Real-time Recognition of Chinese Number Hand Gestures

Daocun Yang
Stanford SCPD
daocun@stanford.edu

Abstract

This report presents the design of a system to perform real-time recognition on Chinese Number hand gestures. Specifically, a system that is able to recognize ten distinct hand gestures representing the natural numbers one through ten used in Chinese culture, with a delay of less than a second. The system combines both traditional techniques in computer vision to perform hand segmentation, as well as recent techniques using deep learning and convolutional neural network (CNN) to perform gesture recognition. The system is built upon open source technologies, including Python, OpenCV and TensorFlow. Two variations of the system will be discussed, in which one uses a simple webcam and performs recognition on RGB images, and the other uses Microsoft Kinect sensor on depth images. Although designed specifically to recognize certain hand gestures, the system can be easily adapted to recognize different hand gestures. It is also possible to port the system to less powerful computing devices, such as mobile phones. In addition, the system presented here can also serve as a gesture recognition module in a larger, more complex system which performs more advanced tasks.

1. Introduction

As a form of non-verbal communication, hand gestures play an important role in numerous applications across various fields. Some examples include human-computer interaction (HCI), control of home devices and Virtual Reality (VR) gaming devices [1]. Many such applications involve recognizing and analyzing hand gestures or signals to perform subsequent tasks.

1.1 Chinese Number Hand Gestures

This report outlines the design of a system to recognize a specific subset of hand gestures, Chinese number hand gestures. As a brief introduction, Chinese number gestures is a system of ten distinct hand gestures used in mainland China to signify the natural numbers one through ten using only a single hand. These hand gestures function to bridge the gap of the different Chinese dialects, in which different numbers can have the same pronunciation. The gestures are used primarily during bargaining and business related talks.

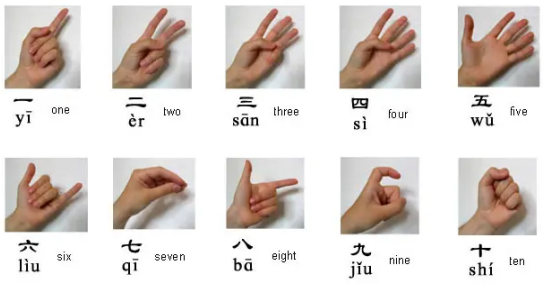

Figure 1. Chinese number hand gestures, photo courtesy of chinahighlights.com [2]

The subject of Chinese number gestures is an appropriate subset of hand gestures to perform recognition on for several reasons. First, it contains ten different gestures, making it more challenging to recognize than some simpler sets of hand gestures, such as rock-paper-scissors. Secondly, the numbers are not just represented by the number of fingers. For example, six is signified with two fingers, and ten is signified using none but a fist instead. This characteristic makes certain approaches such as finger counting ineffective. Finally, numbers such as two and eight, both of which are signified with two fingers, can have similar appearances when viewed from certain

angles. This can also pose challenges for the recognition task.

1.2 Evaluation Metrics

Since the goal is to have the system correctly recognize as many hand gestures as possible, the system will be evaluated based on the accuracy of the recognition, defined as the following:

$$Accuracy = \frac{\text{Number of hand gestures correctly recognized}}{\text{All hand gestures evaluated}}$$

2. Related Works

Numerous studies have been conducted on hand gesture recognition. In this section, I present an incomprehensive review of several major approaches for performing hand gesture recognition, which usually consists of the steps of hand segmentation and gesture recognition.

2.1 Hand Segmentation

For hand segmentation, one approach is to rely on skin color, which occupies a relatively well-defined area in color spaces such as HSV or YCbCr space. In their studies, Chai and Ngan [3] determined that skin color pixels fall into the range of Cb=[77,127] and Cr=[133,173]. Prior hand segmentation studies that make use of skin color include Rogalla et al. [4], and Hong and Yang [5]. However, skin-color based methods are not very robust, since skin color can be affected by lighting conditions, or altered significantly if the person wears gloves. In addition, when the background of the image is of a color similar to that of the skin, or contains objects of similar color, it can bring confusion to the system and lead it to mis-segment other objects as the hand.

