Sign Language Detection

Asmaa Ibrahim, *201701056* Elsayed Mostafa, *201700316* Muhammed Alasmar, *201700038*
and Salma Hassan, *201601152*

***Abstract*—In this project, a hand sign language detection system is developed using computer vision and deep learning techniques. The proposed system performs hand segmentation, hand tracking and hand sign language classification. HSV color space based segmentation, Convex Hull algorithm and transfer learning are all used approaches in the development. The whole pipeline was run on 2 different benchmark devices while computing FPS, accuracy statistics and confusion matrix. The best obtained accuracy is 65% due to multiple limitations on both the deep learning model and the used computer vision techniques.**

***Index Terms*—Sign Language, Convex Hull, CNN, HSV, Object Tracking, Hand Sign.**

I. Introduction

ONE of 1000 babies are born deaf [1]. This difficulty in hearing leads to their inability to talk. Sign language has been of help to diabled talking to each other and communicating with people who know sign language. However their communication with the rest of the world is badly limited. Sign language is not a universal language. Unlike the famous arabic sign language that uses signs to represent a certain set of words. Another sign language called letter sign language or fingerspelling represents letters using hand gestures. This language is customized for each language, but it is a very promising means of communication to people unable to talk or hear if the process of interpreting it could be automated. In this project, MNIST english letter sign language dataset [2] will be used to take steps towards this solution. This project aims at producing a deployable efficient model that captures hands and recognizes the sign letter.

II. Application Idea and Procedures

Our proposed idea is to have an application that reads the video frames sequentially and recognizes the hand letter sign exists in them. First, the video is read into frames. Then, a hand detection algorithm is applied to a frame to detect the region where a hand exists. This hand detection is performed for one frame after some frames. In other words, multiple frames are ignored and then, one frame is used to detect the hand region. In the frames where no hand detection is applied, object tracking is performed to keep track of the region of the detected hand. This alternation between hand detection and hand tracking is done to improve the performance of the application since the hand detection is much more complex than object tracking. Once the hand region is detected, a frame is selected to enter into a pre-trained deep learning model. The output of the model is the sign language letter. Finally, the predicted class/letter is displayed on the screen.

A. Hand Segmentation

The first step in the application pipeline is to detect a region of interest where a hand exists. To do so, two approaches were initially proposed. The first approach is HSV which is a color space based segmentation method [3]. The image is converted from its RGB color space to HSV color space which then can be used to differentiate the specific color region (skin color in our case). Next, we use the convex hull algorithm to draw a polygon around the region that encloses the hand [5]. The second approach is Haar-like features classifier [4]. In this approach, a classifier is trained using both negative and positive data. The positive data is the set of images containing the object to be detected only (hands in our case). On the other hand, the negative data is the set of data that contain multiple objects that can exist around the object to be detected. In case of using a Haar-like approach, we can create our own classifier using a ready-made GUI tool to have the needed xml file to use with openCV library to detect the hand in the image/frame. In our application, we continued with the first approach which is using the convex hull with HSV color space segmentation.

B. Hand Tracking

After the detection, we will perform object tracking. Object tracking makes it possible to get the object when detection fails. In addition, It saves the computation effort of detecting in every frame. So, we will track the segmented hand for several frames. Then, refreshing the object place by running the detection process then running the tracking again. For this project, we are using a ready made object tracking module. The module we are using is OpenCV tracker API[6].

C. Sign Language Classification

For the classification part, we will use a deep learning model that is trained using MNIST dataset. MNIST is english letter sign language dataset [2]. This dataset has training data of 27455 images and test data of 7172 images. The dataset is labelled 0-25 each label signifying one english letter with zero instance for label 9=j and 25=z because they require gesture motion. The images are cropped 28x28 pixels. The dataset is an adaptation of another dataset containing 1703 uncropped instances. The images were cropped, resized and finally augmented by applying filters ('Mitchell', 'Robidoux', 'Catrom', 'Spline', 'Hermite'), along with 5% random pixelation, +/- 15% brightness/contrast, and finally 3 degrees rotation. Another dataset of American Sign Language (ASL) was also explored to compare results with those of the MNIST dataset [7]. The ASL dataset offered a wider range of 87 thousand pictures of 29 classes of 26 letters and another

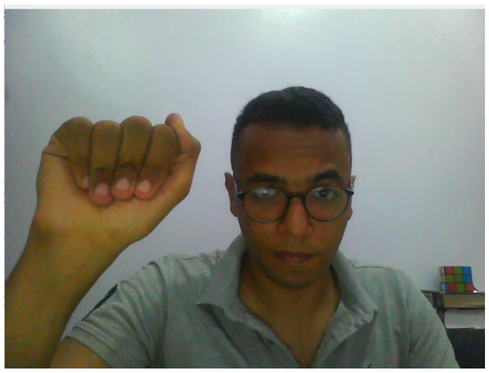
Fig. 1. Random captured image from the camera.

3 classes for space, delete and nothing. After capturing the hand, a trained model should decide what sign letter this hand gesture represents. This model will be generated using convolutional neural networks and trained with the aforementioned dataset. For the generation of the model, we will make use of transfer learning, surveying the ready-made models and their performance, picking the most promising models and optimizing them for our problem.

III. Implementation

Our implementation for the different function depends mainly on python, Jupyter notebook and OpenCV functions. Detailed implementation discussion is given in addition to the main pipeline and evaluation criteria.

A. Hand Segmentation

The first step in hand segmentation is to map the input frame image into HSV color space. The range of HSV color regions is tunable and varies a little bit according to the skin of the user. We tuned the HSV color region to work acceptably during our demo. After HSV conversion, the image is thresholded between 0 and 1 to separate the region of hand from the other similar colored regions. Additional blurring is applied to remove weak points. Next, the Convex Hull python implementation is applied to draw the needed polygon around the region of interest (roi) that contains the hand segment. Figures (1), (2), (3), (4) show the operations sequence in the segmentation step. The hand is now segmented and its region is decided to be passed into the tracker. It is needed to say that the values in HSV thresholding must be tuned continuously according to the application and the skin prediction in addition to the illumination conditions.

B. Hand Tracking

Object tracking is implemented using the ready made API by Opencv. In this implementation, the tracker used is the MOSSE tracker. This tracker is known for its speed.[8] That will be suitable for real-time applications and can handle if the users moved their hands fast. The tracker object is initialized using the region of interest generated in the segmentation process.

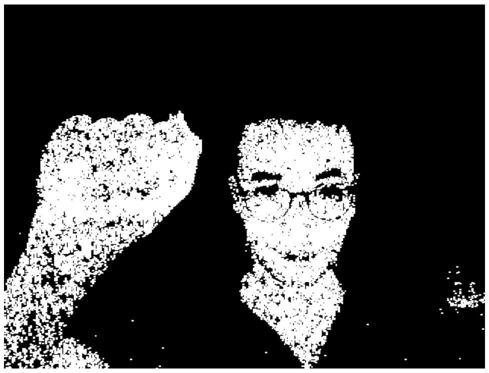
Fig. 2. Threshold-ed image after HSV conversion.

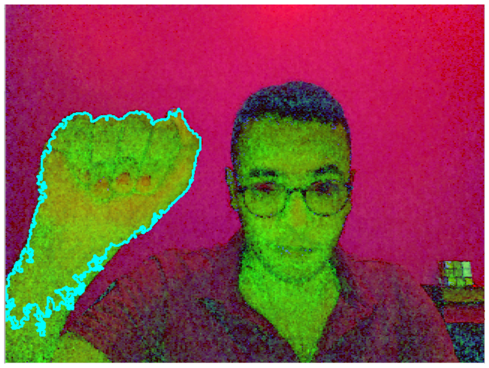
Fig. 3. After drawing contour around the hand.

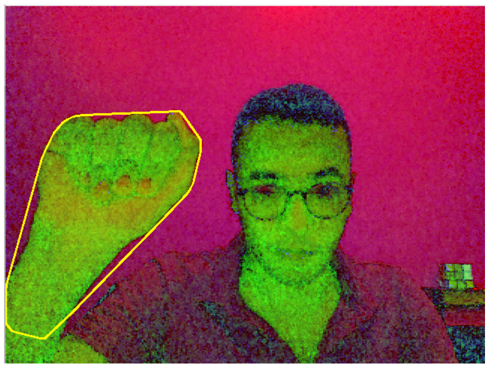
Fig. 4. After using Convex Hull to extract the roi.

C. Sign Language Classification

A deep convolutional neural network was trained on the MNIST dataset to capture features needed to recognize sign language letters. Each row in the MNIST dataset represented a 28*28 image which is fed into the network. Then, a model is trained using categorical cross entropy as a loss function and Adam as an optimizer. Achieved results were satisfying on the test data set of above 95% accuracy of detecting letters, but when the model is used to predict general real life images of letters, it achieved lower accuracy. This may be due to the limitations on the dataset itself such that it has very low resolution images for the model to learn the distribution. It could also be that all images have nearly similar content of each letter containing only the palm representation of a sign letter. Seeking better performance for classifying letters, transfer learning has been used on ASL dataset. ASL offered a wider variety of images that could make a difference in generalizing on real life applications. MobilenetV2 was used as a pretrained model with imagenet weights and average pooling. Before feeding dataset images to the model, they were preprocessed using Mobilenet preprocessing function and other augmentations like changing brightness range and rotation. Training generator was fed into the model and achieved a validation accuracy of 92%. These results were promising, so when the model was used to predict on the test images , it achieved an accuracy of about 95%. When tried on real life images, it achieved a better performance but still can not be reliable. This may be due to the fact that there are sign letters that are similar to each other. It could also be the augmentation done on the data that made the model learn distributions for letters that are supposed to be for others.

D. Pipeline

The three aforementioned parts were assigned together as follows: 1. Every 60 frames a frame is detected using Image Segmentation. 2. Otherwise the last detected frame is tracked. 3. The resulting frame from any of the two tracks is fed into the trained neural network. 4. The result is predicted with a voting algorithm discussed below.

E. Decision Making

The voting algorithm is that a letter is chosen if it was predicted by the model for 7 frames in the last 30 frames. For the first 7 frames the model decision is the chosen prediction. The decision of each frame is saved in a list. We experimented with different algorithms and this algorithm was the most promising one. We've tried saving only the last 5 frames, 10 frames, 20, and 30 frames. We've also experimented with the number of frames required to match from the saved frames, we tried 3, 4, 6, 7, 8, and 10. The best results were obtained with 7 frames of 30 saved ones.

F. Evaluation

The evaluation of our pipeline was done on videos that each of us created containing different numbers of letters, hand labelling was done for each video and encoded in the confusion matrix function. Accuracy is calculated by comparing the prediction and labels lists. The analysis speed was calculated with fps to test how the algorithm would do on a live video. Finally the number of maximum correct consecutive frames was calculated.

IV. Results and Discussion

The results were tested on many benchmark devices although there were limitaions and difficulties on using COLAB as it was running out of memory. So, there was not a lot of testing using COLAB. Table 1 summarizes the most important results obtained.

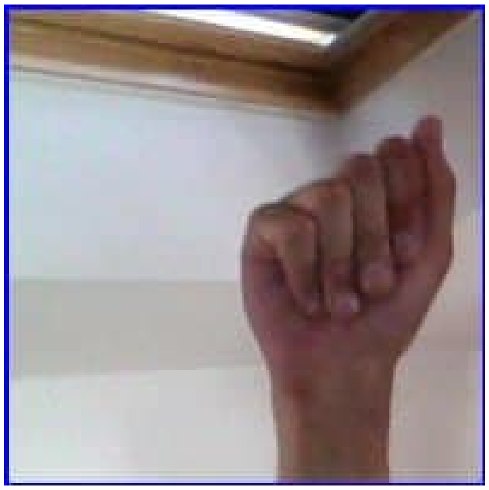
Fig. 5. Letter A.

Video ID	FPS Processed	Benchmark	Accuracy(%)
E3	11	B.A	65
E3	9	B.C	56.5
S1	6	B.A	32.1
E2	11	B.A	26
AS	12	B.A	29

Table 1. Experiments Results

Results were not as fascinating as one would require. However, this dataset and idea are fairly new, little literature was found. Every part of the pipeline was implemented without a direction of what is promising and what is not. However as seen some examples however outperformed others. The video titled: Sayed_Demo3 for instance is our highest accuracy, with 65% and 63% consecutive correct frames. This model had many reasons to outperform other models Photographed Clearly (The orientation, the distance etc) contained clear sign letter C, O, A. Segmentation was mainly tuned on photos of one person. However this does not mean that our model cannot perform well on other instances, so for instance the video titled: Asmaa_Demo contains a variety of different letters B, I, N, and G. Each of the three letters was detected in many frames. However the orientation of the camera used and brightness were not good enough. It is worth mentioning that the model actually made very smart mistakes, such as mistaking B for W which are close. Also, I and A. This proves that our pipeline is working very well, as well as mostly every part of the project. Especially the hand segmentation and the tracking parts. Tuning of the deep learning model, and processing of the incoming frames should be the only steps required for a better performing project.

V. Conclusion

The developed hand sign language detection system can be used and deployed in multiple applications in the industry especially the mass communication industry. More robust deep learning model can be developed using MobileNet or EfficientNet architectures. For hand segmentation, Haar-like machine learning based segmentation can be merged with the deployed

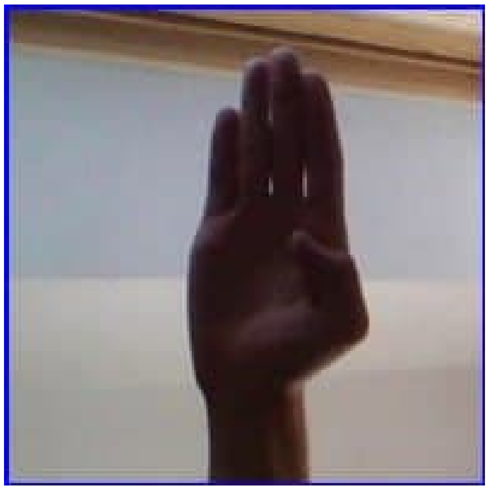
Fig. 6. Letter B.

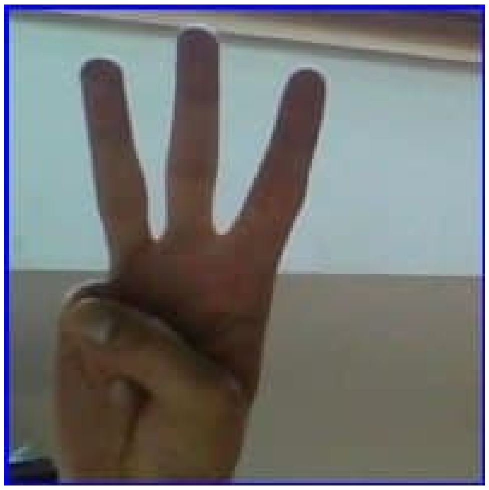
Fig. 7. Letter W.

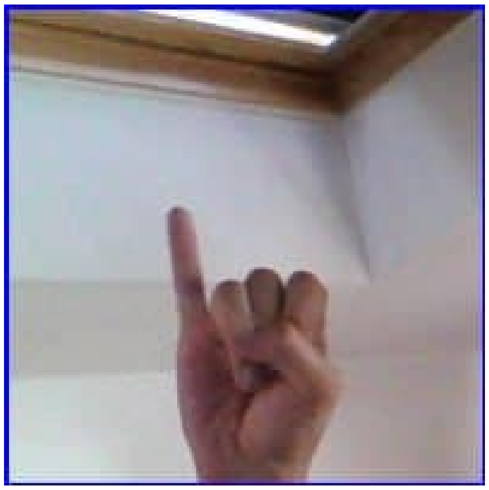
Fig. 8. Letter I.

HSV-based segmentation to enhance the segmentation results. Moreover, the obtained system can be more tested and tuned for different skin colors and illumination conditions to be more robust and generalized.

Appendix A
Benchmark Abbreviation

B.A: Laptop HDD - 16G RAM
B.C: COLAB Free with GPU.

Appendix B
Video ID Abbreviation

E3: Video titled: Sayed_Demo3.mp4
S1: Video titled: Salma_Demo3.mp4
A1: Video titled: Asmar_Demo1.mp4
S2: Video titled: Sayed_Demo2.mp4

Acknowledgment

The authors would like to thank Eng. Ahmed Elsayed for his continuous help and guidance. Special Thanks to Prof. Elsayed Hemayed for his great efforts during the CIE 552 course.

References

[1] ”Deafness and hearing loss”, Who.int, 2021. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss. [Accessed: 22- May-2021].

[2] ”Sign Language MNIST”, Kaggle.com, 2021. [Online]. Available: https://www.kaggle.com/datamunge/sign-language-mnist. [Accessed: 22- May-2021].

[3] Noreen, U., Jamil, M. and Ahmad, N., 2016. Hand Detection Using HSV Model. INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH, [online] 5(1).[Accessed 20 May 2021].

[4] Chen, Q., M. Petriu, E. and D. Georganas, N., 2007. Real-time Vision-based Hand Gesture Recognition Using Haar-like Features. [online] [Accessed 20 May 2021].

[5] Dhawan, A. and Honrao, V., 2013. Implementation of Hand Detection based Techniques for Human Computer Interaction. International Journal of Computer Applications (0975 – 8887), [online] 72(17).[Accessed 20 May 2021].

[7] ”ASL Alphabet”, Kaggle.com, 2021. [Online]. Available: https://www.kaggle.com/grassknoted/asl-alphabet. [Accessed: 22- May- 2021].

[8] ”Object Tracking using OpenCV (C++/Python)”, Learn OpenCV — OpenCV, PyTorch, Keras, Tensorflow examples and tutorials, 2021. [Online].[Accessed: 04- Jul- 2021].