Real Time Indian Sign language Recognition

Paranjoy Paul

National Institute of Technology, Tiruchirappalli, Tamil Nadu - 620015

Dr. G N Rathna

Indian Institute of Science, Bangalore, Karnataka - 560012

Abstract

Sign languages are set of predefined languages which use visual-manual modality to convey an information. We consider the problem of real time Indian Sign Language (ISL) finger-spelling recognition. We collected a dataset of depth-based segmented RGB images using **Microsoft XBOX360 Kinect Camera** for classifying 36 different gestures (alphabets and numerals). The system takes in a hand gesture as an input and returns the corresponding recognized character as the output in real time on the monitor screen. For classification we used **Deep Convolutional Neural Network** and achieved an accuracy of 89.30%.

Introduction

Sign languages are developed primarily to aid deaf and dumb people. They use a concurrent and specific combination of hand movements, hand shapes and orientation in order to convey a particular information. One such set of language is the Indian Sign Language (ISL) system which is predominantly used in south Asian countries. Certain aspects that distinguishes ISL from other sign languages is that ISL devoid of any temporal inflections in its finger spelling chart and also its usage of both the hands.

With the advent of artificially intelligent algorithms coupled with the availability of big data and large computational resources has led to a huge growth in the field of healthcare, robotics, autonomous self-driving vehicles, Human Computer Interaction (HCI) etc.

HCI finds its applications in augmented reality systems, facial recognition systems and also hand-gesture recognition system. This project falls under the broad domain of HCI and aims towards recognizing various alphabets (a-z) and digits (0-9) of the ISL family. Hand-gesture recognition is a challenging task, particularly the ISL recognition is complicated due to its usage of both the hands. In the past many works have been performed in this respect using sensors (like glove sensor) and other image processing techniques (like edge detection technique, Hough Transform etc.) but were unable to achieve satisfactory results. However, with the new deep learning techniques like CNN, the performance in this field has grown significantly leading to many new future possibilities.

Many people in India are speech and/or hearing impaired, and they thus use hand gestures to communicate with other people. However, apart from a handful number of people, not everyone is aware of this sign language and they may require an interpreter which can be inconvenient and expensive. This project aims to narrow this communication gap by developing a software which can predict the ISL alphanumeric hand gestures in real time.

Problem statement

To design a real time software system that will be able to recognize ISL hand-gestures using deep learning techniques. This project aims to predict the 'alphanumeric' gesture of the ISL system.

Objectives of the Research

1. To generate large amount of appropriate dataset using Microsoft Kinect(v1) camera.

2. To apply appropriate image pre-processing techniques in order to remove the background noises and obtain the region of interest (ROI).

3. To design the model and architecture for CNN to train the pre-processed images and achieve the maximum possible accuracy.

4. To develop an algorithm to predict the gesture in real time.

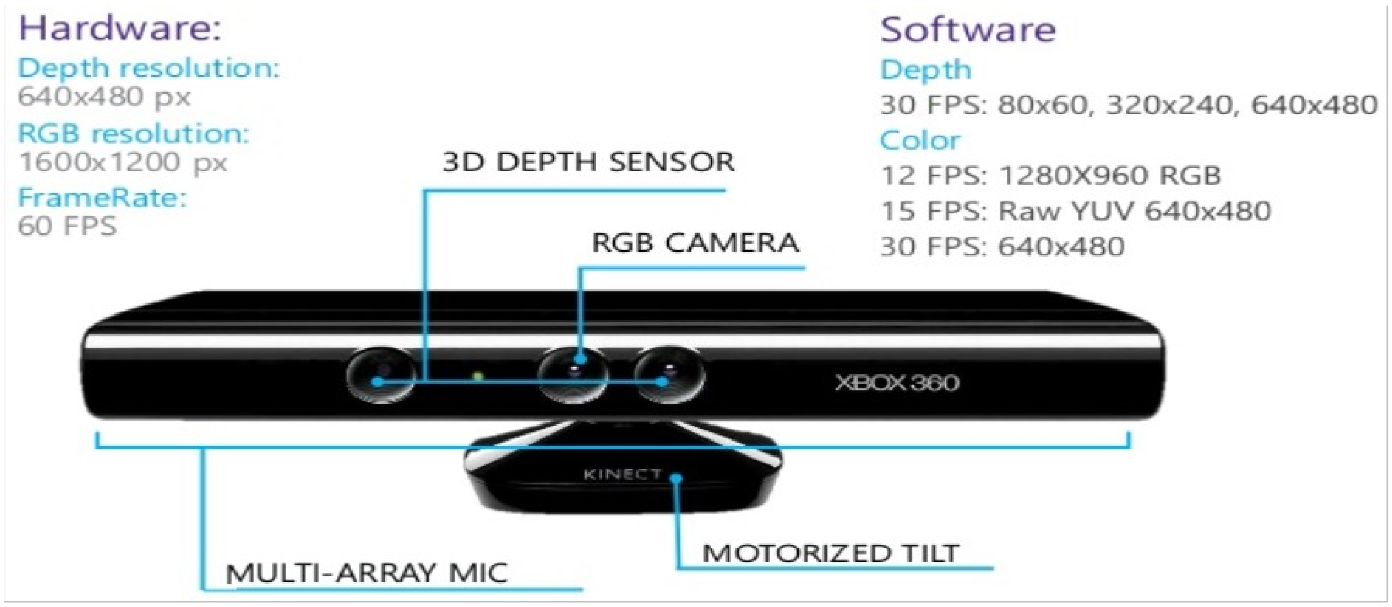
1. LITERATURE REVIEW

Hand-gesture recognition is one of the classic computer vision problem and a lot of work has already been done by researchers and engineers.

