 0.83. The prediction time taken by a system of following specification:

- Intel Core i5 processor
- 8GB RAM
- No GPU used

Is 0.446 seconds approximately. Note that these results are considered as running the system in offline mode i.e. not from the interface. While using the interface accuracy drops because of the frame closing and getting latency introduced by the operations of interface.

5.4 Results

The results are obtained using 10-cross-folds. The simple split across the dataset is 90% for train and 10% for test for each fold. Accuracy and F1 score are reliable and significant.

Results using Random Forest classifiers

Results for all folds while using the Random Forest model are shown in Table 5.1:

	Fold-1	Fold-2	Fold-3	Fold-4	Fold-5	Fold-6	Fold-7	Fold-8	Fold-9	Fold-10
Accuracy	91.667	91.667	91.667	83.333	58.333	83.333	75.0	83.333	75.0	100.0
F1-Score	0.9144	0.9142	0.9142	0.8035	0.5761	0.8309	0.7416	0.8309	0.7321	1.0

Table 5.1: Results using Random Forest classifiers

The accumulated and finalized results obtained the above given technique and prediction using Random Forest are:

Cumulative Accuracy of the dataset: 0.825 (82.5%)

Cumulative F1 Score of the dataset: 0.83

Entropy and Gini both were used as a splitting criterion and in the test many experiments were run using different number of trees as shown in the figure 5.4 below.

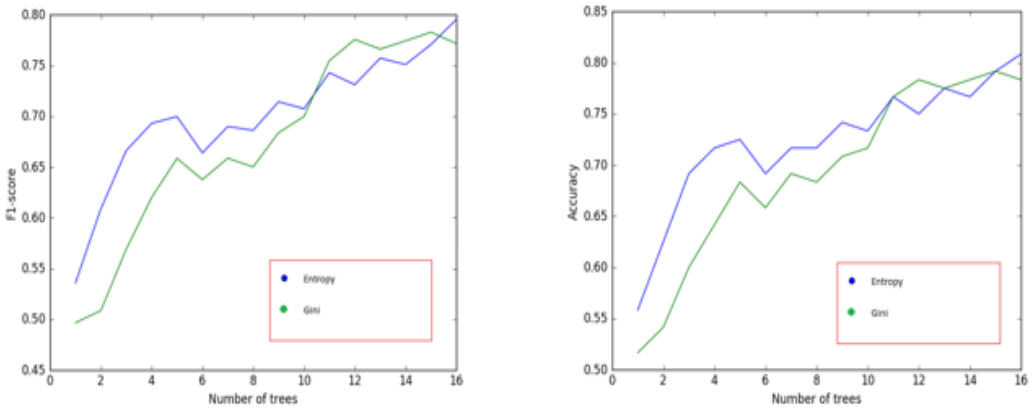


Figure 5.4: Random Forest Result Graph

Alternative Results using SVM classifiers

Results for all folds while using the SVM model are shown in Table 5.2:

	Fold-1	Fold-2	Fold-3	Fold-4	Fold-5	Fold-6	Fold-7	Fold-8	Fold-9	Fold-10
Accuracy	100	83.333	66.6666	75	50	58.333	41.666	66.6666	75.0	100.0
F1-Score	1.0	0.8333	0.675	0.7428	0.5	0.5863	0.3653	0.675	0.7321	1.0

Table 5.2: Results using SVM classifiers

The accumulated and finalized results obtained the above given technique and prediction using SVM are,

Cumulative Accuracy of the dataset: 0.71 (71%)

Cumulative F1 Score of the dataset: 0.72

Experiments were run using different degrees of the hyperbola separating the classes. Results are shown in diagram.

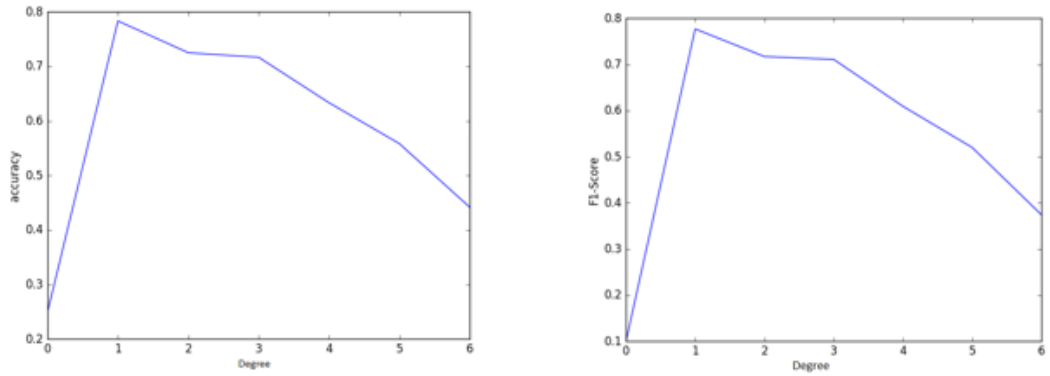


Figure 5.5: SVM Result Graph

Alternative Results using Decision Tree classifiers

Cumulative Accuracy of the dataset: 0.53 (53%)

Cumulative F1 Score of the dataset: 0.55

Many splitting criterion used are entropy and gini with ranging the depth of tree. Results are shown by fig below.

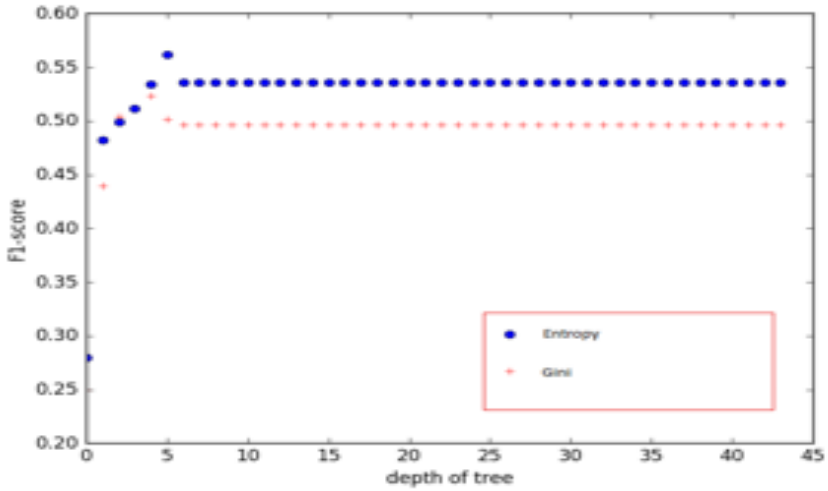


Figure 5.6: Decision Tree Result Graph

Alternative Results using Naïve Bayes classifiers

We also used Naïve Bayes to serve the purpose or build the model. Naïve Bayes worked also fairly well as shown by the following results.

Results for all folds while using the Naïve Bayes model are shown in Table 5.3:

	Fold-1	Fold-2	Fold-3	Fold-4	Fold-5	Fold-6	Fold-7	Fold-8	Fold-9	Fold-10
Accuracy	50	66.666	50	50	25	25	25	41.666	75.0	58.333
F1-Score	0.502	0.675	0.521	0.475	0.182	171	0.214	0.375	0.555	0.497

Table 5.3: Results using Naïve Bayes classifiers

Cumulative Accuracy of the dataset: 0.45 (45%)

Cumulative F1 Score of the dataset: 0.41

5.5 Alternative flows and approaches followed

We tried many flows and strategies but at some point they collapsed or during the feasibility study, they were found to be not compatible with the problem statement. Following are the strategies as listed below.

1. **RGB and Depth-image overlap for segmentation** In the starting few frames user is asked to keep his hands in the front of his body and at minimum distance from Kinect. Now take 'AND' of these pixels of binary segmented image with the pixels of the RGB frame to extract the hand region. And now find the bounding box of this hand region in the RGB frame. Extract this region and find its HOG features and append these HOG features from all frames of the video to make feature vector of the sign video.

- **Problems with the approach**

  (a) The depth frame is zoomed in as compared to the RGB frame. When overlapping these region would also include some noise of the surrounding.

  - To compute HOG feature, Image must be in the 8 or 16 bit-depth. So to do that, we have to convert each frame to Gray first and then compute HOG. Which takes quite much time; we may lose some key frames. This would take much time and may not allow us to predict in real time for a much larger dataset.

  (b) HOG features approach would be redundant because for this approach; an instance feature vector would be very huge.

  - For Each frame; the segmented region would be different; we will have to pad the HOG frame to make the size of all frame equal.

2. **Only RGB hand-tracking and feature Extraction**

   We had done the hand tracking; our algorithm was tracking the hand with always changing hand shape for each successive frame. Tracking was Real time. Tracking was done using Dlib module that uses localized histograms to get features from a frame and then use these features to track the hand in the successive frames. HOG features were also extracted by converting each frame to gray.

- **Problems with the approach**

  - The Kinect RGB camera is quite noisy. So the hand region that was observed by the specified region from first frames was itself noisy. So while defining

the intensity range it would have considered the noise but that was handles by using only the region in the range of standard deviation. But then it starting also segmenting the face and neck region of intensity values. Same were the problems with segmentation based on HSV (Hue, Saturation and Lightness).

- Region filling in this region was a hectic and noisy job to do.
- After even segmentation, we were to use some features like HOG or SURF which were redundant in larger and bigger datasets.
- No significance to use the Kinect Device, if we are to use only the RGB frame.

3. **Hu-Moments and Zu-Moments**

   Hu- moments and Zu-Moment are state-of-the-art features to predict static objects and to check whether an object differs from others or not. They are rotation, position and scale invariant features that is why used in many of the applications. We did not extracted these from our frames because we found it these not worthy enough to serve the purpose in this project purpose.

- **Problems with this approach**

  - In the sign language, there is a great significance of hand angle (if hand is rotated or not). As these features are rotation invariant so this would consider a tilted hand and a straight hand as same.
  - These features are significant for only the case, when there are gray values in the region of interest (hand region) but when hand region is identified then it is all the same (flat) intensity values because there is no significant different in the distance from the Kinect Device.
  - No significant results are extracted for sign recognition in the literature.

4. **LBP features**

   LBP features approach finds the local binary patterns as suggested by the names. It goes across the whole image and finds the intensity shifts and captures those in a feature vector for a frame. These features were exempted during the feasibility study phase.

- **Problems with this approach**
  - These features are significant for only the case, when there are gray values in the region of interest (hand region) but when hand region is identified then it is all the same (flat) intensity values because there is no significant different in the distance from the Kinect Device. So no gray region would be in the hand region and hence intensity shifts will remain undetected.
  - No significant results are extracted for sign recognition in the literature.

5. **SIFT and SURF features**

   SIFT as suggested by the name are Scale invariant features which is a plus point in our case. These features apply on RGB and Gray images.

- **Problems with the approach**
  - To compute SURF and SIFT features, Image must be in the 8 or 16 bit-depth. So to do that, we have to convert each frame to Gray first and then compute features. Which takes quite much time; we may lose some key frames. This would take much time and may not allow us to predict in real time for a much larger dataset.
  - SURF and SIFT features approach would be redundant because for this approach; an instance feature vector would be very huge.
  - For Each frame; the segmented region would be different; we will have to pad the SIFT and SURF frame to make the size of all frame equal.

5.6 Interface Description

There are three blocks on the interface. The left block displays the depth stream (transformed to gray image) from the Kinect. The upper right block displays the segmented and skeletonized frame against the frame in the left block at the same time. And lastly, the lower right block contains four item.

- Text/Label window: Displays the output label against the sign perform in front of the Kinect.
- Start button: Allows user to start performing (recording) the sign.
- End button: Allows the user to stop performing the sign.
- Predict button: Once the sign is performed and saved using the 'end' button. User can predict this sign to sentence label using the 'predict' button.

An interface depiction is given below in which a sign is being performed.

Figure 5.7: User Interface

When you press the ‘predict’ button, the label or sentence against the sign is displayed in the Text/Label box as predicted by the model. As depicted by the following image.

Chapter 6

Future Recommendation

This project is the minimum working example of a bigger, more universal and applied system that can be extensively used in many industry based and personal use application. This project serves the purpose in 4 sentences initially. Although the NLP problem is computationally infeasible, but if make an attempt to fit to some legitimate, small and limited scope application, we can cover most of the domain.

Currently system works for only single handed gesture. Develop a technique (most probably using RGB and Depth both) to segment both hands from the frame and then you can apply it universally to any sign.

Now we can extend this system by identifying a number of generally and frequently used sign language sentences. Prepare a dataset out of them by using at least two different signers. Feed into the system and retrain the model. Now attach (configure) this model to the prediction by configuring the file name and you are good to go. Use it in the environment.