

RECOGNIZATION OF HAND GESTURES USING MEDIAPIPE HANDS

Kavana KM^{*1}, Suma NR^{*2}

^{*1}Dept. Of MCA, Bangalore Institute Of Technology, Bengaluru, India.

^{*2}Dept. Of MCA, Assistant Professor, Bangalore Institute Of Technology, Bengaluru, India.

ABSTRACT

Touchless interaction is an active area of research due to its application in the area of medical system to gaming. Sign language could benefit a people with hearing difficulty. It is important to realize sign language recognition on smartphones. Hand gestures are an effective touchless way to interact with computer systems. Several methods have been proposed for hand gestures. We are presenting a real-time on-device hand tracking solution which predicts a hand skeleton of a human using a single RGB camera. Here we are making use of Mediapipe library provided by google which is open-source and it is a well trained model to achieve high performance. Gestures of a hand can be determined using Mediapipe library using different technologies. In this Mediapipe hands library will use two models. 1) a palm detector, that is providing a bounding box of a hand to, 2) a hand landmark model, that is predicting the hand skeleton. User can easily interact with the computers having RGB cameras using gestures.

I. INTRODUCTION

In recent years we have seen so many people with hearing difficulty. Face-to-face communication is an very important channel that can convey message, feeling, and personal connection. People who have hearing difficulty use two types of communication, which are namely reading writing and a sign language. However, the inability to communicate has become an obstacle to the advancement of the hearing impaired it should be solved. Therefore, we focus on sign language as the most used means of communication. Hand tracking is a requisite component to provide a natural way for interaction and communication in AR/VR, and it has been an active research topic in the industry. Currently, there are many devices that use voice commands, especially on smartphones that we often use, voice commands themselves are based on speech recognition algorithms. Due to some of the problem like noisy background voice command itself has several shortcomings. An alternative to this problem is hand gestures. And also communication using hands is risky due to covid pandemic which requires maintaining distance.

Different countries have their own sign languages. A sign language is not a universally common language, the U.S. has American Sign Language (ASL), the U.K. has British Sign Language (BSL) and China has its own sign language named Chinese Sign Language (CSL), Thailand has Thailand sign language TSL. Basically this sign language employs two signing modes: they are semantic and spelling signings. One of the communication media for deaf friends is to use sign language. In this paper, we propose a innovative approach that does not require any additional hardware and performs in real-time on mobile devices.

Our main contributions are:

- we are doing an efficient two-stage hand tracking pipeline which can track multiple hands in real-time on mobile devices.
- A hand pose estimation model that is capable of predicting 2.5 Dimensional hand pose with only RGB input.
- And open source hand tracking pipeline named media pipe as a ready-to go solution on a variety of platforms, including Android, iOS, Web and desktop PC's.

Therefore, we created a project about hand gesture detection that can detect the movement or pose of the hand to be implemented in sign language recognition. The method used to perform hand gesture detection, of course, is to use computer vision which goes through several adjustments and filters on the hand. Then, the data obtained will be processed to be able to provide the appropriate output.

In this paper, we attempt to simplify fingerspelling recognition and data processing using MediaPipe Hand proposed by Google, which can be used with typical digital camera.

II. LITERATURE SURVEY

We will get the dataset (collection of images) of different sign language's from the google and then our mediapipe models will be work on them. Then the collected data will be in the form CSV for the collected dataset Data cleaning , Data normalization and training of data will be carried out then the machine learning algorithms will be work on those dataset then finally predictions will be done. Different algorithms and their accuracy for the dataset will be calculated. Mediapipe is the best library for the gesture recognition.

Previous contribution

Fan Zhang Valentin Bazarevsky Andrey Vakunov Andrei Tkachenka George Sung Chuo-Ling Chang Matthias Grundman he had done a On-device hand tracking using mediapipe an end to end hand tracking is possible. Arsheldy Alvin¹ , Nabila Husna Shabrina² , Aurelius Ryo³ , Edgar Christian⁴ Fakultas Teknik dan Informatika, Universitas Multimedia Nusantara, Teknik Kompute Hand Gesture Detection for American Sign Language using K-Nearest Neighbor with Mediapipe was achieved.