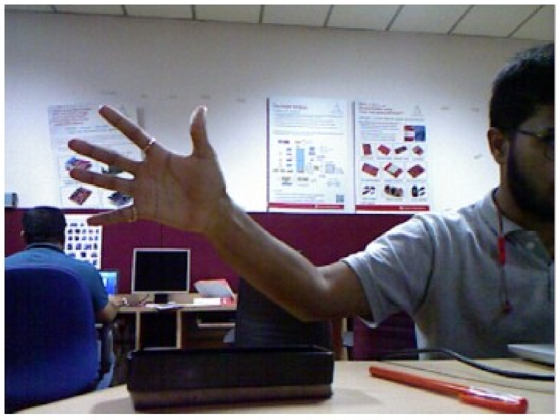
Lionel Pigou and et al. [2] used two networks, consisting of three layers of convolution, each followed by max-pooling. One of the CNNs was trained to capture features from hand and the other from the upper body. The outputs of the two CNNs were concatenated and fed into a fully-connected layer. They utilized the dataset from CLAP14 [3] consisting of 20 Italian gestures by 27 subjects. They used both the depth and color images. They achieved an accuracy of 91.7%on cross-validation containing different users with different backgrounds from the training set and testing accuracy of 95.68% but it contained users and backgrounds from the training set. Alina K. and et al. [4] used a multi-layered Random Forest model, for the dataset collected using the OpenNI + NITE framework on Microsoft Kinect. They achieved an accuracy of 87% for subjects on whom the system was trained whereas 57% for a new subject. Lementec and et.al [5] used glove-based motion tracking sensor to identify gestures. However, given the limitations of having such delicate sensors and gloves makes it unfeasible to use. Hussain and et.al [6] used VGG-16 architecture (CNN) to train and classify hand-gestures. But, in this case the hand gesture must be placed in front of a constant background and any deviation in background will result in incorrect classification. Yamashita and et.al [7] proposed a deep convolutional neural network model for classifying hand-gestures. But they were only able to classify 6 hand-gestures with around 88.78% accuracy. Pei Xum [8], captured raw RGB images using monocular camera, and used background subtraction techniques whereby achieving a classification of over 99% while classifying 16 gestures. B. Liao and et.al [9], used depth sensing camera and Hough transform for hand segmentation and CNN for training. They achieved an accuracy of 99.4% in classifying 24 gestures from the American Sign Language system (ASL).

2. METHODOLOGY

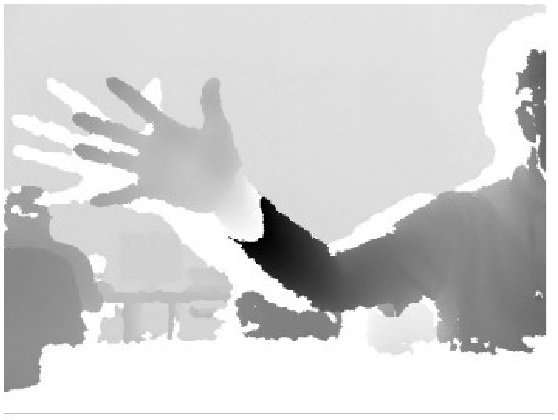
The first step is the data-set collection. We collect the full frame (RGB) image and the corresponding depth map from the Kinect camera as shown in Fig 1.



Microsoft XBOX360 Kinect Camera



Full frame image



Depth image

Fig 1 Kinect camera and images obtained from it

The 3D depth sensor accommodates a monochrome CMOS sensor and an infrared projector that assist to create a 3D virtual imagery throughout its field of view. It actually measures the distance of each point in its field of view by sending out invisible near-infrared rays and measuring its time of flight after it actually reflects off the object present in its field of view.

Extraction of Hand-gesture using depth as a parameter

The depth image functionality of the Kinect camera provides us the ability to extract our region of interest, which in our case is the hand gesture, from the rest of the background. However, given the complexity of the ISL hand gestures, the information present in the full frame image is required in order to differentiate the gestures better.

As described earlier, the depth camera returns the distance of each point in its field of view from the camera. Now using this as a parameter, we can set a threshold, within which the gesture can be shown. Fig 2 depicts a segmented depth image depending upon preassigned distance.

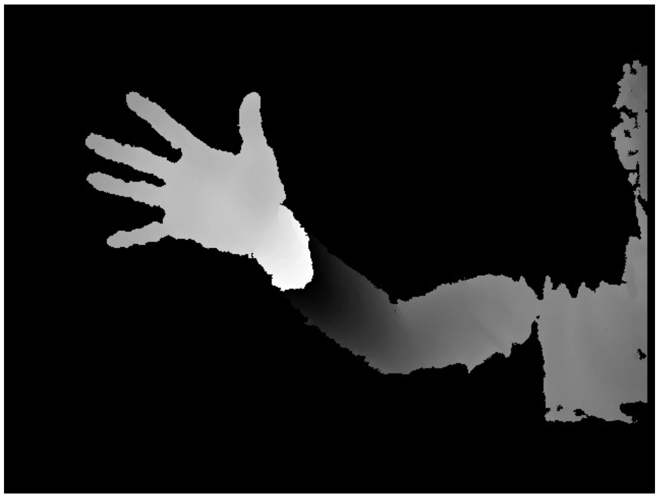


Fig 2 Segmented Depth image

Since RGB image segmentation cannot be performed in any region depending upon the distance, we can use the segmented depth image as a mask and multiply it over the full frame image to obtain the segmented RGB image. However, a major problem arises here. According to the properties of the Kinect v1 camera, the field of view of the depth camera does not coincide with that of the RGB camera and hence due to this constrain there would be a pixel mismatch when we try to perform direct binary masking as shown in Fig 3.