Another approach to hand segmentation is shape-based, which uses hand features such as contours and convexity to perform segmentation. With this approach, the contour of the hand can be considered as a set of edges, and can then be obtained using traditional edge detection techniques. An example study following this approach is done by Rajam and Balakrishnan [6], who used Canny edge detection to extract the edges of the palm, and then identify finger tips based on height measuring with respect to a reference point. Similar studies include those done by Xu et al. [7], and Fiorenza et al. [8].

Hand segmentation can also be performed with the help of depth sensors and stereo video cameras. In recent years, the advent of relatively low-cost depth cameras such as Microsoft Kinect has enabled researchers to easily collect depth images. With depth images, hands can be isolated by using a depth threshold if the hands are held in front of the body for gesturing. Interested readers can refer to Suarez and Murphy [9] for a comprehensive review on hand gesture recognition using depth images.

Additionally, hands can be segmented with specialized tools and devices such as markers or colored gloves, which enable the camera to easily track and detect the region of interest (ROI) such as palm and fingertips. Past studies of this approach include those by Joslin et al. [10], Wang and Popovic [11]. However, this approach is cumbersome since it requires the person to put on special devices.

2.2 Gesture Recognition

One approach to gesture recognition uses Hidden Markov Model (HMM) which helps mitigate errors in hand segmentation. In their study, Yang and Sarkar [12] used an extended version of HMM to compute probabilities and perform recognition.

In recent years, the use of deep learning has changed the paradigm of many research fields in computer vision, including gesture recognition. Good accuracy can be achieved using Convolutional Neural Network (CNN) trained on specific dataset. Deep learning techniques are employed in related studies by Smedt et al. [13] and Kopuklu et al. [14].

3. Technical Approach

3.1 Assumptions

Before diving into the technical aspects of the system. It's important to first make clear my assumptions:

1. The system will be used in an in-door environment, which means the background of the image will be relatively simple, and the lightning condition will be neither too bright nor too dark;
2. To use the system, the person will extend his or her hand and arm forward. This ensures the hand is not at the same depth level as the rest of the body, which facilitates hand segmentation;
3. The hand that is used to perform the gesture will appear at around 0.4 meter away from the camera. If too close, the camera won't be able to capture the full hand. If too far, the hand may be out of range and cannot be detected or distinguished from the environment;
4. For each frame, the gestured hand will only appear in a specific region of interest (ROI) of equal width and height in the image frame. This assumption makes performing hand segmentation easier, as well as proportionally reshaping the image to be used in the recognition step.

3.2 System Overview

I designed the system to have three main steps. First, image data are captured, and then processed. When the image captured is a depth image, the processing stage will include a step for hand segmentation. Otherwise the image will only be resized and smoothened. After the processing step, the image will then be fed into a trained neural network for recognition. The steps are repeated for continuous gesture recognition while the system runs.

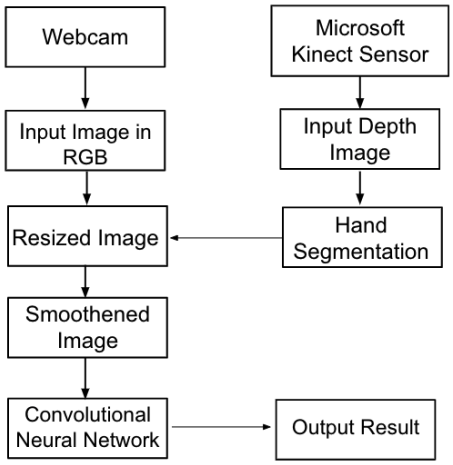

Figure 2. Block Diagram of System Architecture

Two major components in the system are the hand segmentation algorithm, and the convolutional neural network, both of which will be discussed next.