III. METHODOLOGY

MediaPipe :

MediaPipe is an open source cross-platform framework provided by Google to build a pipeline for processing perceptual data from different modalities such as video and audio. The solutions used in MediaPipe include multiple items such as posture estimation and face recognition. In this paper, we will use MediaPipe Hands for hand tracking. A similar skeletal estimation model, OpenPose , has been proposed. OpenPose can estimate 2D joint point coordinates from the input image, along with their reliability. In contrast, MediaPipe uses regression analysis to calculate the finger coordinates from the detected palm of the hand. Compared to OpenPose, our MediaPipe library reduces the amount of computation by using a smaller detection range. Also, by reducing the detection range, we were able to improve the accuracy of coordinate estimation for finger shapes, which OpenPose was not good at. Along with that from a single frame captured by a monocular camera, 21 3D coordinates in X, Y, and Z can be inferred and obtained. Of the output 3D coordinates obtained, the X and Y coordinates are normalized by the width and height of the bounding box. The Z coordinate represents the depth information, and its value becomes smaller as it gets closer to the camera and larger as it gets further away from the camera, using the wrist location as the origin. By using this method, we can perform advanced hand and finger tracking with less processing, and it can work in low performance environments such as mobile environments.

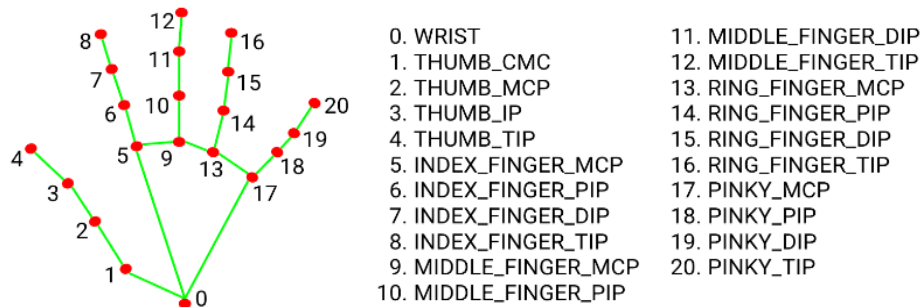
a)Palm detection Model

To detect initial hand locations, we employ a single shot detector model optimized for mobile real-time application similar to Blaze Face, which is also available in MediaPipe. Detecting hands is a not so easy it's a complex task: In this our model has to work across a variety of hand sizes with a large scale span and it should be able to detect occluded and self-occluded hands. Whereas our faces have high contrast patterns, e.g., around the eye and mouth region, the lack of such features in hands makes it comparatively difficult to detect them reliably from their visual features alone. Our solution is an alternative to the above challenges using different strategies. First, we train a palm detector instead of training a hand detector, since estimating bounding boxes of rigid objects like palms and fists is significantly simpler than detecting hands with articulated fingers. In addition, as palms are smaller objects, the non-maximum suppression algorithm(NMS) works well for the two-hand self-occlusion cases, like handshakes. Moreover, palms can be modelled using only square bounding boxes and ignoring other aspect ratios, and therefore reducing the number of anchors by a factor of 3~5. Second, we are also using encoder-decoder feature extractor similar to FPN for a larger scene-context awareness even for small objects. Lastly, we will minimize the focal loss during training to support a large amount of anchors(rigid boxes) resulting from the high scale variance.

b) Hand Landmark Model






After the palm detection is over the whole image our subsequent hand landmark model performs precise key point 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 our Hand

landmark model is robust even to partially visible hands which are blurry sometimes and self-occlusions. To obtain ground truth data in our model, we have manually using 30K real-world images with 21 3D coordinates, as shown below (we take Z-value from image depth map, if it exists per corresponding coordinate). To cover the possible hand poses and also to provide additional supervision on the nature of hand geometry, we also render a high-quality synthetic hand model over various backgrounds and map it to the corresponding 3D coordinates.



Dataset

Table 1: Details of different sign language fingerspelling datasets used in this work

Database	Type	No. of classes	No. of images	Image Samples
American	Alphabets	26	15600	
Indian	Alphabets	26	4972	
Italian	Alphabets	26	12856	
American	Numbers	10	1400	
Turkey	Numbers	10	4124	

Architecture to detect hand gestures

4. Architecture

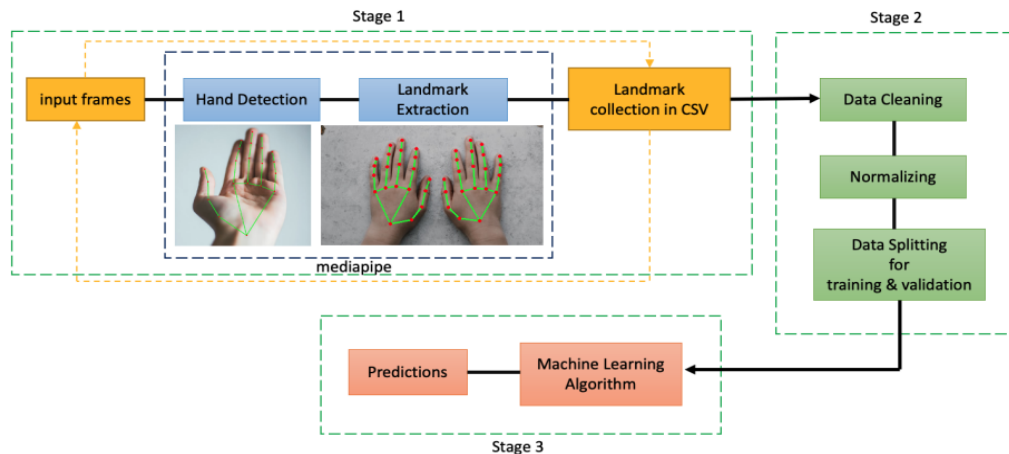


Figure 1: Proposed architecture to detect hand gestures and predict sign language finger-spellings

Activate Windows
Go to Settings to activate

Stage 1: Pre-Processing of Images to get Multi-hand Landmarks using MediaPipe

In this step we will be making hands landmarks detection model with the profound library called as mediapipe as base library and for other computer vision pre-processing CV2 library. There are many use cases in the market for this problem statement whether it's for business related virtual reality or in the gaming section for the real-time experience.

MediaPipe is a library like we already knows that framework that enables developers for building multi-modal like video, audio, any times series data cross-platform applied ML pipelines. MediaPipe library has a large collection of human body detection and also tracking models which are trained on a most diverse dataset of Google. As the skeleton of nodes and edges or landmarks, they track key points on different parts of the body. All co-ordinate points are three-dimension normalized. Models build by Google developers using Tensorflow lite facilitates the flow of information easily adaptable and modifiable using graphs. MediaPipe pipelines are composed of nodes on a graph which are generally specified in ptxt file. These nodes are connected to C++ files. Expansion upon these files is the base calculator class in MediaPipe. Just like a video stream this class gets contracts of media streams from other nodes in the graph and ensures that 12 International Journal of Research Publication and Reviews it is connected. Once rest of the pipelines nodes are connected, the class generates its own output. Packet objects encapsulating many different types of information are used to send each stream of information to each calculator. Into a graph, side packets can also be imposed, where a calculator node can be introduced with auxiliary data like constants or static properties. The simplified structure in the pipeline of dataflow enables.

The Hand tracking solution has an ML pipeline at its backend consisting of two models working dependently with each other: a) Palm Detection Model b) Land Landmark Model. The Palm Detection Model provides an accurately cropped palm image and further is passed on to the next model landmark model. This process also diminishes the use of data augmentation such as Rotations, Flipping, Scaling. Then the traditional way is to detect the hand from the frame and then do landmark localization over the current frame. But in this Palm Detector using ML pipeline challenges with a different strategy. Detecting hands is a complex procedure as you have to perform image processing and thresholding and work with a variety of hand sizes which leads to consumption of time. Instead of directly detecting hand from the current frame, first, the Palm detector is trained which estimates bounding boxes around the rigid objects like palm and fists which is simpler than detecting hands with coupled fingers. Secondly, an encoder-decoder is used as an extractor for bigger scene context.



After the palm detection is skimmed over the whole image frame, subsequent Hand Landmark models comes into the picture. This model precisely localize 21 3D hand-knuckle coordinates (i.e., x, y, z-axis) inside the detected hand regions. The model is so well trained and robust in hand detection that it even maps coordinates to partially visible hand. Figure 2 shows the 21 landmark points detection by the Hand Landmark model. Now that we have is a functional Palm and Hand detection model running, this model is passed over our dataset of various languages. Considering the American Sign Language dataset, we have a to z alphabets. So, we pass our detection model over every alphabet folder containing images and perform Hand detection which yields us the 21 landmark points as shown in Figure 2. The obtained landmark points are then stored in a file of CSV format. A simultaneous, elimination task is performed while extracting the landmark points. Here, only the x and y coordinates detected by the Hand Landmark model is considered for training the ML model. Depending upon the size of the dataset around 10-15 minutes is required for Landmark extraction.

Stage 2: Data cleaning and normalization

As in stage 1, here we are considering only x and y coordinates from the detector, in this stage each image in the dataset is passed through stage 1 to collect all the data points under one file. This file is then scraped through the pandas' library function to check for any nulls entries. Sometimes due to blurry image, the detector cannot detect the hand which leads to null entry into the dataset. we need to remove those entries in the dataset. Hence, it is necessary to clean these points or null entries otherwise it will lead to biasness while making the predictive model. And rows containing these null entries are searched and using their indexes removed from the table. After the removal of unwanted points, we normalized x and y coordinates to fit into our system. Then the data file is then prepared for splitting into two namely training and validation set. 80% of the data is retained for training our model with various optimization and loss function, whereas 20% of data is reserved for validating the model.

Stage 3: Prediction using Machine Learning Algorithm

Predictive analysis of different sign languages are performed using machine learning algorithms and Support Vector Machine (SVM) outperformed other algorithms. The details of the analysis are discussed in the result section. SVM is effective in high dimensional spaces. In the case where the number of samples are greater than the number of dimensions, SVM performs effectively. SVM is a cluster of supervised learning methods capable of classification, regression and outliers detection.

This following formula shows the optimization problem tackled by

$$\min(w, b, d) \frac{1}{2} w^T w + C \sum_{i=1}^n d_i \quad (1)$$

$$y_i (w^T \phi(x_i) + b) > 1 - d_i \quad (2)$$

In equation (1) and equation (2), d_i denotes the distances to the correct margin with $d_i \geq 0$, $i = 1, \dots, n$, C denotes a regularization parameter, $w^T w = w$ denotes the normal vector, $\phi(x_i)$ denotes the transformed input space vector, b denotes a bias parameter, y_i denotes the i-th target value. The objective is to classify as many data points correctly as possible by maximizing the margin from the Support Vectors to the hyperplane while minimizing the term $w^T w$. The kernel function used is RBF (radical basis function) that turns the input space into a higher-dimensional space, so that not every data point is explicitly mapped. Support Vector Machine works relatively well when there is a clear margin of separation between classes. Hence, we used SVM to classify multiple classes of sign language alphabets and numeric.

Stage 4 : Quantitative Analysis

In this step to analyze the results for each of this datasets, we have used performance matrix such as accuracy, precision, recall and F1 score. Accuracy is the number of correctly predicted data points out of all the data

points. $ASLR$ can be calculated as the number of all correct predictions to the total number of items in the data measures, shown in equation (3)

$$A_{SLR} = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

$$P_{SLR} = \frac{TP}{TP + FP} \quad (4)$$

$$R_{SLR} = \frac{TP}{TP + FN} \quad (5)$$

P describes how accurate our model is out of those predicted positives, how many of them are actual positive. $PSLR$ is a good measure to determine, when the cost of False positive is high. $RSLR$ calculates how many of the actual positives our model capture by labelling them as positive. $RSLR$ represent the model metric we will select when there is a high cost associated with False Negatives. The mathematical formulation of Precision and Recall are given in equation (4) and (5) respectively.

$$F_{SLR} = \frac{2 \times P \times R}{P + R} \quad (6)$$

F-Measure in equation 6 provides us a way to combine both the precision and recall into a single measure that captures both the properties. It is used to handle imbalanced classification. And Confusion matrix was also analyzed and used to have a better understanding of the types of errors being made by our classifier. The key to confusion matrix is number of correct and incorrect predictions are summarized with count values and broken down by each class.

IV. RESULT AND DISCUSSION

A K-Fold Cross-Validation was performed on the dataset by taking ten folds. The average accuracy over ten iterations of different algorithms is demonstrated in Table 2. It can be observed from the presented accuracies that SVM outperformed other machine learning algorithms such as KNN, Random Forest, Decision Tree, Naïve Bayes and also achieved higher accuracy than deep learning algorithms such as Artificial Neural Network (ANN) and Multi-Layer Perceptron (MLP).

Table 2: Average accuracy obtained using machine learning and deep learning algorithms.

DATASET	SVM	KNN	RANDOM FOREST	DECISION TREE	NAÏVE BAYES	MLP	ANN
ASL(ALPHABET)	99.15%	98.21%	98.57%	98.57%	53.74%	94.69%	97.12%
ITALIAN(ALPHABET)	99.19%	98.87%	98.59%	86.77%	96.48%	72.14%	78.63%
INDIAN(ALPHABET)	98.29%	98.87%	98.59%	98.59%	86.77%	96.48%	94.79%
TURJRY(NUMBERS)	96.22%	93.08%	94.33%	94.33%	83.64%	83.64%	93.71%
ASL(NUMBERS)	99.18%	99.18%	97.56%	97.56%	96.74%	97.56%	95.12%

The highest accuracy achieved using the model is bolded in the above table for each of the sign language datasets

Table 3: Performance analysis using SVM algorithm on different dataset

Dataset name	Training Accuracy	Testing Accuracy	Precision	Recall	F1-Score
ASL(alphabet)	99.50%	99.15%	99.15%	99.15%	99.15%
Indian(alphabet)	99.92%	99.29%	99.29%	99.29%	99.29%
Italian(alphabet)	99.72%	98.19%	98.19%	98.19%	98.19%
Turkey (numbers)	99.37%	96.22%	96.22%	96.22%	96.22%
American (numbers)	98.77%	99.18%	99.18%	99.18%	99.18%

The trained model is explicitly lightweight which makes our machine learning model appropriate for deployment in mobile application. Real-time sign language detection makes our methodology fast, robust, adaptable specifically for smart devices. Media pipe's state-of-art makes feature extraction easy by breaking down and analyzing complex hand-tracking information, without the need to build a convolutional neural network from scratch. The proposed methodology uses minimum computational power and consumes less time to train model than other state-of-arts present. Table illustrates comparison of the performance of other works of literature using machine learning / deep learning algorithms and ours.

Table 4

Sign Language	Reference	Type	Number of classes	Methods	Accuracy
American	P Das et.al.[14]	Alphabets	26	Deep CNN	94.3%
	M Taskiran et.all[150]	Alphabets and Numbers	36	CNN	98.05%
	N Saquib and A.Rahaman[16]	Alphabets	24	Random Forest	96.13%
				KNN	96.14%
				ANN	95.87%
				SVM	94.91%
	OURS	Alphabets	26	SVM	99.15%
		Numbers	10	SVM	99.18%
Indian	K.K Dutta et ..al[17]	Alphabets	24	KNN	94%-96%
	M sharma et ..al[18]	Numbers	10	KNN and neural network	97.10%
	J L Raheja et .. al[19]	Alphabets	24	SVM	97.5%
	OURS	Alphabets	26	SVM	99.29%
Italian	L.Pigou et al..[20]	Alphabets	20	CNN	91.7%
	OURS	Alphabets	22	SVM	98.19%

V. CONCLUSION

With an average accuracy of 99% in most of the sign language dataset using Media Pipe's technology and machine learning, our proposed methodology show that MediaPipe can be efficiently used as a tool to detect complex hand gesture precisely. Although, sign language modelling using image processing techniques has evolved over the past few years but methods are complex with a requirement of high computational power. Time consumption to train a model is also high. From that perspective, this work will provides us new insights into this problem. Less computing power and the adaptability to smart devices makes the model robust and cost-effective. Training and testing with various sign language datasets show this framework can be adapted effectively for any regional sign language dataset and maximum accuracy can be obtained. Faster real-time detection demonstrates the model's efficiency better than the present state-of-arts. In the future, the work can be extended by introducing word detection of sign language from videos using Media pipe and different algorithms.

VI. REFERENCES

- [1] MediaPipe.(n.d.).MediaPipeHands.mediapipe.
<https://google.github.io/mediapipe/solutions/hands.html#python-solution-api>. [8] Hassanat, Ahmad & Abbadi, Mohammad & Altarawneh.

- [2] Valentin Bazarevsky, Yury Kartynnik, Andrey Vakunov, Karthik Raveendran, and Matthias Grundmann. Blazeface: Sub-millisecond neural face detection on mobile gpus. der C. Berg. SSD: single shot multibox detector. CoRR, abs/1512.02325, 2015. 2.
- [3] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3d tracking of hand articulations using Kinect.
- [4] P. Sharma, R. Joshi, R. A. Bobby, S. Saha, and T. Matsumaru, "Projectable interactive surface using microsoft kinect v2: Recovering information from coarse data to detect touch," in 2015 IEEE/SICE International Symposium on System Integration (SII). IEEE, 2015, pp. 795–800.
- [5] <https://arxiv.org/abs/2006.10214> MediaPipe Hands: On-device Real-time Hand Tracking
- [6] <https://ijrpr.com/uploads/V2ISSUE5/IJRPR462.pdf> Real-time vernacular sign language recognition using mediapipe and machine learning.
- [7] https://www.jstage.jst.go.jp/article/nolta/13/2/13_288/_article/-char/ja/
- [8] Japanese fingerspelling identification by using MediaPipe.
- [9] <https://ijrpr.com/uploads/V2ISSUE5/IJRPR462.pdf>Real-time vernacular sign language recognition using mediapipe and machine learning.
- [10] <https://www.atlantis-press.com/article/125962696.pdf> e Cite All 5 versions
- [11] [PDF] Applying Hand Gesture Recognition for User Guide Application Using MediaPipe