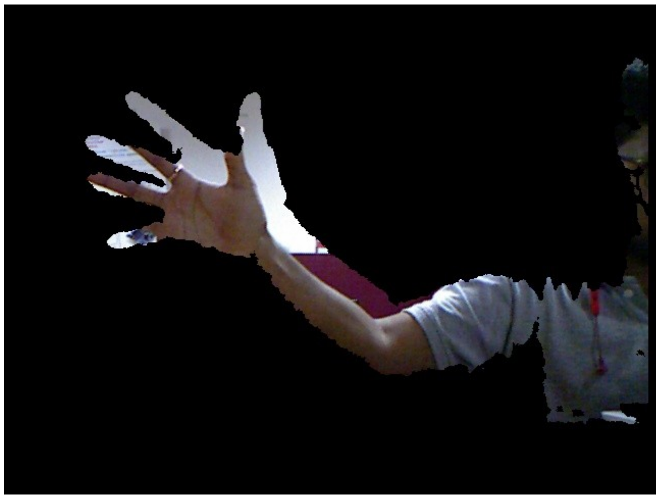


Fig 3 pixel mismatch on direct binary masking

From this it becomes quite clear that certain advanced computer vision techniques must be used to calibrate the pixel distribution mapping from depth image to RGB image in order to obtain one-to-one correspondence between them.

Using the calibration information provided in the website[10], the following parameters are required for calibration

fx_rgb = 5.2921508098293293e+02	fx_d = 5.9421434211923247e+02
fy_rgb = 5.2556393630057437e+02	fy_d = 5.9104053696870778e+02
cx_rgb = 3.2894272028759258e+02	cx_d = 3.3930780975300314e+02
cy_rgb = 2.6748068171871557e+02	cy_d = 2.4273913761751615e+02

fx_rgb, fy_rgb, cx_rgb and cy_rgb denotes the intrinsic of Kinect rgb camera and fx_d, fy_d, cx_d and cy_d denotes the intrinsic of Kinect depth camera.

Rotational matrix,

R = [[9.9984628826577793e-01, 1.2635359098409581e-03, -1.7487233004436643e-02],

[-1.4779096108364480e-03, 9.9992385683542895e-01, -1.2251380107679535e-02],

[1.7470421412464927e-02, 1.2275341476520762e-02, 9.9977202419716948e-01]]

Translational Matrix,

T = [[1.9985242312092553e-02],

[-7.4423738761617583e-04],

[-1.0916736334336222e-02]]

Transformation of raw depth values into meters by the equation (i)

depth (in meters) = 1.0 / (raw_depth * -0.0030711016 + 3.3309495161) (i)

Now each pixel in depth map is projected into a 3D metric space using the equations (ii), (iii) and (iv)

P3D.x = (x_d - cx_d) * depth(x_d,y_d) / fx_d (ii)

P3D.y = (y_d - cy_d) * depth(x_d,y_d) / fy_d (iii)

P3D.z = depth(x_d,y_d) (iv)

Now we re-project each 3D point on the color image and get its color using the equations (v),(vi) and (vii)

P3D' = R.P3D + T (v)

P2D_rgb.x = (P3D'.x * fx_rgb / P3D'.z) + cx_rgb (vi)

P2D_rgb.y = (P3D'.y * fy_rgb / P3D'.z) + cy_rgb (vii)

This method nearly maps the depth and RGB pixel together and helps in segmenting the gestures. However this method has got its own overheads. The processing is tedious with high time complexity which is not a suitable solution for real-time systems.

To overcome the above problem, we actually change the python wrapper itself in the Kinect sdk to align the field of view of both depth and RGB map. This proved to be the most effective solution to this underlying problem and accurate results were obtained with no time overheads. Fig 4 depicts the images obtained after making changes in the python wrapper

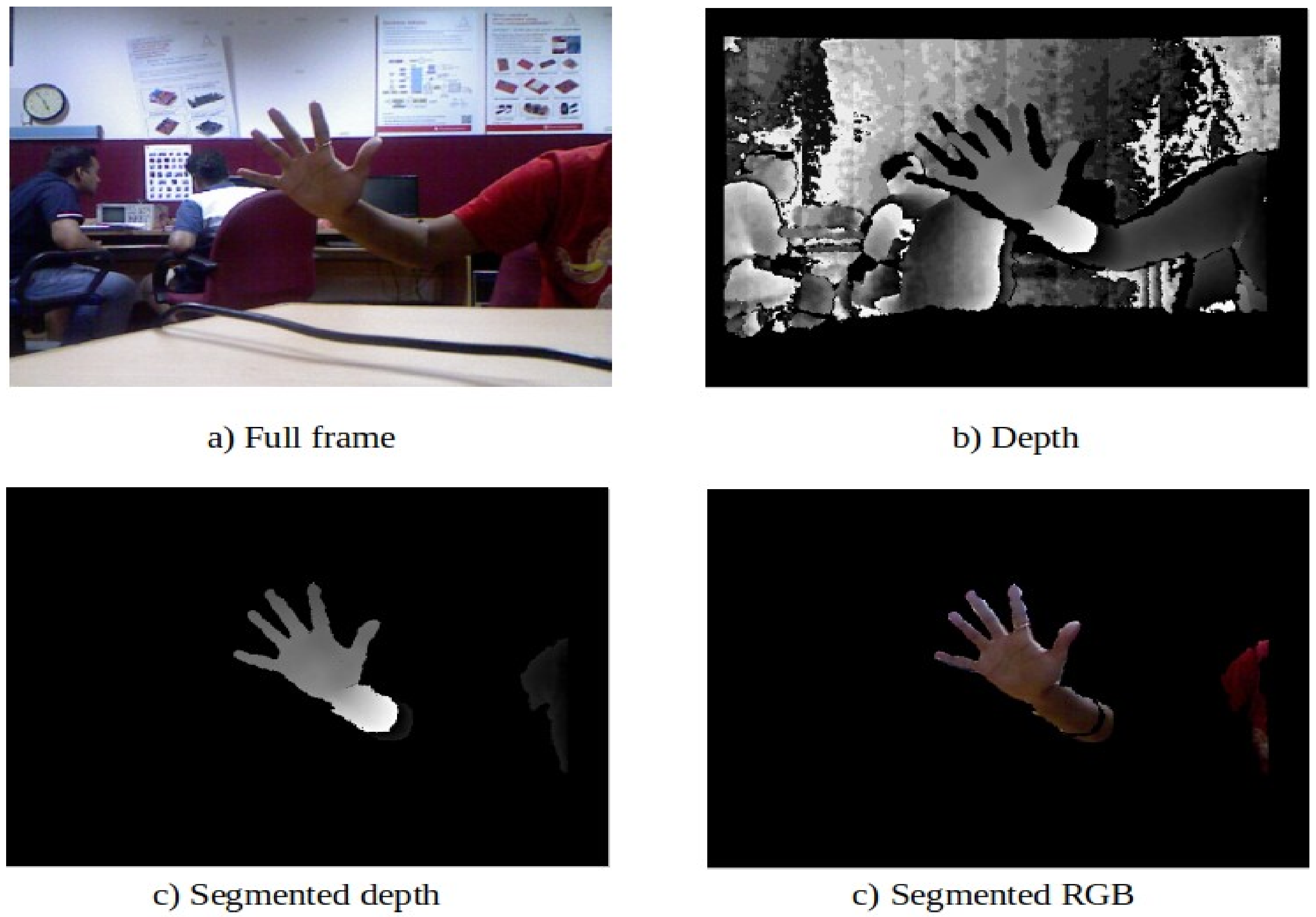


Fig 4 Images obtained after making certain modifications in the Kinect python wrapper

We thus use these generated segmented RGB hand gestures to train our neural network for its accurate classification in real-time. Sample segmented RGB images of each class from our dataset is shown below in Fig 5.

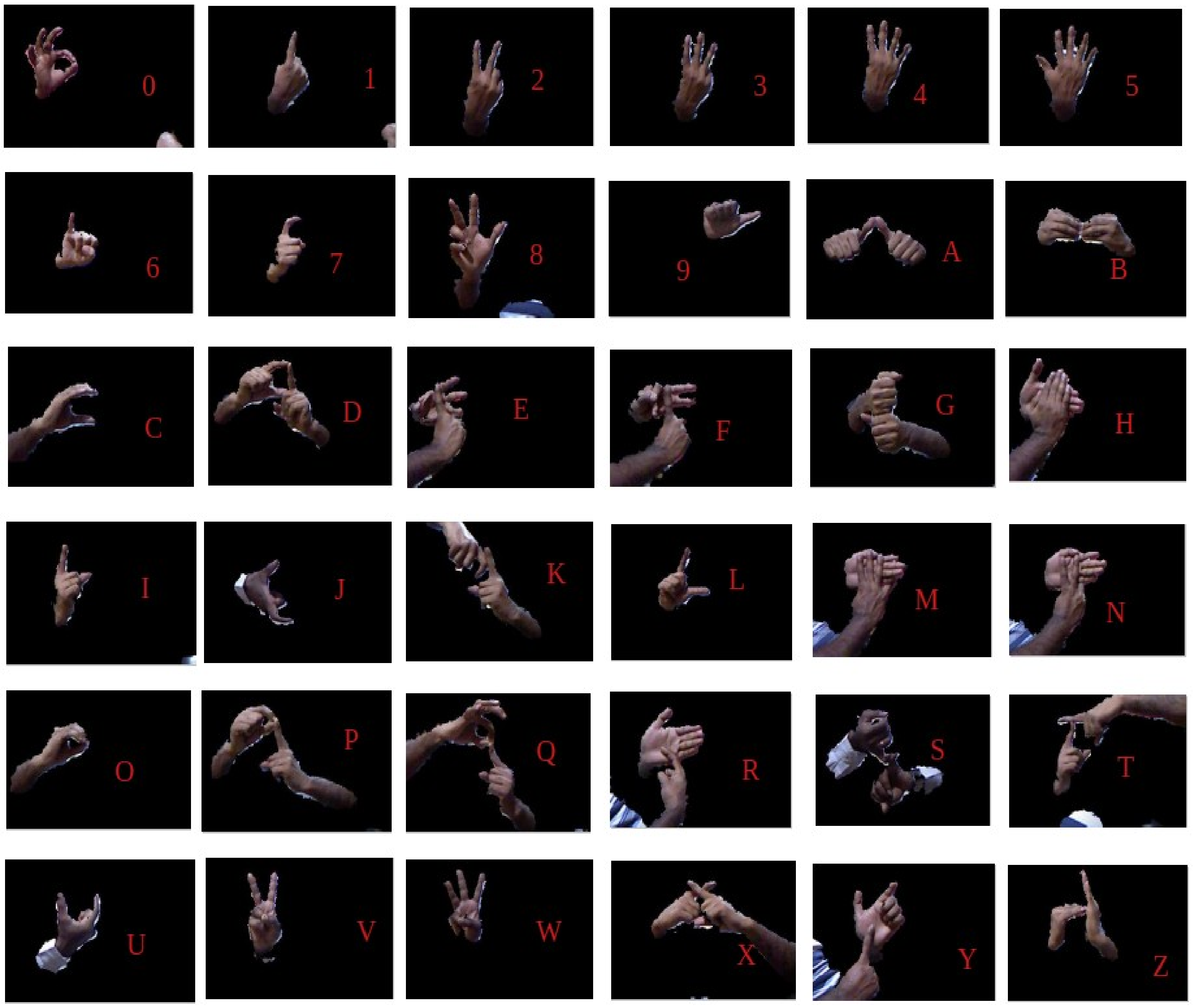


Fig 5 Segmented RGB images with reference to the actual Indian Sign Language Alphanumeric system.

Feature extraction and hand-gesture recognition using Deep Learning

As explained earlier, ISL hand gestures are complex and traditional feature extraction algorithms performs poorly. For example, Canny Edge detection algorithm fails due to the usage of both the hands where edges of one hand can get overlapped or nullified due to the other hand.

Lately, deep learning algorithms has proved to be beneficial in extracting complicated features. Convolutional Neural Network uses the property of convolution, mainly devised for analyzing visual imagery. It consists of one input layer and one output layer and numerous hidden layers in between. The hidden layer consists of convolutional layers that computes the dot product between the weights and regions of the input image. This is usually followed by ReLU(an activation function) and MaxPooling (for down sampling and reduce the output volume). The advantage of using CNN over ordinary ANN is the reduction of number of parameters to train, feature sharing etc. Although high computation power is required for training. Fig 6 depicts the actual architecture of our CNN model and Fig 7 describes the model summary.

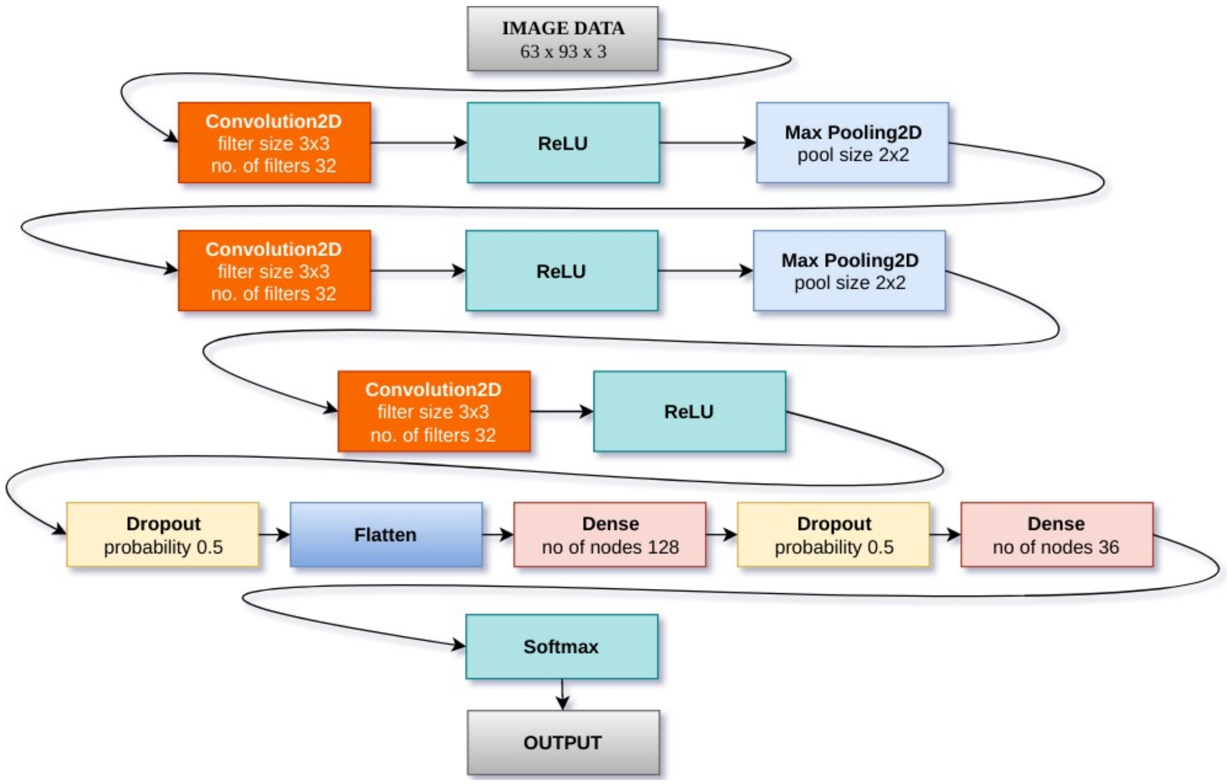
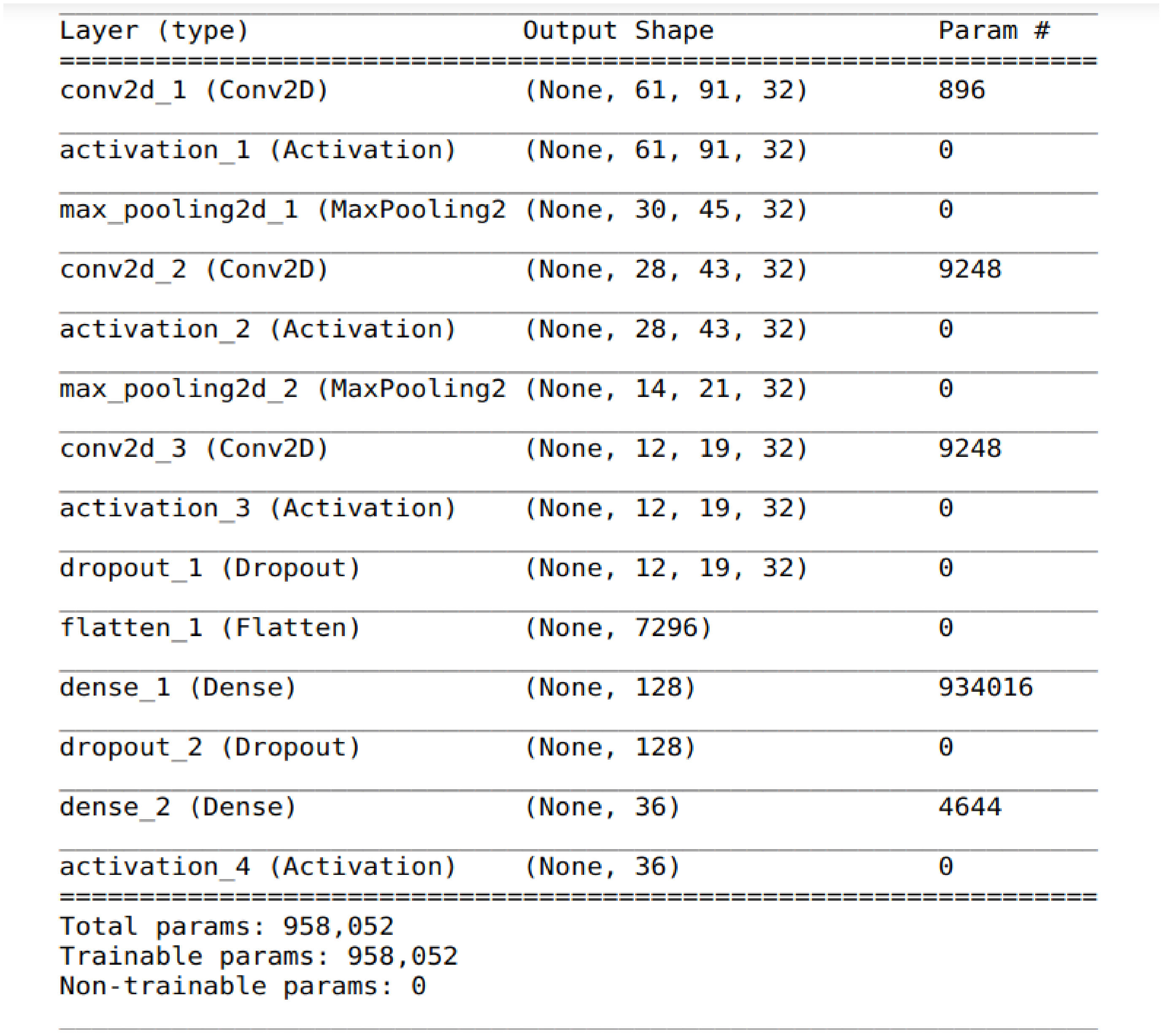


Fig 6 Architecture of the CNN model implemented



```
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 61, 91, 32)        896       
_________________________________________________________________
activation_1 (Activation)    (None, 61, 91, 32)        0         
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 30, 45, 32)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 43, 32)        9248      
_________________________________________________________________
activation_2 (Activation)    (None, 28, 43, 32)        0         
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 21, 32)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 19, 32)        9248      
_________________________________________________________________
activation_3 (Activation)    (None, 12, 19, 32)        0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 19, 32)        0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 7296)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 128)               934016    
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 36)                4644      
_________________________________________________________________
activation_4 (Activation)    (None, 36)                0         
=================================================================
Total params: 958,052
Trainable params: 958,052
Non-trainable params: 0
_________________________________________________________________
```

Fig 7 CNN model summary

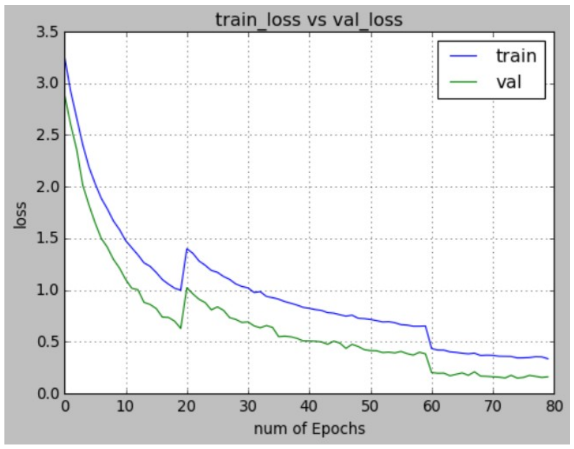
Using this CNN model we were able to achieve training accuracy of about 89% with validation accuracy of about 96%. The various training performance of our model is shown below in Fig 8.

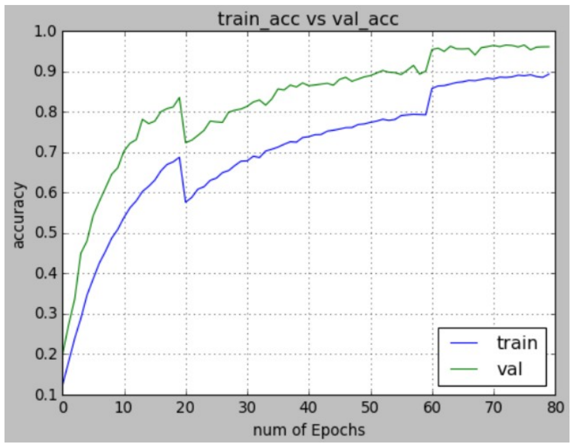
Our image dataset is around 46000 images which was divided into two parts in the ratio 80:20 for training and testing. The training images we duplicated and added to the training set. Thus we now have around 74000 images to train. We divide this into 4 parts of around 18000 images each. We randomly synthesize these images and train each set for 20 epochs one after the another.

Parameters for image augmentation

i. Rotation range = (+/-) 35 degrees ii. height and width shift range = 0.16

Artificial synthesis (augmentation) was performed in order that the neural network learn the image patterns better so that it will be able to classify the images better in real time.





a) training loss and validation loss b) training accuracy and validation accuracy

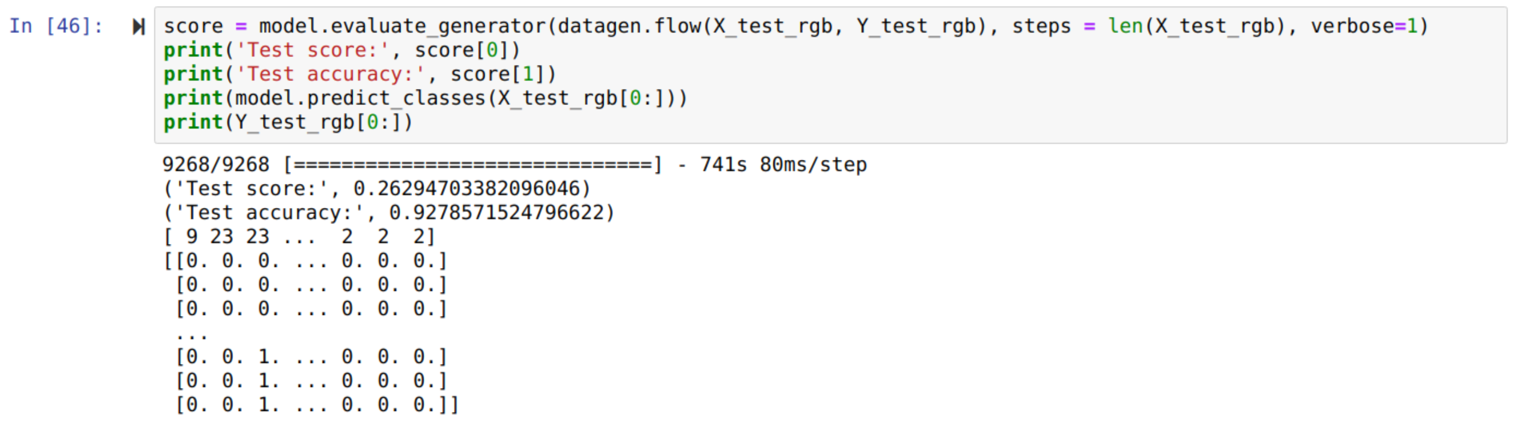
Fig 8 variation of losses and accuracy with epochs

The model was saved as a .h5 format file, and this was called during real time testing.

For real time implementation, each video frame of the Kinect camera was processed and segmented based on depth, and finally passed through the called trained model for its classification.

3. RESULTS AND DISCUSSION

Upon training the image dataset without any augmentation, the training accuracy achieved was very high (around 99%) but, the real time performance was not up to the mark. It was predicting incorrectly most of the times because in real time hand-gestures were not placed exactly at the center and aligned vertically. In order to overcome this shortcoming, we trained our model by augmenting our dataset. The training accuracy was reduced to 89% but the real-time predictions were predominantly correct. Offline testing of about 9000 augmented images showed an accuracy of 92.7% as shown in Fig 9.



```
In [46]:  score = model.evaluate_generator(datagen.flow(X_test_rgb, Y_test_rgb), steps = len(X_test_rgb), verbose=1)
          print('Test score:', score[0])
          print('Test accuracy:', score[1])
          print(model.predict_classes(X_test_rgb[0:]))
          print(Y_test_rgb[0:])

9268/9268 [==============================] - 741s 80ms/step
('Test score:', 0.26294703382096046)
('Test accuracy:', 0.9278571524796622)
[ 9 23 23 ...  2  2  2]
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 1. ... 0. 0. 0.]
 [0. 0. 1. ... 0. 0. 0.]
 [0. 0. 1. ... 0. 0. 0.]]
```

Fig 9 Offline testing of randomly augmented images

Also, when we increase our training image size the accuracy betters itself both during training and real time implementation (as the number of parameters to be trained increases). However, training images with large size is directly proportional to the computational power of the system. My system RAM is limited to 12 GB, so the optimal image size can be trained was 93X63 pixels (frame size received from Kinect is 640X480). Some real time testing instances of our proposed model with different people is shown in Fig 10.

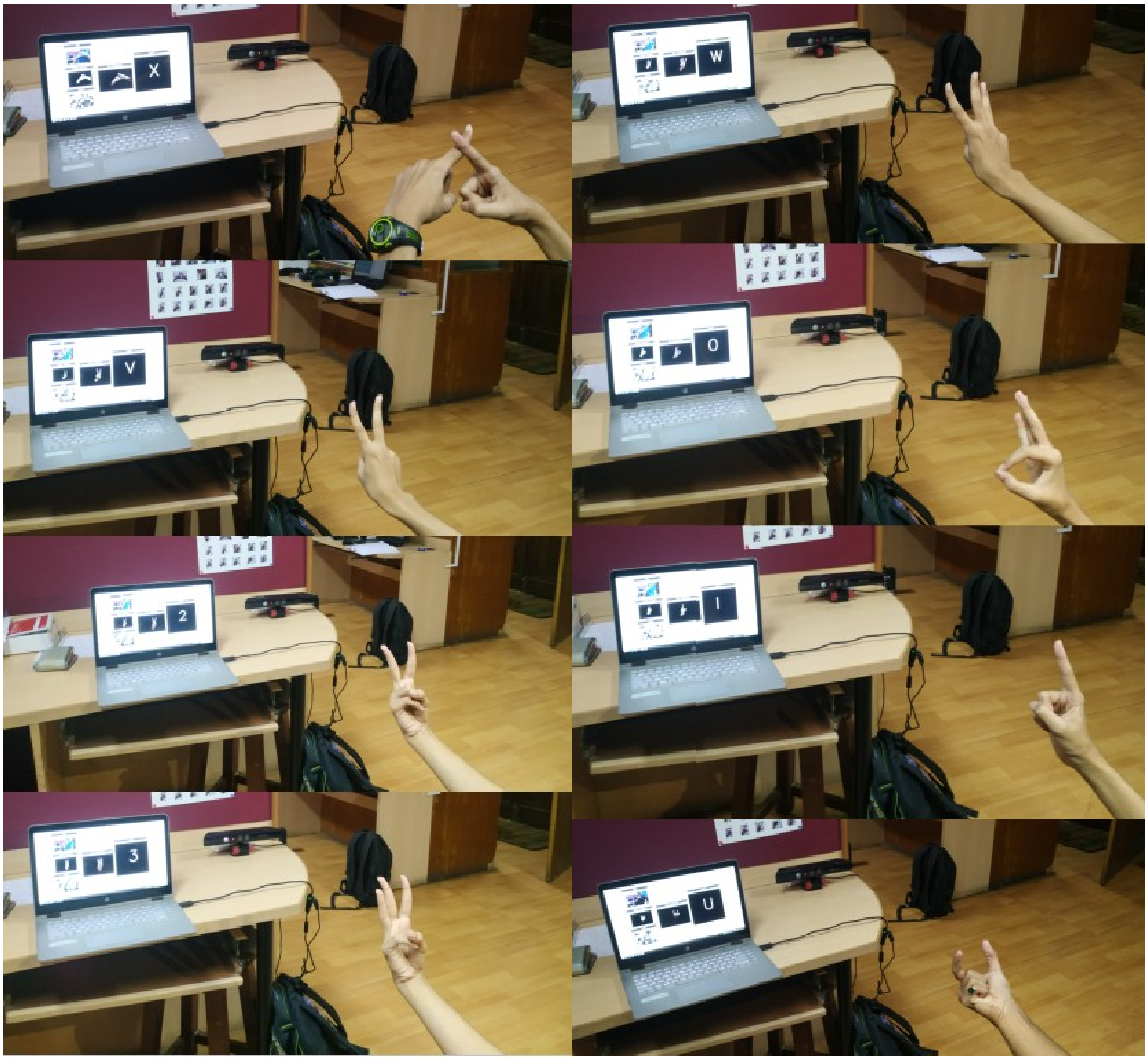


Fig 10 Some instances of real time implementation

4. CONCLUSION AND FUTURE WORK

The aim of this project is to predict the ISL alphanumeric hand-gestures in real time. The above work shows that it can be solved with better accuracy when we actually consider the depth-based segmented RGB hand-gestures. By applying this depth-based segmentation we remove the overheads of dynamic background. These segmented RGB hand-gestures were fed to 3 layered CNN for training and testing in real time. We were able to achieve training accuracy of 89.30%. Our model showed good accuracy while predicting results both offline and online.

Future scope

1. We can develop a model for ISL word and sentence level recognition. This will require a system that can detect changes with respect to the temporal space.

2. We can develop a complete product that will help the speech and hearing impaired people, and thereby reduce the communication gap.

5. REFERENCES

[1] Mukesh Kumar Makwana, " Sign Language Recognition", M.Tech thesis, Indian Institute of Science, Bangalore

[2] Pigou, Lionel, et al. ”Sign language recognition using convolutional neural networks.” Workshop at the European Conference on Computer Vision. Springer International Publishing, 2014._

[3] Escalera, Sergio, et al. ”Chalearn looking at people challenge 2014: Dataset and results.” Workshop at the European Conference on Computer Vision. Springer International Publishing, 2014.

[4] Kuznetsova Alina, Laura Leal-Taix, and Bodo Rosenhahn. ”Real-time sign language recognition using a consumer depth camera.” Proceedings of the IEEE International Conference on Computer Vision Workshops. 2013.

[5] J. -. Lementec and P. Bajcsy, "Recognition of arm gestures using multiple orientation sensors: gesture classification", Proceedings. The 7 th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No.04TH8749), Washington, WA, USA, 2004, pp. 965-970.doi: 10.1109/ITSC.2004.1399037.

[6] S. Hussain, R. Saxena, X. Han, J. A. Khan, and H. Shin, “Hand gesture recognition using deep learning”, 2017 International SoC Design Conference (ISOCC) , Seoul, pp. 1-6.

[7] T. Yamashita and T. Watasue, "Hand posture recognition based on bottom-up structured deep convolutional neural network with curriculum learning", 2014 IEEE International Conference on Image Processing (ICIP) , Paris, 2014, pp. 853-857.

[8] Pei Xum “A real time hand gesture recognition and human computer interaction,” Dept. of Electrical and Computer Engineering, University of Minnesota, 2017, pp. 1-8.

[9] B. Liao, J. Li, Z. Ju and G. Ouyang, "Hand Gesture Recognition with Generalized Hough Transform and DC-CNN Using Realsense," 2018 Eighth International Conference on Information Science and Technology (ICIST) , Cordoba, 2018, pp. 84-90.

[10] http://nicolas.burrus.name/index.php/Research/KinectCalibration

6. ACKNOWLEDGMENTS

I would like to convey my heartfelt thanks and gratitude to one and everyone who has helped me throughout the course of this project. I thank my guide, Dr. G N Rathna for giving me this wonderful opportunity to work on this project. Her valuable suggestions and motivations were of immense help. I would also like to thank Navin Kumar, Neelkamal Bhagat, Srujana Subramaniya and Vishnusai Y to take their time out and help us with the dataset generation. I thank all the people of Digital Signal Processing Lab for their support and encouragement. Last but not the least, I would like to thank my parents for their constant support, love and blessings.