3.3 Hand Segmentation & Image Processing

After capturing depth images with depth sensor, hand segmentation will be performed to remove non-hand objects in the background. The hand segmentation algorithm is adapted from the one proposed by Beyeler [15], and contains five steps:

1. Find the median of depth values in the predefined region of interest (ROI), where the hand appears. Denote the median as d;
2. For all pixels in the captured frame, set the color of the pixel to gray if its difference between d is less than a certain deviation; otherwise set the color to black;
3. Apply morphological closing to close small holes inside the foreground object (i.e. the hand);
4. Paint the pixels that are connected to the hand to white
5. Apply thresholding to remove all non-white pixels from the image, and return the result image

Since it's assumed the hand will appear in a specific region, the median depth value in the region of interest will be the depth of the hand. The next step is to set the color of all pixels that

are at the same depth as the hand to gray. This will also include pixels that potentially belong to objects in the background, which are at the same depth level as the hand. The choice of gray is rather arbitrary; the only requirement is to ensure the color is neither black nor white, so that we can distinguish the pixels from black and white pixels in the depth image.

The next step is to apply morphological closing, which is a type of morphological transformations technique commonly used in computer vision. Morphological closing is two basic morphological operations performed in sequence: dilation and erosion. Both operations require two inputs, one is the image to operate on, and the other a structuring element or kernel. To operate, the kernel slides through the image as in 2D convolution. For dilation, a pixel in the original image will be set to 1 if at least one pixel under the kernel is 1; for erosion, all pixels under the kernel need to be 1 for the original pixel to be set to 1. Morphological closing helps remove the small holes inside the segmented hand in the foreground.

After morphological closing, I then set the pixels that belong to the hand to white. Since we know the region where the hand appears in the image frame, the colors of those pixels are set, along with the connected pixels of those pixels to white. For this task, the *floodFill* method implemented in OpenCV comes in handy. Lastly, thresholding was applied to all pixels in the image, and thus removing the gray pixels that belong to potential irrelevant objects which share the same depth as the hand.

Figure 3. Depth image before (Left) and after (Right) hand segmentation

For both RGB and depth input images, resizing is needed to transform the images into shapes that are expected by the neural network, which will be discussed in the next section.

Additionally, I further smoothed the resized image with a bilateral filter. Bilateral filter is similar to the 2D Gaussian convolution, where the Gaussian weighted average is computed for the neighbourhood around a pixel. Unlike Gaussian filter, bilateral filter also takes into account the variation of intensities to preserve edges while removing noise. Denoted by BF[.], the bilateral filter is defined by:

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|)\, G_{\sigma_r}(I_{\mathbf{p}} - I_{\mathbf{q}})\, I_{\mathbf{q}}$$

which combines the normalization factor Wp, space parameter σs and range parameter σr.

3.4 Hand Gesture Recognition

To recognize the ten Chinese number hand gestures, I modeled the recognition task as a multi-class image classification problem. The recognition unit will output one of eleven classes, with ten of the classes corresponding to the ten numbers, and an additional class corresponding to invalid or no-hand scenarios.

A convolutional neural network (CNN) is used to perform gesture recognition on image input. When choosing the architecture of the neural network to be used by the system, several considerations were taken into account:

1. Training the neural network should be relatively fast;
2. The network should have relatively fewer parameters, while maintaining good accuracy on recognition tasks;
3. The network architecture should allow recognition tasks to be performed in a timely fashion, which is especially important given the real-time nature of the system.

After some research, I decided to adopt MobileNet [16] and its later variants such as MobileNetV2 [17] and EfficientNet [18] for the image classification task, which I believe satisfy the requirements listed above.

The key characteristic that distinguishes MobileNet from other neural network architectures

is the incorporation of depthwise separable convolution to reduce computation. Depthwise separable convolution consists of two layers: depthwise convolutions and pointwise convolutions, which together replace the standard convolution filters.

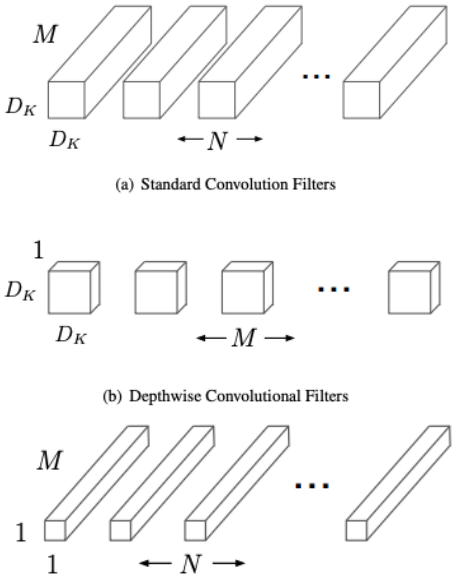

Figure 4. Depthwise separable convolution, photo courtesy of Howard et al.

With depthwise separable convolution, the standard convolution operation is transformed into two addition operations, which results in a reduction in computation. The network can also be made more compact with width multiplier and resolution multiplier. MobilenetV2 further incorporated the idea of residual blocks into the depthwise separable convolution layer, and reduced the number of parameters.

As a more recent study, EfficientNet explored the idea of simultaneously scaling the width, depth and input resolution of a base model to achieve state-of-the-art performance. The base model EfficientNetB0 was built primarily using Inverted Residual Blocks (MBConv), which was introduced in MobileNetV2.

4. Experiments

To capture images from a video stream, two types of cameras are used. For capturing RGB image input, the built-in camera of Apple MacBook Pro (Retina, 13-inch, Early 2015) was used. For capturing depth images, the device being used was a Microsoft Kinect Sensor v1.

Figure 5. Experiment setup. A Microsoft Kinect v1 sensor (left) connected to Apple MacBook Pro (right).

4.1 Microsoft Kinect Sensor

First introduced in November 2010, Kinect is a line of motion sensing input devices produced by Microsoft [19]. Kinect is a light-coded range camera capable of capturing depth images as well as RGB images at 30 frame-per-second (fps) with a resolution of 640 × 480. The depth sensor consists of an infrared laser projector and a monochrome CMOS sensor, which allows it to capture information from 3D scenes.

4.2 Data Collection

To perform recognition using a neural network, it is needed to train the network using images containing Chinese number hand gestures. Due to a lack of prior studies on this particular subject, I built a dataset by capturing images of my own hand gestures as both RGB and depth images.

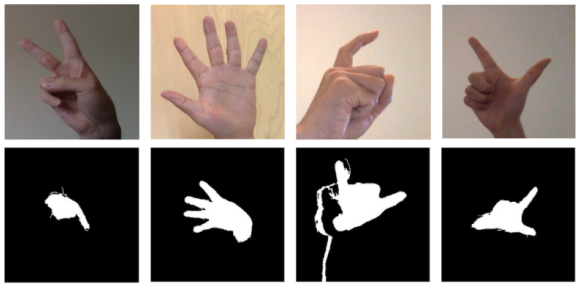

Figure 6. Custom dataset of RGB and depth images posing Chinese number hand gestures

For each class of number, around 110 RGB and depth images were collected, respectively. Each image is of size 224 by 224 pixels. Each class of number gestures was performed with both my left and right hand. To increase the robustness of recognition, variations were added during the image collection process. For RGB images, variations include different lighting conditions and different backgrounds. For depth images, certain noisy images were included. Noise in depth images refers to parts that don't belong to the hand, which could be caused by a combination of imperfect hand segmentation and noise in the sensor measurements. In addition, hand orientations and angles were altered during the capture of both RGB and depth images.

The final datasets contain 1222 RGB images and 1205 depth images. Having collected the data, I then split the images into train, validation and test sets with a ratio of 77%, 18%, and 5%.

4.3 Training the Neural Networks

To simplify the training of neural networks and reduce training time, I applied transfer learning and took the weights from models trained on the ImageNet dataset, and then fine-tuned the network to fit the hand gesture dataset using TensorFlow. Training was done using Google Colab with GPU runtime.

Several modifications were needed to fit the custom hand gesture dataset. With MobileNet and MobileNetV2, I first modified the last few layers of the neural network, and replaced the layers after the global average pooling layer found in both network architecture with a softmax layer to output the 11 classes.

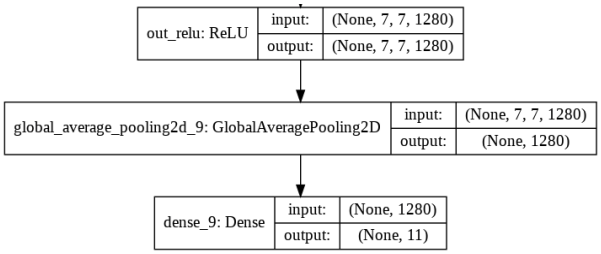

Figure 7. The last layers of modified MobileNetV2.

Next, several neural networks were trained with different parameters. One parameter to tune is the number of layers that are trainable in the network. Through experiments, I discovered that it's best to unfreeze and train the final 20 to 25 layers for MobileNetV1 and MobileNetV2 to achieve good accuracy. In addition, I also experimented with some common values for the width multiplier α, which further thins the network. For optimization, an Adam optimizer with a learning rate of 0.0001 is used.

With MobileNetV2, a test accuracy of 0.946 and 0.964 is achieved on depth and RGB images, respectively. With the MobileNet architecture, an accuracy of around 0.95 can usually be achieved after 30 epochs of training. Training is also fast, with each epoch taking around 5 seconds.

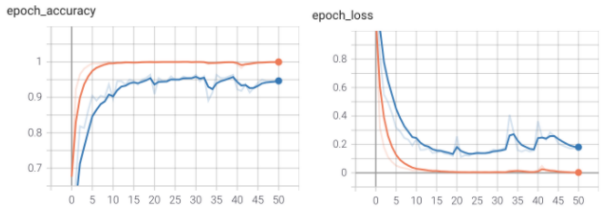

Figure 8. Change in accuracy (left) and loss (right) during the training of a MobileNetV2 network on depth images.

For experimentations with EfficientNet, three networks of different variants were trained, which include EfficientNetB0, EfficientNetB4, and EfficientNetB7. From the perspective of training the networks, the difference between these three variants is the resolution of the input image, which is 224 × 224, 380 x 380, and 600 x 600 respectively. Only depth images from the dataset were used to train on EfficientNet, after being scaled first to meet the input shape requirements.

Similar to the modifications to MobileNets, the final layers of each EfficientNet model were replaced by a softmax layer, and also a global average pooling layer and a batch normalization layer. A drop out rate of 0.2 was applied to prevent overfitting.

The EfficientNet models were trained in two steps. First, I trained the model for 60 epochs with all layers frozen except the final added layers; a learning rate of 0.001 was used. Next, the final 20 layers, except the BatchNorm layer, were unfrozen and trained.

4.4 Results Analysis

Overall, the trained MobileNet neural networks were observed to have very good accuracy on recognizing the number gestures. When the inputs are RGB images, it even achieved a perfect accuracy of 1. It is worth noting that the perfect accuracy is achieved with a width multiplier less than 1, which means the parameters are further reduced from the base model. I believe this is because the RGB images are less noisy, which results in less variations in hand shapes. In this scenario, an alpha value of 1 may lead to slight overfitting, while $\alpha = 0.5$ may slightly underfit.

	Width Multiplier	Cross-entropy Loss	Accuracy
MobileNetV1	1	0.025	0.982
MobileNetV1	0.75	0.034	1.000
MobileNetV1	0.5	0.053	0.982
MobileNetV2	1	0.050	0.982
MobileNetV2	0.75	0.012	1.000
MobileNetV2	0.5	0.04	0.982

Table 1. Classification results on RGB images

When testing on depth images, the two base models where $\alpha = 1$ led to the best result. This is expected since the depth images are more noisy, and contain parts that don't actually belong to the hands; therefore more parameters can be helpful in obtaining good accuracy.

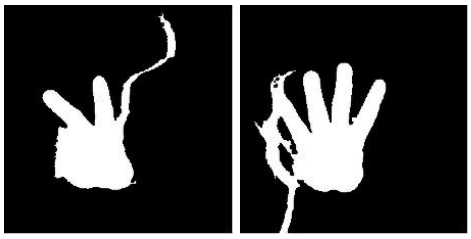

Figure 8. Samples of noisy depth images used in testing, representing number 2 (left) and number 4 (right).

	Width Multiplier	Cross-entropy Loss	Accuracy
MobileNetV1	1	0.083	0.982
MobileNetV1	0.75	0.322	0.909
MobileNetV1	0.5	0.105	0.946
MobileNetV2	1	0.128	0.946
MobileNetV2	0.75	0.280	0.946
MobileNetV2	0.5	0.146	0.946

Table 2. Classification accuracy on depth images

As for EfficientNet, the results obtained were surprisingly far from satisfactory. Among the three models of EfficientNetB0, EfficientNetB4, and EfficientNetB7, the best test accuracy obtained on depth images was a mere 0.273. I believe the poor result could be due to several reasons. First, given the time constraint, I only trained each model for 130 epochs with two the two steps combined. More training epochs is needed given the increased complexity of the models. Secondly, certain important parameters may need to be tuned further, such as the number of layers to unfreeze, learning rate, etc..

	Total Layers (modified)	Total Params	Trainable Params (combined)
MobileNet V1	88	3.24M	1.87M
MobileNet V2	156	2.27M	1.07M
EfficientNetB0	241	4.07M	1.15M
EfficientNetB4	478	17.7M	2.68M
EfficientNetB7	817	64.1M	5.42M

Table 3. Model Summary of Keras Implementations. Total layers is counted after modifying the final layers. For EfficientNet models, the number of trainable parameters are combined from the two training steps.

Eventually, I chose to use MobileNetV2 as the implementation for gesture recognition. With MobileNetV2, each inference only takes around 0.04 second for an image. Although the inference is not fast enough to be performed for each frame (30 fps for the Kinect sensor), it satisfies the requirement of less than one second of delay. For

simplicity, a frame is captured every 0.5 second by the Kinect sensor, went through hand segmentation and processing, and then sent to the trained MobileNetV2 classifier for classification. The system was able to output the result timely and accurately.

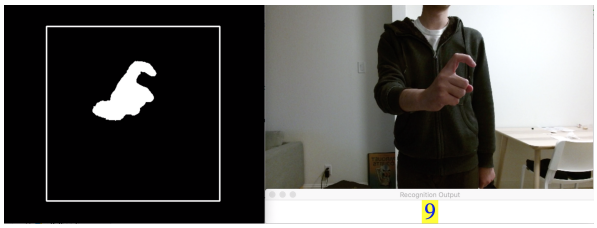

Figure 9. Real-time recognition of Chinese Number Gestures.

5. Conclusion

A system was built successfully to recognize Chinese number hand gestures in real-time in an indoor environment. To recognize hand gestures, the system combined both traditional computer vision techniques such as morphological transformations and smoothing to first perform hand segmentation, and a convolutional neural network trained on a custom dataset to perform gesture recognition.

The study showed that using lightweight neural network architecture such as MobileNetV1 and MobileNetV2 results in a good classification accuracy of around 95% and a fast interference time of around 0.04 second. From the results, it is worth noting that MobileNetV2, while having fewer parameters, is able to perform at a comparable accuracy as MobileNetV1. The experiments with EfficientNet also showed that complex models aren't always the best choice for all image classification tasks. In this case, simpler models work better for my particular dataset.

For future work regarding image classification, some tasks include to study fine-tuning in more details, to spend more time tuning and training EfficientNet models, and also to come up with custom neural network architecture suitable for this particular gesture recognition task. It will also be worthwhile to experiment with MobileNetV3, which may give us more interesting results and insights.

The study also showed that hand segmentation can be done relatively easily with commercial depth sensors such as Microsoft Kinect and open-source frameworks such as OpenCV. However, more research on hand segmentation algorithms is needed to further improve the quality of hand segmentation and to reduce noise. Alternatively, one can try running the same segmentation algorithm with different depth sensors such as newer versions of Kinect and compare the results.

To make the system more robust, more data with variations could be collected. The current dataset only contains images of my own, naked hands. It can be worthwhile to collect images of other people's hands, and hands that wear gloves, rings or other objects, which could alter the shape and size of the hand.

Additionally, it can be interesting to work on extensions of the current system. Since MobileNet architecture was designed to be run on devices with limited computational power, one should be able to run the existing recognition system on mobile devices, and perform real-time gesture recognition using the built-in RGB camera of the mobile device. One can also extend the system to be used in real-world scenarios, and incorporate techniques from fields such as the Internet of Things (IoT) to build interesting applications. For example, to use number gestures for home automation tasks such as controlling the light, the TV, and so on.

The code of this project can be found at: https://github.com/daocunyang/Chinese-Number-Gestures-Recognition.

References

[1] Juan Pablo Wachs, Mathias Kölsch, Helman Stern, and Yael Edan. 2011. Vision-based hand-gesture applications. