Islamic University of Technology (IUT)

Supervisor

Md. Kamrul Hasan, PhD

Professor

Dept. of CSE, IUT

Co-Supervisor

Hasan Mahmud

Assistant Professor

Dept. of CSE, IUT

A thesis submitted in partial fulfilment of the requirements

for the degree of B. Sc. Engineering in Computer Science and Engineering

Academic Year: 2019-2020

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT)

A Subsidiary Organ of the Organization of Islamic Cooperation (OIC)

Dhaka, Bangladesh

March 04, 2021

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by under the supervision of Dr. Md. Kamrul Hasan, Professor of the Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Dhaka, Bangladesh. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:

S. M. Rayeed

____

Student ID - 160041045

Gazi Wasif Akram

____

Student ID - 160041005

Golam Sadman Zilani

____

Student ID - 160041040

Approved By:

Supervisor:

Md. Kamrul Hasan, PhD

Professor

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT), OIC

Acknowledgement

We would like to express our grateful appreciation for **Dr. Md. Kamrul Hasan**, Professor, Department of Computer Science and Engineering, IUT for being our adviser and mentor. His motivation, suggestions and insights for this research have been invaluable. Without his support and proper guidance this research would never have been possible. His valuable opinion, time and input provided throughout the thesis work, from first phase of thesis topics introduction, subject selection, proposing algorithm, modification till the project implementation and finalization which helped us to do our thesis work in proper way. We are really grateful to him.

We are also grateful to **Hasan Mahmud**, Assistant Professor, Department of Computer Science and Engineering, IUT for his valuable inspection and suggestions on our proposal of Bangla Sign Language Dataset Generation using Depth Information.

Abstract

Sign Language Recognition (SLR) targets on interpreting the sign language into text or speech, so as to facilitate the communication between deaf-mute people and ordinary people. This task has broad social impact, but is still very challenging due to the complexity and large variations in hand actions. Existing dataset for SLR in our country is based on RGB images (converted to grayscale) and CNN is used for classification. However, it is difficult to design model to adapt to the large variations of hand gestures in the dataset, also the computational expense is high. Modern researches on other sign languages have shown that using depth information in Sign Language Recognition (SLR) gives better accuracy, which hasn't been introduced yet in our country. In this paper, we intend to build a complete Bangla Sign Language (BSL) dataset using depth information from depth images. In order to do, we'll be collecting our depth information from our captured image samples using MediaPipe which is a cross-platform framework for. building multimodal applied machine learning pipeline. It is quite a new and advanced technology in hand tracking and gesture recognition. As opposed to the existing image dataset, we intend to build a feature-based depth dataset, that, if accurately modeled, is more likely to give better results than the existing one in sign language recognition (SLR).

Keywords: *Bangla Sign Language, MediaPipe, Hand Landmark Model, Intel RealSense, Depth Information, Gesture Recognition, Sign Language Dataset, Hand Key-points, Hand Tracking*

Contents

1 Introduction

1.1 Overview

Sign language is a visual language that uses hand shapes, facial expression, gestures and body language. Hearing-impaired people use Sign Language to communicate with others, in fact it is the basic method of communication for them in their day-to-day life, but it is difficult to interpret for most of the people, so the communication needs to be developed, and that's where machine learning may become very handy. Pre-trained machine translator can be helpful to ease the interpretation of what a person is trying to express and therefore to ease the communication of hearing-impaired people with others.

From the technological aspect, with the advancement in technology, the ways of user interaction have been changing from physically operating devices via keyboard to, interacting without any touch such as gestures, virtual reality etc. Gesture recognition has been one of the main focuses in recent times in the field of Computer Science in various areas. 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 analyses of the gestures. As said, sign language is nothing but a collection of hand gestures, so it can be made easier to interpret the sign language to general people, and for this, several computer vision based solutions are getting well known nowadays in many countries.

In Bangladesh, almost 2.6 million people are deaf and mute, but not much has been done to help them communicate easily with the rest of our community. In 1974 a book named Bengali Sign Language Dictionary was published by National Centre for Special Education Ministry. Another book named Ishara Bhasay Jogajog was reprinted in 2015 for learning purpose of deaf children in Bangladesh. In terms of modern technology as well, Bangla Sign Language Recognition has got off to a start, when some of the Deep Learning enthusiasts (Sanzid et. al.)

[1] used Convolutional Neural Network to train an RGB image dataset and later evaluate the performance. Yet much has left to be done in Bangla Sign Language Recognition.

1.2 Problem Statement

The primary goals of Sign Language is easier communication between people who have auditory impairment and people who do not. For making this communication tech-based and software-oriented, Sign Language Recognition was introduced, and classification models are used for this. But prior to that, the machine (i.e. the model) needs to be trained properly, with sufficient amount of data samples, from which the model can extract the distinguishable features from different samples and later use that in classification. So, we need a Sign Language Dataset – that will contain samples or important features from Sign Language Alphabets.

Now, there are two types of image datasets - RGB dataset that uses color images and depth dataset using depth images (includes the z-coordinate). In Bangla Sign Language, the existing Dataset used for classification is an RGB dataset, no depth dataset has been generated yet in BdSL. So, we intend to develop an open access Bangla Sign Language Dataset using depth information (extracting key features from depth images) to get better accuracy than the existing RGB dataset when used for classification. In order to do so, we will be using MediaPipe framework for detecting hand key-points from our collected image samples, which is a relatively new technology in skeleton tracking, hand tracking and gesture recognition.

1.3 Motivation and Scope of Research

Sign Language Recognition has become a popular topic, ever since the researchers have incorporated the idea of using depth information into the recognition process. In different countries, many researches have been conducted to analyze the improvement in recognition rate using depth information. And, studies on other sign languages have shown that using depth information increases the accuracy a lot. On the other hand, not much work has been done there for Bangla Sign Language Recognition, that can ease the communication of hearing-impaired people to others. Although Bangla sign alphabets and digits are available for more than three decades, there is still a lot to do in this field from the perspective of Computer-Vision and Human-Computer Interaction.

In Bangla Sign Language, the existing open source dataset is an RGB dataset, no renowned work has been done using depth information. This motivated us to do something new in order to have some contribution in serving the impaired-people. We intend to generate a dataset based on depth information, so that it can be used in Bangla Sign Language Recognition, and give a better accuracy than the existing one, as delineated in many research works across the globe.

In our thesis work, we will be using MediaPipe framework to detect the hand key-points from the collected image samples, and then normalize the x, y values, and estimate the z-values from the image depth map. Using the coordinate values, we will be generating our dataset, consisting 21 3D co-ordinate values for each image sample. Generating distinguishable features from these co-ordinate values and hence enhance the dataset with more valuable features – can be a very promising topic to work on, as a continuation of this work. Also, in case of dynamic hand gestures, how accurately RealSense can detect the hand joint-points and keep track of those points while the hand is moving – will be good research work, as a continuation of working on RealSense. Very little work has been in Bangla Sign Language, in fact, almost no work has been done using depth information. Moreover, MediaPipe is a very recent technology for detection and tracking, so there is endless scope to work on this topic in future.

1.4 Research Challenges

Our thesis work is to generate a dataset, where the key challenge is to make the dataset work for a complex environment. There are multiple factors that affects while generating a dataset – first one is the variability in hand signs from user to user. Different users perform the same hand signs in different ways, which makes the task of the classifier difficult to extract necessary distinguishable information from the environment. Another one is the similarity in signs, in many sign languages, in fact, in Bangla Sign Language, there are several letters that looks very similar while performed in hand-signs. So, this is a challenge for us that the RealSense device can detect the joint-points with very little differences. Also, as we are using the depth sensing technology, the background is a vital factor. Noisy background deteriorates the performance of the depth sensors in RealSense, as a result, the device cannot detect the joints successfully. In order to deal with that, we need depth calibration. The next factor is the user variability – taking data from same kind of user makes the dataset biased and eventually any model be trained on that dataset will underperform. So, our challenge is to take samples from different users of different ages, races, genders so that the dataset can work well in a complex environment.

Another challenge for us is the lack of work being done using RealSense. It is the latest technology in depth sensing field, and outperforms the older ones by a margin. But the challenge is as it is new, not much work has been done using this device – so sometimes dealing with various functions, modules can be troublesome. Finally, the main challenge of this research is to consider the factor of user variability and to consider the goal to build a workable dataset in a complex environment.

1.5 Thesis Outline

In Section 1, we have discussed the overview of what we intend to do and why so, in a precise and concise manner. Section 2 deals with the necessary literature review for our study and there development so far. In Section 3, we have stated the skeleton of our proposed dataset, proposed methodology to extract information for the dataset, implementation process to construct the dataset and also our progress so far. Section 4 shows the Result analysis after a successful implementation of our proposed method. The final section of this study contains all the references and credits used.

2 Background study

2.1 Bangla Sign Language

Sign language involves non- vocal communication with a combination of hand gestures, lip patterns and facial expressions in order to identify as a meaningful expression. In this paper, we are focused only on the hand gestures.

In early 15$^{\text{th}}$ century, people with impairments were neglected and nobody respected them, because they were unable to communicate with the world. In the following 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, Abbe Charles-Michel delEpee 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) is complete and widely used, consisting of 26 signs that are used to express words from the English language.

In Bangladesh, community of people having hearing impairments is considered to be the largest minority community (based on language). In Bangla Language, there are total 50 alphabets. Among which, 11 are vowels and 39 are consonants. For all these 50 alphabets, according to Bangla Sign Language Dictionary, there are in total 38 different gestures. Also, for numbers from 0 to 9, there are 10 gestures to represent the sign digits. [Click here to Download]

Again, there are two ways to represent Bengali Sign Language alphabets - using one hand, or both hands. Here we will work with mainly gestures which are represented with one hand. Figure 1 shows the standard gestures for Bengali alphabet, using one hand. It is necessary to mention that, in case of using one hand, there are 38 characters and in case of both hands, the number is 36.

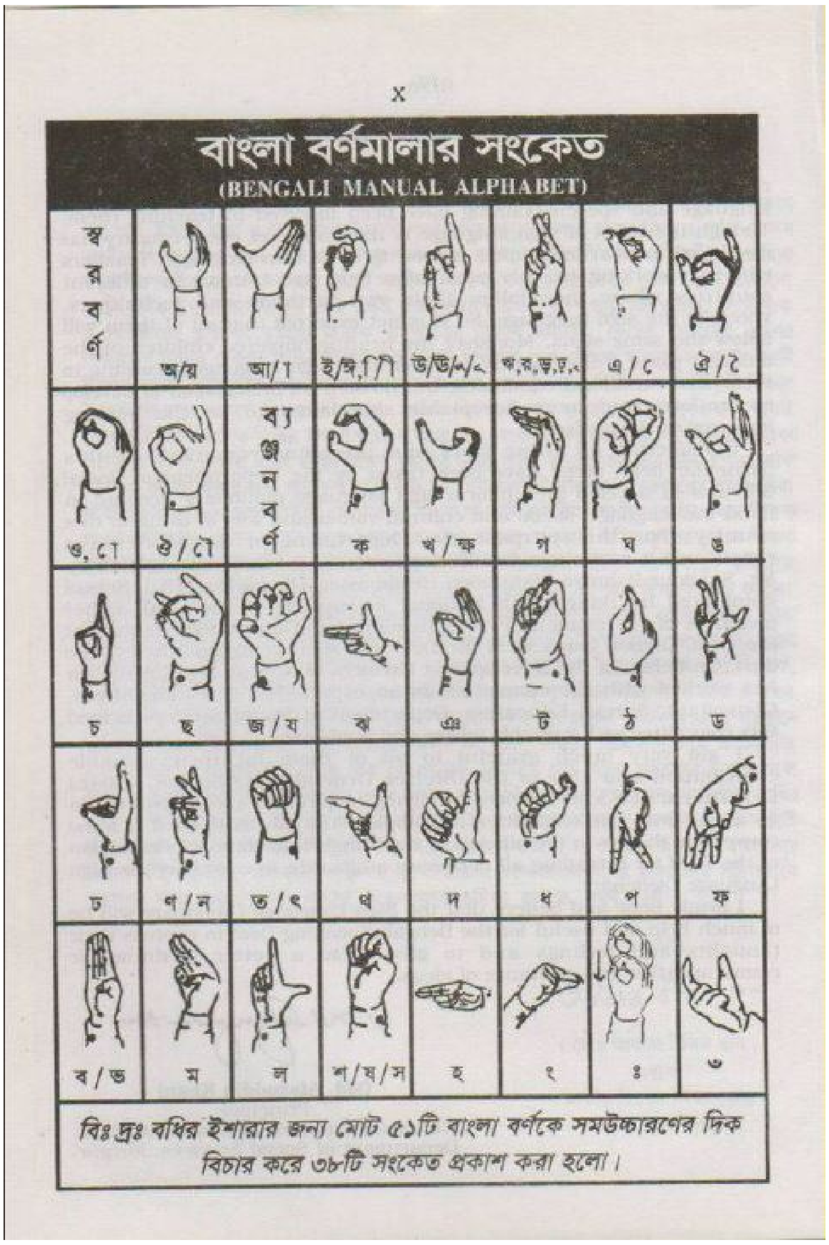


Figure 1: Bengali Sign Alphabets - One hand

2.2 Existing Dataset - Ishara Lipi

The existing dataset on Bangla Sign Language is Ishara-Lipi [1], from Sanzidul (et. al) published in 2018. This is the First Multipurpose Open Access Dataset for Bangla Sign Language Isolated Characters, in order to help the researchers to work with sign recognizer, machine translator and to develop the aiding tools.

The whole dataset is divided into two portions- one is for digits(1, 2, 3, ..., 9) and another is for characters(1, 2, 3, ..., 36). It is better to mention that, they have considered 36 characters of Bangla Sign Language using two hands. The digits dataset contains 100 sets of 10 Bangla basic sign digits (0, 1, 2 . . ., 9), collected from different deaf and general volunteers from different institutes. After discarding maximum errors and performing different preprocessing methods, 1000 images of Bangla sign language isolated digits were included in the final dataset. After collecting raw data, some effective preprocessing methods were performed for making those data usable for computer vision or any other development purposes.

And the characters dataset contains 50 sets of 36 Bangla basic sign characters, collected by the help of different deaf and general volunteers from different institutes. In Bangla Sign Language sign characters there have 6 vowels and 30 consonants by which they can finger spell all Bangla words. In Ishara-Lipi dataset, after discarding mistakes and preprocessing, 1800 character images of Bangla Sign Language were included in the final state. This dataset could be used to develop computer vision based or any kind of system that approves users to search the meaning of BdSL signs. This dataset is an RGB Image Dataset, consisting of color images, later converted to grayscale as a part of data preprocessing, prior to constructing the dataset. Steps of data pre-processing includes –

2.2.1 Steps in developing Ishara-Lipi Dataset

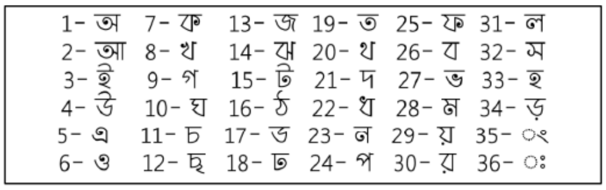
Proposed data collection and pre-processing method has been stated. There are six different states assigned to perform the entire process.[1]

- **Capturing images –**

  A comparatively wide sized BdSL image dataset has been arranged in this book. The dataset contains total 1800 images of sign characters and 1000 images of sign digits.[1] The authors had collect data from many Deaf School Community and took images of uncovered hands in white backgrounds.

- **Labeling data –**

  This move is essential to minimize the noise achieve in the capturing process and to improve the images quality. Some of BdSL character sign vary little from another. Images of this kind of characters seem intimate and enhance the exception of experiment. So, the characters were categorized individually – 36 characters by naming with numeric convention from 1 to 36 (1, 2, 3... 36). The naming convention of all characters are given below in a chart.[1]



1- অ	7- ক	13- জ	19- ত	25- ফ	31- ল
2- আ	8- খ	14- ঝ	20- থ	26- ব	32- স
3- ই	9- গ	15- ট	21- দ	27- ভ	33- হ
4- উ	10- ঘ	16- ঠ	22- ধ	28- ম	34- ড়
5- এ	11- চ	17- ড	23- ন	29- য়	35- ◌ং
6- ও	12- ছ	18- ঢ	24- প	30- র	36- ◌ঃ

Figure 2: Numeric Representation of BdSL Characters

- **Cropping images –**

  The captured images cannot be used without cropping for any character recognition purpose. Cropping is necessary for better feature extraction from the image. The images are cropped to show the hand region (region of interest).[1]

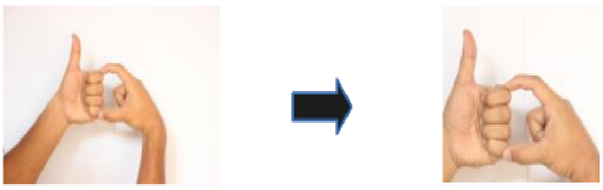


Figure 3: Cropped Images

- **Resizing image and convert to gray scale –**

  For making Ishara-Lipi dataset usable in machine learning, deep learning or computer vision based works, the images were resized into 128 X 128 pixels first and then convert from RGB to grayscale.[1]

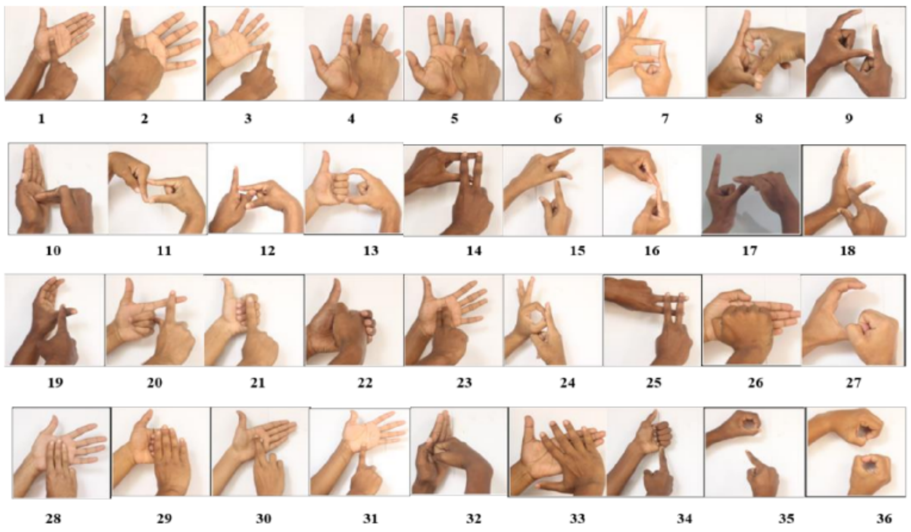


Figure 4: Resized Images (128 X 128)

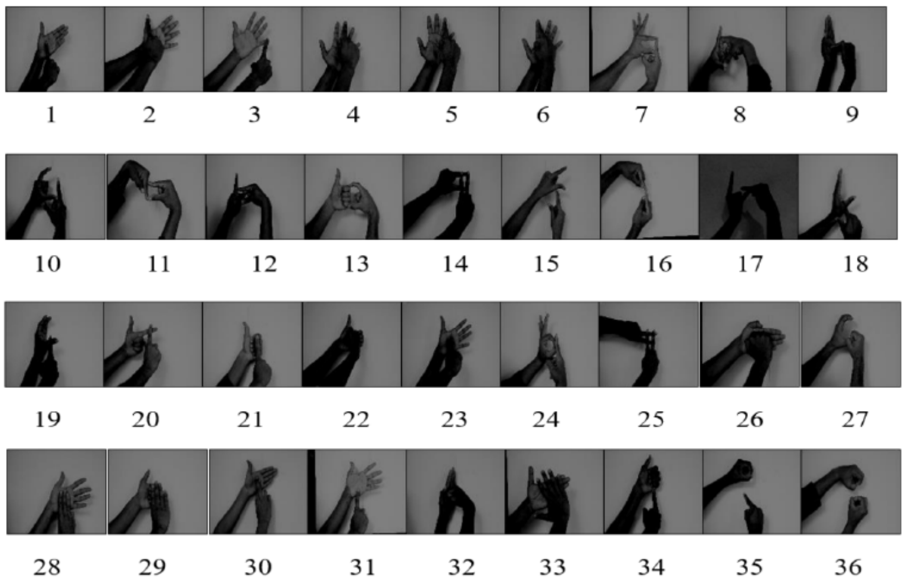


Figure 5: Final Gray Scaled Images

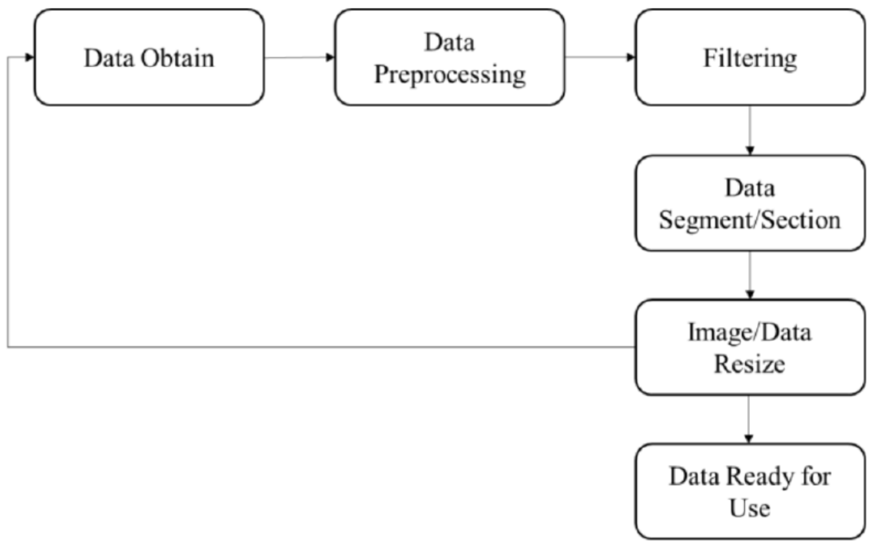


Figure 6: Steps of Dataset Generation - Ishara Lipi

2.2.2 Classification Model and Performance

- **Model Construction –** After constructing the dataset, the image samples were fitted into a 9-layer CNN classification model for training. Adam Optimizer was used for optimization, with a learning rate of 0.001%.
- **Performance –** For Ishara-Lipi sign character database, through data augmentation 15% data kept for testing and 8% data kept for training. For Ishara-Lipi sign character database, after 50 epochs, the model gets 92.65% accuracy on the training set and 94.74% accuracy on the validation set. [1]

2.3 Why Depth Information?

Sign Language Recognition was first introduced using the RGB images, which yielded a moderate accuracy then. But later, several researches on gesture recognition have stated that using depth information gives an extra advantage (information about z-coordinate) to the model for classification, hence yields a better accuracy. Back in 2011, in their research in comparison between hand gesture recognition depth and RGB images, Paul Doliotis et. al [3] has shown that using depth information gives a much better performance than RGB. They used

Microsoft Kinect depth sensing camera, which at that time was the latest technology.

In their experiment, they used a variety of data – one was easy data, with static hand gestures having a stable background, another was hard data, where there were static hand gestures with people moving in the background. They experimented using both depth data (using Kinect) and simple RGB data. The Results they got were as follows –

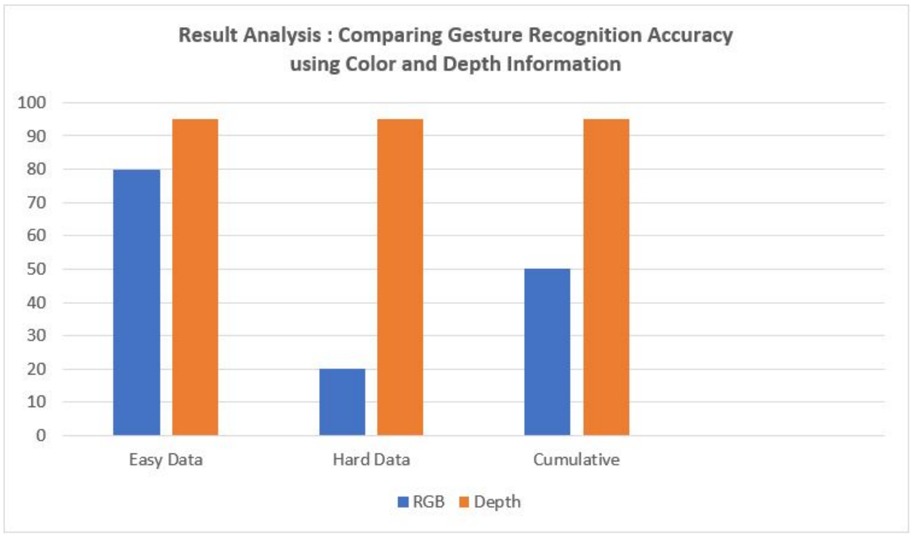


Figure 7: Comparing Gesture Recognition Accuracy using Color and Depth Information [3]

There are several other researches that delineates how the performance gets improved by using depth information. Some of them are mentioned here –

- Bo Liao et. al. [4] has shown how the performance in hand gesture recognition gets better with Generalized Hough Transform and Double Channel CNN Using RealSense. Recognition system maps depth images to color images based on generalized Hough transform in order to segment hand from a complex background in color images using the depth information. Then, it recognizes different hand gestures by a novel double-channel convolutional neural network containing two input channels which are color images and depth images. Moreover, they built a hand gesture database of 24 different kinds of hand gestures representing 24 letters in the English alphabet, containing a total of 168,000 images which are 84,000 RGB images and 84,000 depth images. Experimental results on the newly collected hand gesture database demonstrate the effectiveness of the proposed approach, and the recognition accuracy is 99.4%.

- Molchanov et al. [5] applied 3D convolution neural networks to recognizing drivers hand gestures from challenging depth and intensity data, which achieved a classification rate of 77.5% on the VIVA challenge database.

- Pugeault et al. [6] proposed a hand-shape recognition system that used Microsoft Kinect to collect appearance and depth images and OpenNI+NITE framework for detecting and tracking the hand, in which hand shapes corresponding to the alphabet letters were characterized using appearance and depth images, and then classified using random forests. The method reached 75.0% accuracy on the American Sign Language (ASL) database.

- Otiniano et al. [7] used hand gesture recognition to aid in sign language and finger spelling. Firstly, the hand area was segmented from background using depth map and precise hand shapes were extracted using both depth data and color data. Then, the gradient kernel descriptor was extracted from depth images and the RGB image content was described using Scale-Invariant Feature Transformation (SIFT). Finally, these features were used as the input of a support vector machine (SVM) classifier to recognize hand gestures, and the classification accuracy on the ASL database reached 90.2%.

- Nai et al. [8] proposed to use a set of depth features extracted from pixels on randomly positioned line segments in depth images for static hand gesture recognition. The random forest was used to combine these features to discover high-level unseen informative structure in an infinite dimensional feature space. The recognition accuracy reached 78% in ASL database and the speed is much faster than other methods.

- Liang et al. [9] proposed a novel distance-adaptive feature selection method to generate more discriminative depth-context features for hand parsing and the experimental result showed it produced 17.2% higher accuracy on the synthesized database for single-frame parsing.

2.4 Devices used for Depth Information

2.4.1 Leap Motion Controller

The Leap Motion controller is a sensor device that aims to translate hand movements into computer commands. It detects and tracks hands, fingers and finger-like objects reporting discrete position and motion.The LMC uses two high precision infrared cameras and three LEDs to capture hand information within its active range. As the LMC tracks hands and fingers in its field of view, it provides updates as a set, or frames of data. Each frame contains a list of the basic tracking data that describes the overall motion in the scene. [10] To date, it has been used primarily in conjunction with apps developed specifically for the controller. As of September 2013 there were 95 apps available through the Leap Motion app site, called Airspace. [11]

The main strength of the Leap Motion controller is the accurate level of detail provided by the Leap Motion API. The API provides access to detection data through a direct mapping to hands and fingers. Data provided by the API is determinate in that a client application does not need to interpret raw detection data to locate hands and fingers. [10] The granularity of the Leap Motion controller is an asset towards SLR. The controller can consistently recognise individual digits in a hand. Being able to identify, address and measure digit and fingertip location and movement is critical for accurately tracking of sign language motions. The controller is also capable of tracking very small movements, another essential capacity for accurate sign language recognition. [10]

The main drawback of Leap Motion controller is, it has difficulty maintaining the accuracy and fidelity of detection when the hands do not have direct line of sight with the controller. [10] This limitation makes it extremely hard to make an accurate recognition of a sign.

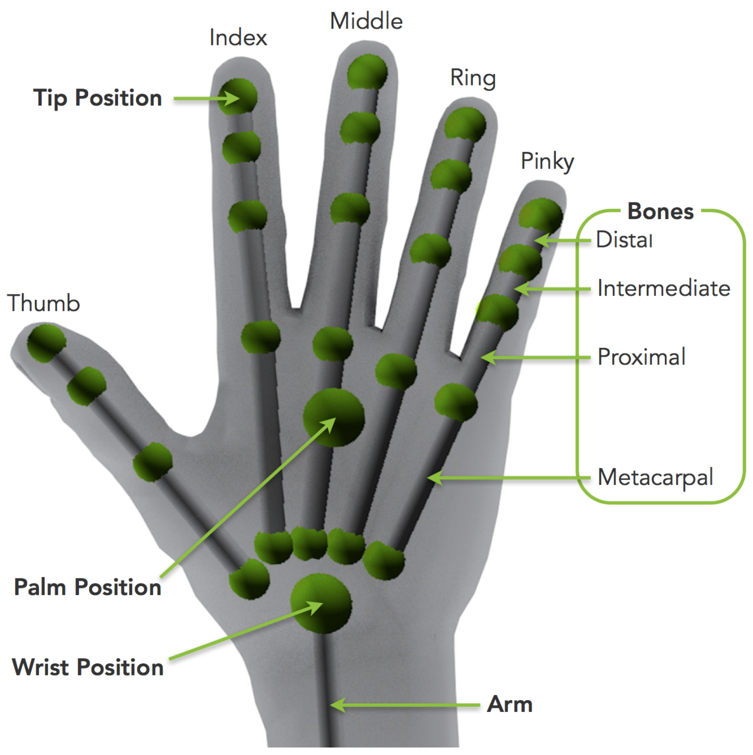


Figure 8: Joints tracked by LMC API

Leap motion API include different features for hand, fingers, bones and gesture. Some of them as follow -

1. Palm Features – Stabilized palm position, Palm normal, Palm direction
2. Finger Features – Finger direction, Metacarpal start position, Metacarpal end position, Proximal end position, Intermediate end position, Distal end position

2.4.2 Microsoft Kinect

Paul Doliotis et. al [3] used Microsoft Kinect in 2011, which yielded a stable 95% accuracy, even in the worst case. Since then, many of sign language recognition researches have been conducted using the Kinect depth sensing camera. In fact, in the early days of Sign Language Recognition (SLR) using depth sensors, the most popular depth sensing device was the Kinect. It was developed by Microsoft and PrimeSense and was released on November 2010. Kinect combines an RGB camera, a depth sensor and a multi array microphone. The best feature of the Kinect camera is its depth sensor, which uses an infrared projector and a CMOS sensor and is capable of tracking users in 3D independent of the lighting condition. There are three software frameworks available for Kinect – Microsoft SDK, OpenNI and OpenKinect. Although these frameworks can be used to do skeleton tracking, they do not support tracking of individual fingers and thus hand shape cannot be recognized using these frameworks, which is the main drawback of Kinect over RealSense, the latest depth sensing technology. In hand gesture recognition using Kinect, the hand of user is detected using the coordinates of palms, whereas in RealSense, all five fingers along with palm points (22 key joints in total) can individually be detected used for gesture recognition - yielding a better performance than Kinect.

2.4.3 Intel RealSense

The development of SLR is much promoted by two recent advances – one is the prosperity of Deep Learning and Deep Neural Network, and the other is Intel RealSense technology. RealSense features an RGB camera and a depth sensor. It supports hands and fingers tracking, and is recently being very popular in gesture recognition pose estimation, as this device is second to none in skeleton tracking and object detection using 3D coordinates by its depth sensor. With a Real-Sense, we can capture the 3D positions of fingertips and palm locations and can acquire better performance on skeleton tracking, gesture recognition, pose estimation etc. More specifically, Real-Sense is introduced to acquire 3D position of 22 key points of each hand of the user. For our research, we will be using the D435 model. These cameras primarily differ in the field of view (FOV) expressed in angles and the type of shutter that adjusts the exposure. RealSense D435 has a wider field of view, (HxVxD -– Horizontal x Vertical x Depth) : 91x 65x100 that minimizes blind, black spots in the depth image, after which a depth image pleasant for the eye is obtained.

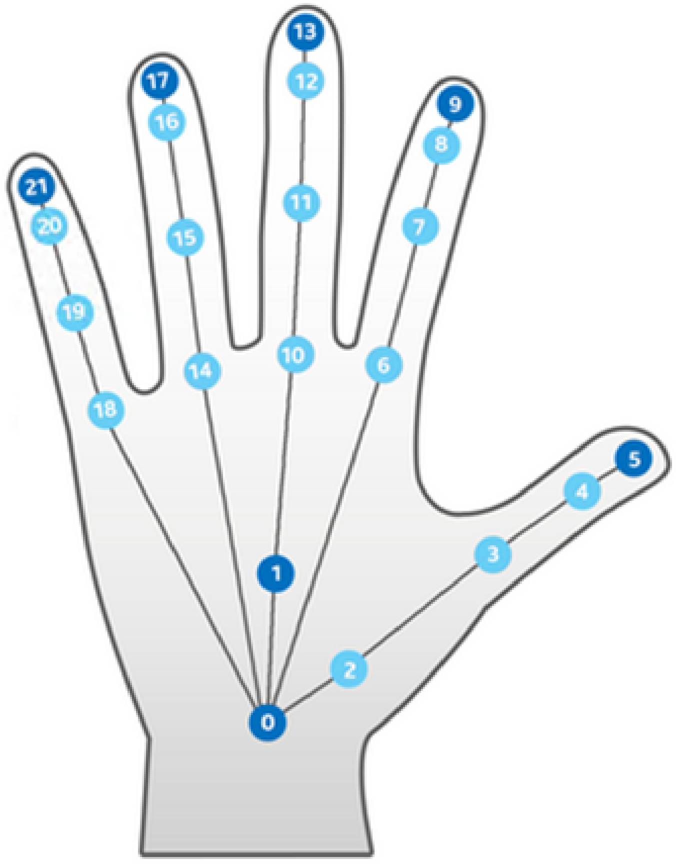


Figure 9: Joints tracked and numbered by Intel RealSense

2.5 Frameworks used for Depth Dataset Generation

2.5.1 MediaPipe

MediaPipe is a framework for building multimodal (e.g. video, audio, any time series data), cross platform (i.e Android, iOS, web, edge devices) applied ML pipelines. With MediaPipe, a perception pipeline can be built as a graph of modular components, including, for instance, inference models (e.g., TensorFlow, TFLite) and media processing functions. Cutting edge ML models using MediaPipe includes –

- Face Detection
- Multi-hand Tracking
- Hair Segmentation
- Object Detection and Tracking
- Objectron: 3D Object Detection and Tracking
- AutoFlip: Automatic video cropping pipeline

The module we will be using is MediaPipe Hands module, that consists of two models – Palm Detection Model and Hand Landmark MOdel. MediaPipe Hands is a high-fidelity hand and finger tracking solution. It employs machine learning (ML) to infer 21 3D landmarks of a hand from just a single frame. Palm detection model operates on the full image and returns an oriented hand bounding box. Hand landmark model operates on the cropped image region defined by the palm detector and returns high-fidelity 3D hand key-points.

- Palm Detection Model – To detect initial hand locations, a single-shot detector model was designed, optimized for mobile real-time uses. While building the model, at first, a palm detector was trained instead of a hand detector. In addition, as palms are smaller objects, the non-maximum suppression algorithm works well even for two-hand self-occlusion cases. Lastly, the focal loss was minimized during training to support a large amount of anchors resulting from the high scale variance.
- Hand Landmark Model – After the palm detection over the whole image our subsequent hand landmark model performs precise keypoint localization of 21 3D hand-knuckle coordinates inside the detected hand regions via regression, that is direct coordinate prediction. The model learns a consistent internal hand pose representation and is robust even to partially visible hands and self-occlusions. To obtain ground truth data, 30K real-world images were manually annotated with 21 3D coordinates, as shown below (Z-value

was taken from image depth map, if it exists per corresponding coordinate). To better cover the possible hand poses and provide additional supervision on the nature of hand geometry, a high-quality synthetic hand model was also rendered over various backgrounds and mapped to the corresponding 3D coordinates.

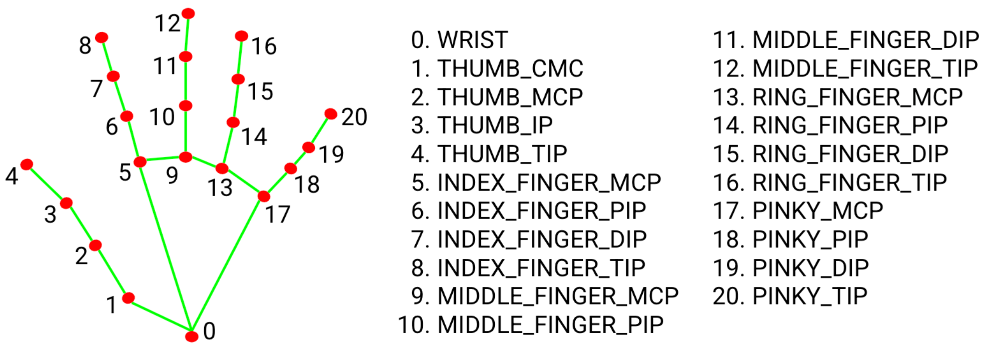


Figure 10: Hand Landmarks – 21 knuckle joints

2.6 Reference Paper – Indian Sign Language Numbers Recognition using Intel RealSense Camera

This is a Masters Thesis presented to the Faculty of California Polytechnic State University by Sravani Mudduluru.In this study, the author has used Intel RealSense to extract the joint information of 22 joint labels per single hand. [12] The Sign Language dataset was built using depth information and was later fed into several pretrained models to classify the Indian Sign Language(ISL) numbers from 0 to 9. Analyzing the results for different classifiers, the author has stated that multi-class SVM model showed higher accuracy of 93.5% when compared to the decision tree and KNN models.[12] In this thesis, Hands Viewer API of RealSense has been used that consists of 3 modes –

1. Cursor mode
    - High accuracy but limited to only few gestures
2. Extremity mode

- Tracks and shows the entire hand boundary along with the fingers
- Does not provide details about the joints on the fingers

3. Full hand mode

    - Provides the 3D skeleton of the hand including 22 joints information
    - Supports many gestures when compared with other modes

2.6.1 ISL Numbers Recognition using Intel RealSense – Feature Extraction Functions and Feature Vectors

In this paper, hand module is used to extract joint information. The module uses following functions to extract the rotation and position data of all joints that are detected per hand –

- enableTrackedJoints – otherwise no joints can be tracked
- isCalibrated – triggers the hand-joint tracking process
- HasTrackedJoints – boolean function (existence of joints)
- QuerytrackedJoints – information about hand-joints

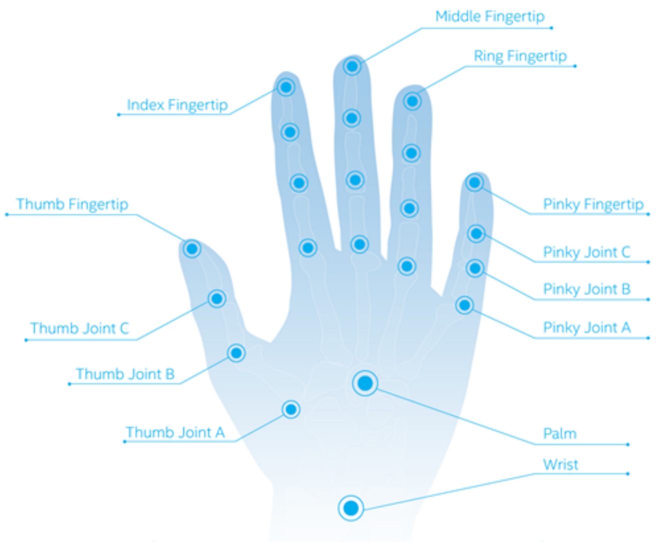


Figure 11: 22 Hand Joint Labels

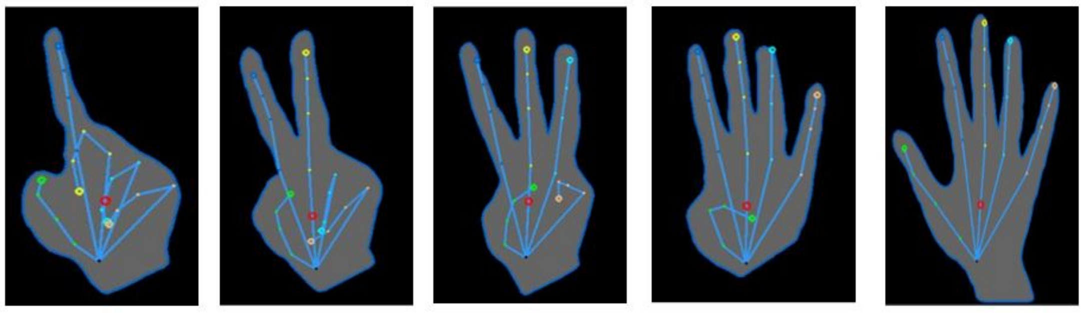


Figure 12: Feature Extraction from Depth Dataset

For training, 121 features have been extracted per sign per user in the feature vector. Following are some of feature vectors considered for this research –

- Openness Property – This will range from 0 (all fingers completely closed) to 100 (All fingers wide spread).
- Finger Data – foldedness value and radius of the finger
- Position World – Geometric position of a joint in world coordinates
- Position Image – Geometric position of a joint in depth coordinates

2.6.2 ISL Numbers Recognition using Intel RealSense – Classification Model and Outcome

For training purposes, multi-class SVM was chosen as the supervised learning model. As mentioned, 121 features have been extracted from each depth frame. But for training purpose, the features considered were – radius of finger, finger foldedness value and global and local positions of the joints in the x, y, and z coordinates. Feature vectors that did not contribute to the classification process had been removed. [12]

This study generated a significant amount of analyzations to understand the better performance of the model as well as the camera. They tried to make different splits of the training samples in order to analyze which percentage of training split showed better performance. They also 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.

It was 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, the accuracy found when all the features considered was 93.82% on average.

2.6.3 Other Sign Languages using Depth Information

1. Arabic Sign Language Recognition using Leap Motion Controller and Microsoft Kinect v2 – Miada ei. al. [13] tried develop a supervised machine learning hand-gesturing model to recognize Arabic Sign Language (ArSL), using two sensors: Microsoft's Kinect with a Leap Motion Controller. Their proposed model relies on the concept of supervised learning to predict a hand pose from two depth images and defines a classifier algorithm to dynamically transform gestural interactions based on 3D positions of a hand-joint direction into their corresponding letters whereby live gesturing can be then compared and letters displayed in real time. The results indicated that using both devices for the ArSL model were essential in detecting and recognizing 22 of the 28 Arabic alphabet correctly 100 %. [13] Some of the results were incorrect, such as those for the letter 'ha' which the device cannot get correctly. It could be because many fingers need to cross over one another. So, some points were not detected because they were not in sight of the controller, which is a major drawback of Leap Motion Controller.

2. Sign Language Recognition Using Kinect in German Sign Languages – Simon et. al. [14] has presented and tested an open source framework for general gesture recognition with isolated 25 signs of German sign language using Microsoft Kinect. Recognition is done using Hidden Markov models with a continuous observation density. The framework also offers an easy way of initializing and training new gestures or signs by performing them several times in front of the camera. First results with a recognition rate of 97% show that depth cameras are well-suited for sign language recognition.

3. Sign Language recognition using Real Sense – In 2015, Huang et. al. proposed a methodology of alphabet sign language recognition, using Real-Sense to detect and track human hands and fingers, followed by recognizing the alphabet sign using a Deep Neural Network. [15] They claimed that their purposed method can greatly improve the performance on recognition tasks. They used Real-Sense to acquire 3D position of 22 key points of hand. Then the 3D co-ordinates of those points were fed into DNN directly instead of extracting and selecting features manually like – finger length, finger width, angle of finger. Also, to demonstrate the effectiveness of Real-Sense, they built a Kinect-DNN system for comparison which uses Kinect as input device. In contrast to RealSense, Kinect only supports the tacking of palm points. Two datasets were constructed, each containing 26 alphabets signs - one collected by Real-Sense, other one by Kinect. For each dataset, 26×3 video clips were recorded from 3 different signers across these alphabet signs. Then, Huang et. al. extracted total 65,000 frame images from those clips and used for training and testing. Analyzing the results, they found that, the average recognition rate of DNN based on Real-Sense is 98.9% , while that of DNN based on Kinect is 97.8% – which shows how RealSense improves the performance in Sign Language Recognition.

4. An Approach to Sign Language Translation using the Intel RealSense Camera – This proposed approach is for Recognizing American Sign Language (ASL) by RealSense. Inden et. al. [16] compared the two frequently used classification model - SVM and ANN for recognizing the alphabets of ASL, using RealSense. The dataset of 26000 images from 10 users was built by the normalized values of x, y, and z co-ordinates of the hand joint-points extracted by Intel RealSense. In result analysis, they found SVM yielding a recognition rate of 95%, whereas an Artificial Neural Network model giving an accuracy of 92.1%.

5. Hand Gesture Recognition with Generalized Hough Transform and DC-CNN Using Realsense – In 2018,[4] Gaoxiang et. al proposed a hand gesture recognition system based on the data captured by Intel RealSense Front-Facing Camera SR300. Considering that the pixels in depth images collected by RealSense are not one-to-one to those in color images, the recognition system maps depth images to color images based on generalized Hough transform in order to segment hand from a complex background in color images using the depth information. Then, it recognizes different hand gestures by a novel double-channel convolutional neural network containing two input channels which are color images and depth images. Moreover, they built a hand gesture database of 24 different kinds of hand gestures representing 24 letters in the English alphabet. It contains a total of 168,000 images which are 84,000 RGB images and 84,000 depth images. Experimental results on our newly collected hand gesture database demonstrate the effectiveness of the proposed approach, and the recognition accuracy is 99.4%.

3 Proposed Approach

3.1 Proposed Methodology of Generating the Dataset

From the background study we have done about using depth information in Sign Language Recognition and using MediaPipe for dataset generation, this is fundamental that the device detects the hand joint-points from captured image samples and gives the 3D (x, y, z) co-ordinates of the detected hand joint-points. Our dataset will be generated based on the value of those joint-points. Our proposed approach involves following steps –

1. RGB Real-time Streaming - The next step is to do the real-time Streaming in RGB mode using general purpose camera. As we intend to generate the dataset from real-time hand-sign display, streaming in RGB mode is our first and foremost task. The recommended settings for streaming is 30 frames per second and an aspect ratio of 4:3.

2. Capturing Input Image Frames - While streaming in RGB mode, the next task is to capture frames from the stream. This is necessary because in real-time recognition, the device might not be able to track the hands and the hand joint-points instantly, or, may detect a wrong point which causes faulty detection. This may occur in some of the first frames, as well as in some of the last frames – so it is necessary to discard those images from the dataset. Also, there can be faulty or no hand detection in certain lighting condition, background and for some specific angles of a sign, where occlusion is higher. And more importantly, the values of the 3D co-ordinates of the corresponding frames must also be discarded, as our work is focused on these 3D co-ordinate values.

3. Detect Hand Joint-points from image samples using MediaPipe – the next step is Detection of hand joint-points. The image frames are to be fed into MediaPipe framework, where it goes under hand key-points detection using Palm Detection Model Hand Landmark Model. This process involves –

- Detect the hand of the user from image frames
- Set the region of interest (ROI) using Palm Detection Model
- Detect the hand joint-points of the hand using Hand Landmark Model
- get the normalized values of x and y co-ordinates for each point
- get the estimated z-value for each point, generated from the depth map of the RGB input image sample
- store the x, y, z values for each frame

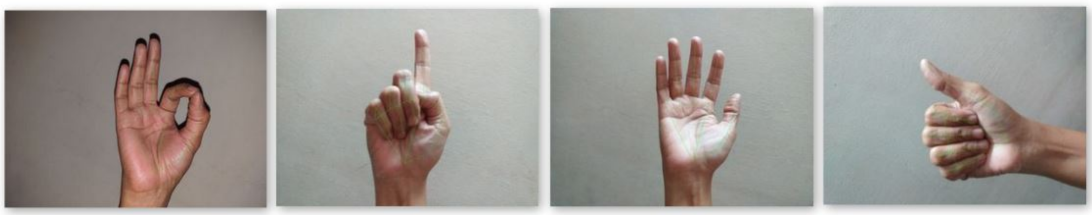
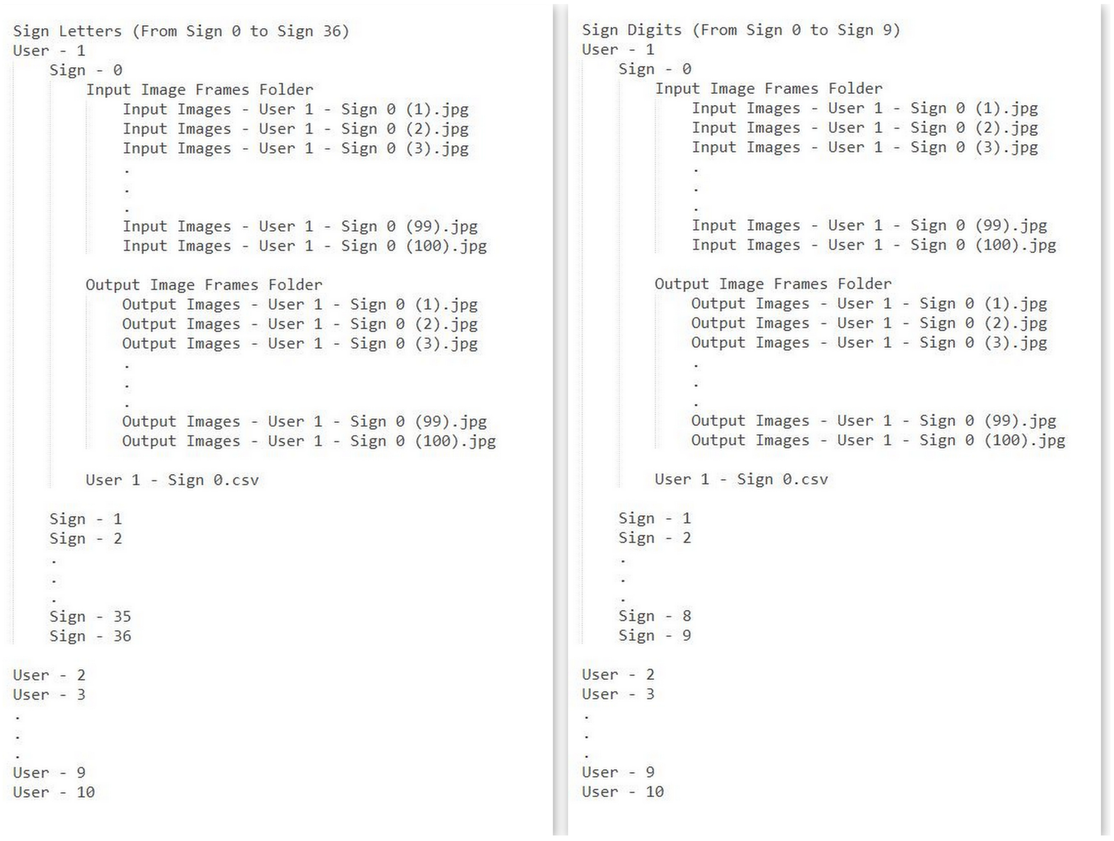


Figure 13: Detection of 21 Hand joint-points from input image sample

4. Building the Final Dataset – In this step , we have extracted the 3D information of the 21 key points of our hand. So, the final task is to generate the dataset from the information we have. As 3D co-ordinate values of 21 joint-points result in 63 values for each frame - these 63 values are the core of our dataset. We store these 63 values one in each column, like this – x1, y1, z1, x2, y2, z2, ..., x22, y22, z22. This is the information for one frame. Likewise, we can iterate this process the number of times we intend to have a frame for each sign per user. In our case, the number is 100. We intend to store these values into a .csv file – one for each sign of each user. These .csv files will collectively build our dataset with depth information.

3.1.1 Structure of the Proposed Dataset

Construction of a dataset is not be complete unless it is in a structured way. This dataset will later be used in Recognition of Bangla Sign Languages using depth information – if it is not in a structured way, the entire dataset is nothing but a storage of some garbage values. So, as a part of our proposal in building BSL depth dataset, we would like to propose the structure of our dataset as well.



```
Sign Letters (From Sign 0 to Sign 36)
User - 1
    Sign - 0
        Input Image Frames Folder
            Input Images - User 1 - Sign 0 (1).jpg
            Input Images - User 1 - Sign 0 (2).jpg
            Input Images - User 1 - Sign 0 (3).jpg
            .
            .
            .
            Input Images - User 1 - Sign 0 (99).jpg
            Input Images - User 1 - Sign 0 (100).jpg

        Output Image Frames Folder
            Output Images - User 1 - Sign 0 (1).jpg
            Output Images - User 1 - Sign 0 (2).jpg
            Output Images - User 1 - Sign 0 (3).jpg
            .
            .
            .
            Output Images - User 1 - Sign 0 (99).jpg
            Output Images - User 1 - Sign 0 (100).jpg

        User 1 - Sign 0.csv

    Sign - 1
    Sign - 2
    .
    .
    .
    Sign - 35
    Sign - 36

User - 2
User - 3
.
.
.
User - 9
User - 10
```

```
Sign Digits (From Sign 0 to Sign 9)
User - 1
    Sign - 0
        Input Image Frames Folder
            Input Images - User 1 - Sign 0 (1).jpg
            Input Images - User 1 - Sign 0 (2).jpg
            Input Images - User 1 - Sign 0 (3).jpg
            .
            .
            .
            Input Images - User 1 - Sign 0 (99).jpg
            Input Images - User 1 - Sign 0 (100).jpg

        Output Image Frames Folder
            Output Images - User 1 - Sign 0 (1).jpg
            Output Images - User 1 - Sign 0 (2).jpg
            Output Images - User 1 - Sign 0 (3).jpg
            .
            .
            .
            Output Images - User 1 - Sign 0 (99).jpg
            Output Images - User 1 - Sign 0 (100).jpg

        User 1 - Sign 0.csv

    Sign - 1
    Sign - 2
    .
    .
    .
    Sign - 8
    Sign - 9

User - 2
User - 3
.
.
.
User - 9
User - 10
```

Figure 14: Proposed Structure for the BSL Dataset

There are 37 static signs for alphabets (using single hand) and 10 signs for numbers (0 – 9), hence in total 47 signs in Bangla Sign Language. Initially, we intend to collect data from 5 users. For each user, we intend to have a collection of 100 RGB frames per sign. Considering the fact that, we will be doing this in RGB Streaming in 30 fps, each user needs to perform each of the hand signs in front of the camera for 10 seconds. Though it might seem that we are taking too many extra frames, but we have considered the matter that in some cases, we may have to discard the many frames from first and last, if they cannot detect the hand joint-points correctly. In terms of the size of our dataset, 47 signs performed by 10 users, each generating 100 RGB frames results in – 47 X 10 X 100 = 47000 RGB Images in total. The information about the 3D co-ordinates extracted will also be stored in a structured way –

- .csv is the file where we will store the normalized values of x, y, and estimated values of z co-ordinates, extracted from the input image samples. Each row will contain the information of a depth frame. As there are 21 joints, we will have 63 values for each frames in a sequence like this – x1, y1, z1, x2, y2, z2, ..., x22, y22, z22. Also, one column for storing the name or absolute path of the corresponding input image and another column for saving the label – hence, each .csv file will be of 65 columns.

4 Implementation Process

4.1 Image Sample Collection

The first step of implementing the process was to collect RGB images and generate dataset. We collected the RGB images from Real time streaming. For every digit and every letter of every user, more than 300 frames was captured from the stream. For every sign of each user, among those initially captured frames, we selected 100 suitable frames from which the .CSV file will be generated. Firstly, we eliminated the frames where user had to initially adjust his/her hand position into the specific hand pose for that particular sign. Secondly, last few blurry frames where the hand of the user was moving because of deviating away from the particular hand pose was also eliminated. Then we manually checked every frame that was left after the first two elimination. We eliminated those frames where faulty detection of hand joint-points occurred. Also, in some of the initial frames, the lighting ambience tends to be darker than usual, because this takes some time for the device to adjust to the given lighting ambience.

These manual checks and eliminations resulted in more than 150 frames for hand key-points detection using MediaPipe. This process was repeated for every sign from every user. After having a look on the output images where hand key-points detection took place, we had to manually go through each and every one of them, discarding if faulty detection was there, and finally keeping 100 samples where most accurate hand point detection took place. With those images for that sign, a .csv file was created which consists of the x, y and z co-ordinates of the 21 hand joint-points. Total 10 users participated and for every user, a total of 4 image frames has been collected to generate our final dataset.

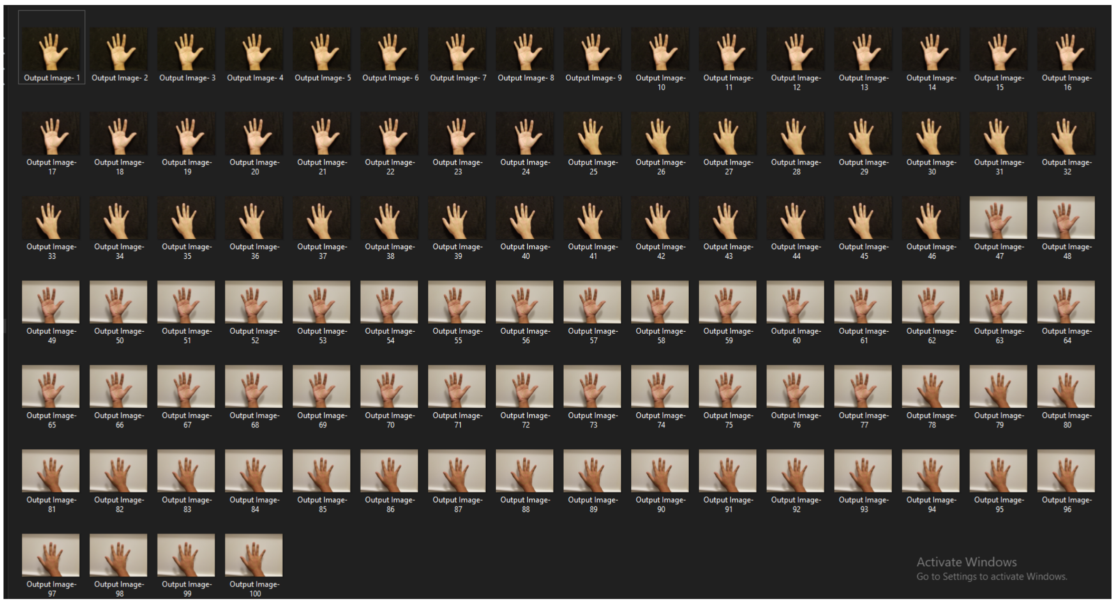


Figure 15: Selected 100 Frames for digit 5 in a specified Folder

4.2 Variation in Sample Collection

While collecting the samples, we have tried to consider variations as much as possible to make the dataset versatile and more challenging for classification. All the variations we considered can be categorised into two categories –

- User Variation
- Environment Variation

4.2.1 User Variation

A total 10 users participated for the generation of our dataset. They were different from each other. The variation occurred in terms of age, gender, size of hand and skin color. Each user was different from every other user in term of at least one of the factors mentioned above. Among the users, 5 of them were female users and 5 of them were male users. 3 users were below age 20, 3 users were between the age 20 and 50 and 4 users were above the age 50. Among the skin colors, our users were of light, tan, brown or dark brown skin color.

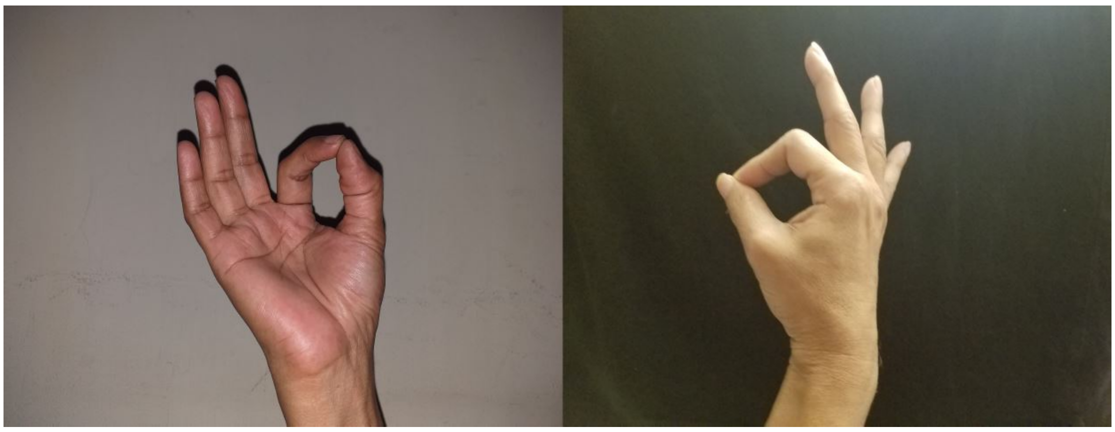


Figure 16: Variation of Hands Between a Male User(R) and a Female User(L)

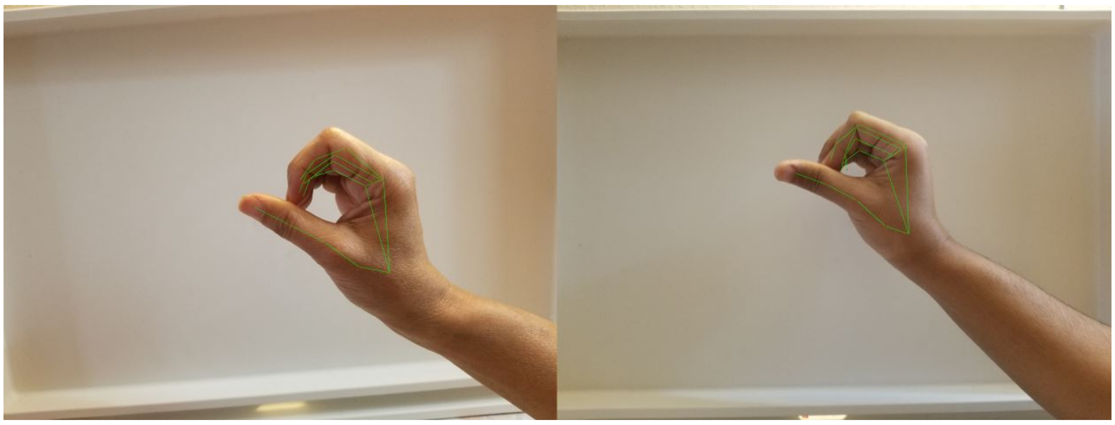


Figure 17: Variation of Hands Between Users of Different Ages; Age-62(L), Age-23(R)

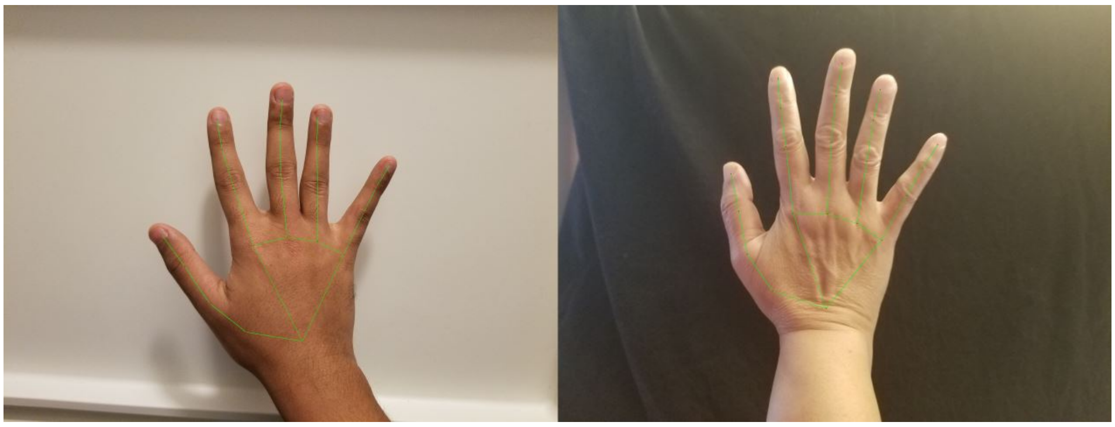


Figure 18: Variation of Hands Between Users of Different Skin colors

4.2.2 Environment Variation

While implementing the process, variations in environment was also maintained. The features of environment which were variables were :

- Angles : The angle between the lens of the camera and the front/back face of the hand.
- Luminance : The lighting of the environment in which the image frames were captured.
- Scaling : The distance between the camera and the user's hand.
- Background : The background behind the user's hand.
- Translation : The region of interest (hand of the user) in different position of the image sample – top right, top mid, bottom left, bottom right etc.
- Hand- Facing : The side of hand (front/ back) which is facing the camera.

Same environment setup for capturing one user's every image frame for one sign may create redundant data as the hand joint-point co-ordinates will not change much in same environment setup. Redundant data will cause over-fitting when training a model, which means it will perform very well for our labeled dataset but the accuracy will drop very drastically when classifying new and unknown data.

The aforementioned features were changed for every sign of every users to ensure our dataset does not become redundant can train our model well enough so that, it can classify accurately in different kinds of environment. Different environment setup will create variations in our dataset as it will result in great difference in positions of the hand joint-points.

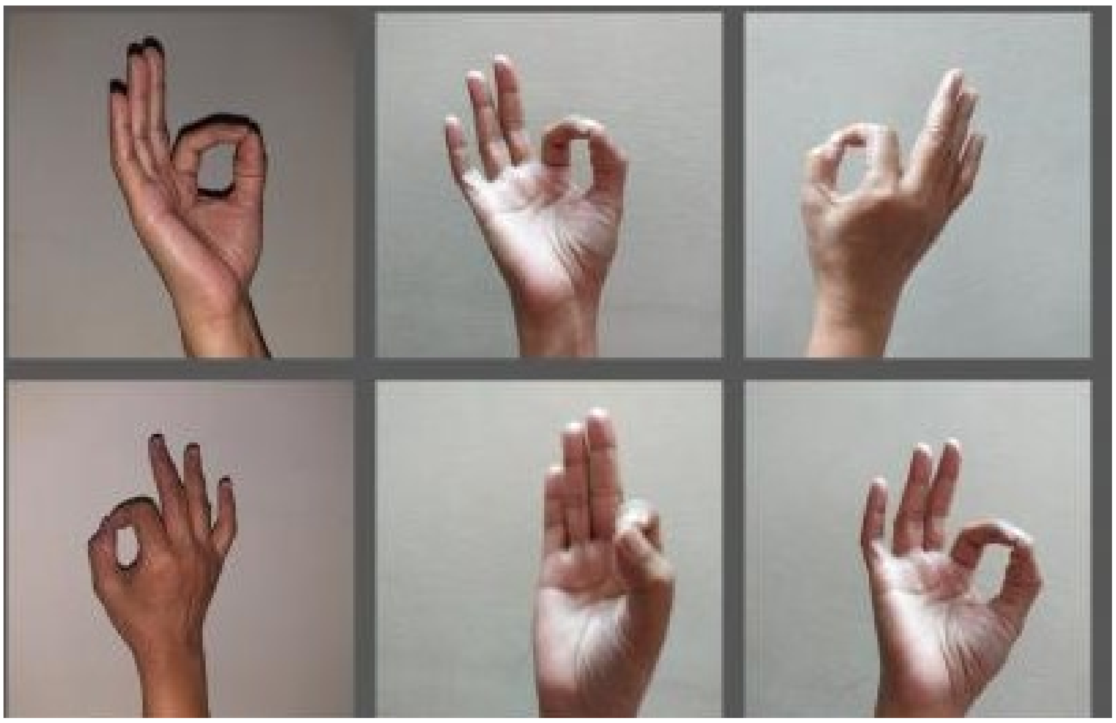


Figure 19: Variation of Angle Between the Lens of the Camera and the Front/Back Face of the Hand

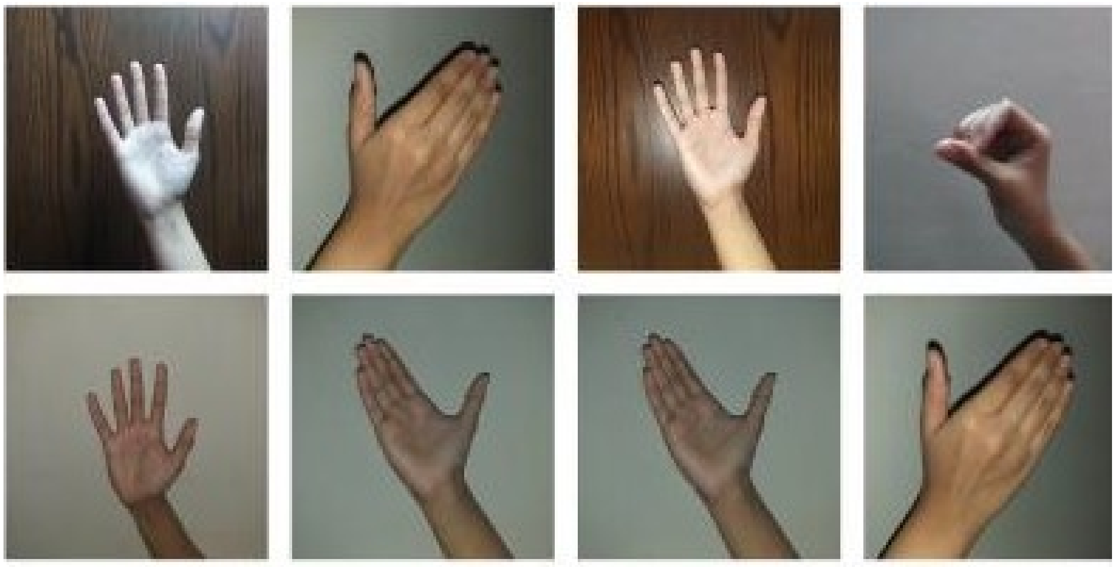


Figure 20: Variation of the Lighting of the Environment

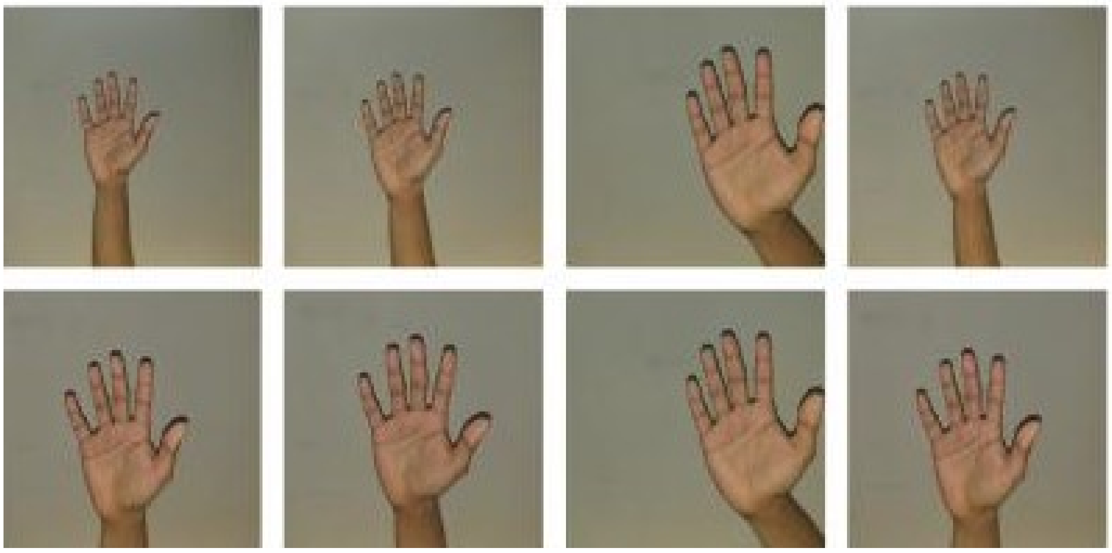


Figure 21: Variation of Distance Between the Camera and the User-Hand

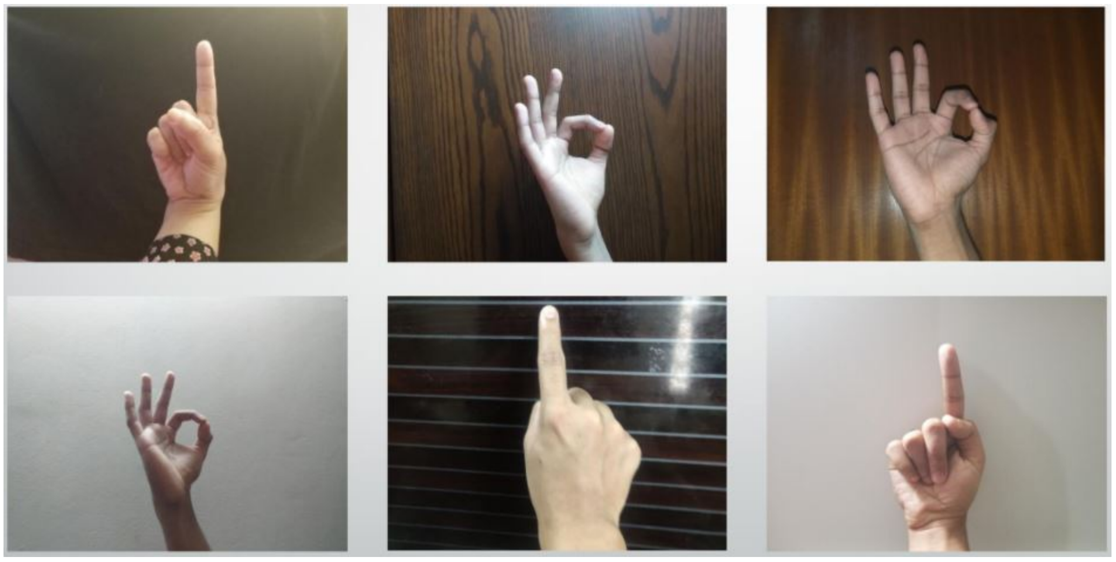


Figure 22: Variation of the Background Behind the User-Hand

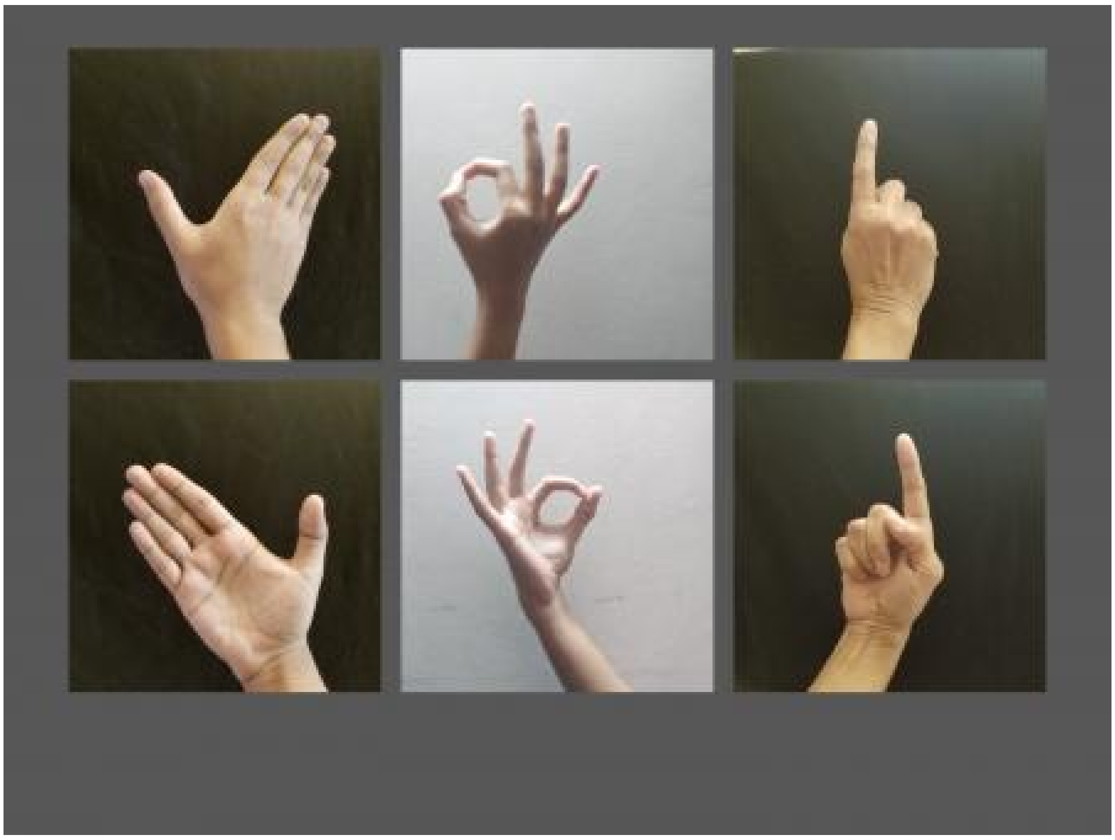


Figure 23: Variation of the Side of Hand (Front/ Back) which is Facing the Camera

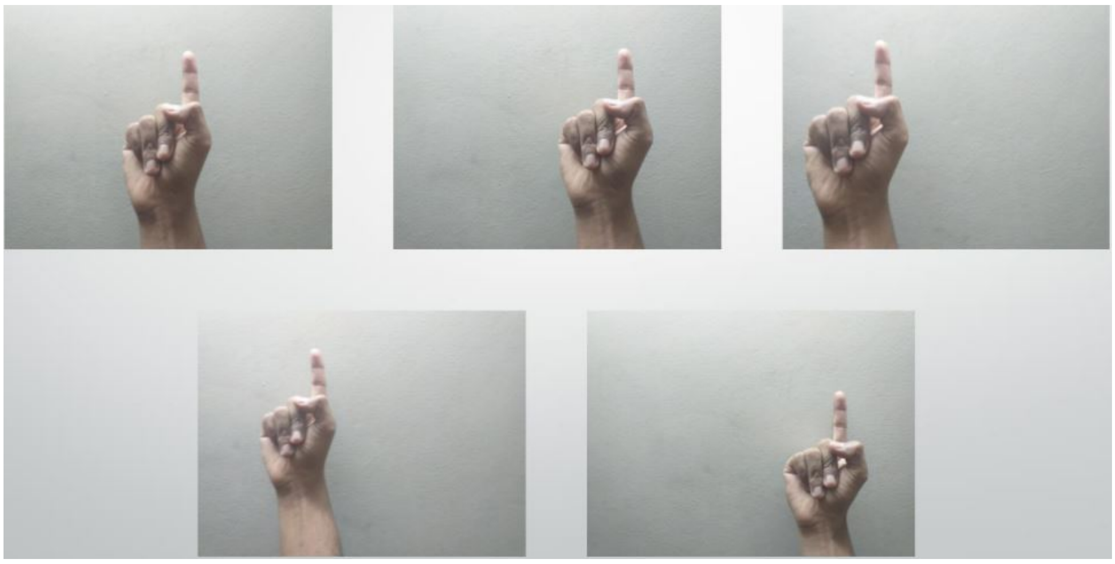


Figure 24: The region of interest (hand of the user) in different position of the image sample – top right, top mid, bottom left, bottom right

4.3 CSV File Generation

For hand joint-point detection, we used **MediaPipe**. It is a cross-platform framework for building multimodal applied machine learning pipelines. For our purpose, we have used **OpenCV** .After capturing the RGB images, we fed our image frames into MediaPipe. MediaPipe detected user-hand, hand joint-points and saved the normalized x,y and estimated z co-ordinates of each key point. It also saved the output images with detected hand points.

In one image frame, there 21 hand joint-points and each point has 3 co-ordinates. For each image frame, there were 63 co-ordinates and there were 100 image frames per sign for each user. MediaPipe saved 63 co-ordinates of 100 image frames of the same sign for each user in one .CSV file. So one .CSV file consists of 65 columns and 101 rows (with labeling) and one .CSV file represents one sign's 100 frame's co-ordinates from one user.

5 Classification

After constructing the dataset, the next step is classification. While doing the classification, we have considered the following factors –

- Sign Digits Sign Letters have been classified separately – Because some of the digit-signs are identical to some of the letter-signs
- Sign Digit Classification has 10 labels, Letter Classification has 37 labels
- 10 fold cross validation has been used for classification
- Training-Testing Ratio was – 0.80 : 0.20
- Values of the coordinates (from x0, y0, z0 to x20, y20, z20) were fed as input features, hence $21 \times 3 = 63$ input features

5.1 Classification in Sign Digits

- Sign Digits Dataset has 10000 image samples and 10 labels
- Classifiers used for classification – SVM, KNN, RFC, DTC, XGB, ANN.
- ANN model structure for classifying Sign Digits –
  - Input Layer – 63 nodes
  - Hidden Layer – 64 nodes
  - Hidden Layer – 128 nodes
  - Dropout Layer (rate = 0.3)
  - Hidden Layer – 64 nodes
  - Dropout Layer (rate = 0.3)
  - Hidden Layer – 64 nodes
  - Output Layer – 10 nodes
- ANN model hyper-parameters –

  - Optimizer – Adam
  - Loss Function – Sparse categorical cross-entropy
  - Activation Function – ReLu function
  - Output Activation Function – Softmax function

5.2 Classification in Sign Letters

- Sign Digits Dataset has 37000 image samples and 37 labels
- Classifiers used for classification – SVM, KNN, RFC, DTC, XGB, ANN.
- ANN model structure for classifying Sign Letters –
  - Input Layer – 63 nodes
  - Hidden Layer – 64 nodes
  - Hidden Layer – 128 nodes
  - Hidden Layer – 128 nodes
  - Dropout Layer (rate = 0.3)
  - Hidden Layer – 74 nodes
  - Dropout Layer (rate = 0.3)
  - Output Layer – 37 nodes
- ANN model hyper-parameters –
  - Optimizer – Adam
  - Loss Function – Sparse categorical cross-entropy
  - Activation Function – ReLu function
  - Output Activation Function – Softmax function

5.3 Accuracy in Sign Digits

We have used different classifiers and in those classifiers, we have tried to evaluate the performance using different hyper-parameters in those classifiers –

No.	Classifier	Accuracy
1	KNN	92.39%
2	SVM (Linear Kernel)	98.31%
3	SVM (Polynomial Kernel, degree = 3)	93.56%
4	SVM (Polynomial Kernel, degree = 4)	86.63%
5	SVM (Polynomial Kernel, degree = 5)	75.75%
6	SVM (RBF Kernel)	98.63%
7	RFC	95.72%
8	DTC	95.67%
9	XGB	97.78%
10	ANN (Training Accuracy)	98.51%
11	ANN (Testing Accuracy)	90.23%

5.4 Accuracy in Sign Letters

For same classifiers, classification accuracy for those classifiers are as follows –

No.	Classifier	Accuracy
1	KNN	88.61%
3	SVM (Polynomial Kernel, degree = 3)	82.67%
4	SVM (Polynomial Kernel, degree = 4)	74.30%
5	SVM (Polynomial Kernel, degree = 5)	68.09%
6	SVM (RBF Kernel)	90.81%

No.	Classifier	Accuracy
7	RFC	91.58%
8	DTC	84.61%
9	XGB	90.96%
10	ANN (Training Accuracy)	96.81%
11	ANN (Testing Accuracy)	98.55%

6 Result Analysis

6.1 Classification Result Analysis – Sign Digits

- Most of the classifiers have yielded a decent accuracy – Taking more samples from more users with more signer and environment variation may decrease the accuracy
- Reasons for higher accuracy —
  - Lesser number of labels : 10-class classification problem
  - Lesser similarity between signs, coordinates for signs are different
- Maximum Accuracy using SVM —
  - This classifier operates well in case of large number of features
  - RBF kernel specialized to operate in higher-dimensional feature space
- Incorrect classification – Mostly between 4 & 5, 1 & 7, and 2 & 8. Because, in these signs, all other Coordinates are similar except thumb tip, thumb dip & thumb mcp.

6.2 Classification Result Analysis – Sign Letters

- Most of the classifiers have yielded a moderate accuracy, while some of them underperformed. Because of the variety in signs, the accuracy is lesser than that of sign-digits.

- Reasons for lower accuracy than sign-digits —
  - Higher number of labels : 37-class classification problem
  - Similarity between signs – coordinates for some signs are very similar
- Maximum Accuracy using ANN — the classifier is constructed for dealing with complex data with a combination of several hidden layers and dropout layers.
- SVM Linear classification model takes a lot of time in calculating and yielding an accuracy, because of the higher number of labels and samples in the dataset – it is very much difficult and time consuming to classify this much samples (37000 samples) linearly into this many labels (37 labels). Therefore, it is not feasible to use it in such a problem where the number of labels are very high.

6.3 Experimental Result Analysis Remarks

- Classifiers yields moderate to high accuracy because of taking all the 63 3D coordinates as input features. In that case, the probabilities to repetition in input was decrease, also taking samples from different users in different environments makes the dataset lesser-biased.
- SVM outperforms other well-known classifiers, because in higher-dimensional feature space, SVM is meant to operate in a better way using the kernels.
- In case of Sign-digits ANN classification, the model overfits as it yields a higher accuracy in training dataset than that of testing dataset. By training the dataset with more data and ensembling, we can deal with this overfitting.
- In case of Sign-Letters ANN classification, the model performs better in the testing dataset than the training set, therefore is underfitting. By increasing number of features, performing feature engineering and increasing the number of epochs or increasing the duration of training set, underfitting is likely to be mitigated.

7 Limitations and Challenges

- Faulty or No detection due to occlusion – Hand point detection is not satisfying from some of the angles for some of the signs
- In plain background (wall, black sheet or curtains) detection rate is higher more accurate, but in complex background (wooden texture), faulty or no detection is frequent
- In brighter lighting ambience, detection rate is higher more accurate than that of darker lighting ambience
- We haven't taken samples from any expert user; samples from expert user could have eased the detection yielded higher accuracy
- In Sign letters, some of the signs are very much similar (in case of covering all the angles) –

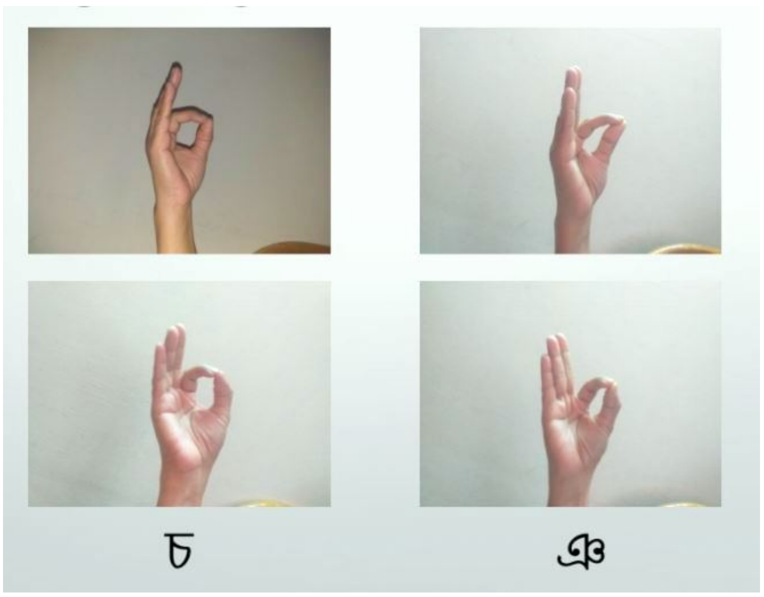


Figure 25: Signs very much similar to each-other

8 Conclusion and Future Work

To summarize, in future we can work on the betterment of the existing dataset by taking more samples from more users in different environmental setups. We can collect samples from more users and try to make as much variations as possible to make the dataset more versatile and challenging

We can develop the neural network model we have designed for this dataset in order to increase the accuracy, by changing the hyper-parameters.

We can consider other features instead of 3D-coordinates – From 3D-coordinates, we can determine the finger-foldedness, finger-height, angles between fingers which can be fed into training as input features. Later, we can train the dataset on different classification model and evaluate the performance, and compare that with the current depth dataset.

We can also start building Bangla Sign Language Depth Dataset using two-handed signs, as MediaPipe can also detect the two-handed samples. Moreover, this thesis work is focused on only static signs, we can build Bangla Sign Language Depth Dataset using dynamic signs.

In the process of building the dataset using depth information, we have extracted the x and y co-ordinate values for each joint-points. In Bangla Sign Language, there is no such work been done using the x y co-ordinates of hand joint-points. There is a scope to work on that as well, and to analyze whether or not the created dataset of RGB co-ordinates can outperform the existing BSL dataset of RGB images.

Finally, we can evaluate the performance of the individual datasets on different classifiers and compare the results.

References

[1] M. S. Islam, S. S. S. Mousumi, N. A. Jessan, A. S. A. Rabby, and S. A. Hossain, "Ishara-lipi: The first complete multipurposeopen access dataset of isolated characters for bangla sign language," in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pp. 1–4, IEEE, 2018.

[2] J. K. Chen, D. Sengupta, and R. R. Sundaram, "Cs229 project final report sign language gesture recognition with unsupervised feature learning."

[3] P. Doliotis, A. Stefan, C. McMurrough, D. Eckhard, and V. Athitsos, "Comparing gesture recognition accuracy using color and depth information," in *Proceedings of the 4th international conference on PErvasive technologies related to assistive environments*, pp. 1–7, 2011.

[4] B. Liao, J. Li, Z. Ju, and G. Ouyang, "Hand gesture recognition with generalized hough transform and dc-cnn using realsense," in *2018 Eighth International Conference on Information Science and Technology (ICIST)*, pp. 84–90, IEEE, 2018.

[5] P. Molchanov, S. Gupta, K. Kim, and J. Kautz, "Hand gesture recognition with 3d convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 1–7, 2015.

[6] N. Pugeault and R. Bowden, "Spelling it out: Real-time asl fingerspelling recognition," in *2011 IEEE International conference on computer vision workshops (ICCV workshops)*, pp. 1114–1119, IEEE, 2011.

[7] K. O. Rodriguez and G. C. Chavez, "Finger spelling recognition from rgb-d information using kernel descriptor," in *2013 XXVI Conference on Graphics, Patterns and Images*, pp. 1–7, IEEE, 2013.

[8] W. Nai, Y. Liu, D. Rempel, and Y. Wang, “Fast hand posture classification using depth features extracted from random line segments,” *Pattern Recognition*, vol. 65, pp. 1–10, 2017.

[9] H. Liang and J. Yuan, “Hand parsing and gesture recognition with a commodity depth camera,” in *Computer Vision and Machine Learning with RGB-D Sensors*, pp. 239–265, Springer, 2014.

[10] L. E. Potter, J. Araullo, and L. Carter, “The leap motion controller: a view on sign language,” in *Proceedings of the 25th Australian computer-human interaction conference: augmentation, application, innovation, collaboration*, pp. 175–178, 2013.

[11] D. Naglot and M. Kulkarni, “Real time sign language recognition using the leap motion controller,” in *2016 International Conference on Inventive Computation Technologies (ICICT)*, vol. 3, pp. 1–5, 2016.

[12] S. Mudduluru, “Indian sign language numbers recognition using intel realsense camera,” 2017.

[13] M. Mohandes, S. Aliyu, and M. Deriche, “Arabic sign language recognition using the leap motion controller,” in *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, pp. 960–965, 2014.

[14] S. Lang, M. Block, and R. Rojas, “Sign language recognition using kinect,” in *International Conference on Artificial Intelligence and Soft Computing*, pp. 394–402, Springer, 2012.

[15] J. Huang, W. Zhou, H. Li, and W. Li, “Sign language recognition using realsense,” in *2015 IEEE China Summit and International Conference on Signal and Information Processing (ChinaSIP)*, pp. 166–170, IEEE, 2015.

[16] J. Mistry and B. Inden, “An approach to sign language translation using the intel realsense camera,” in *2018 10th Computer Science and Electronic Engineering (CEEC)*, pp. 219–224, IEEE, 2018.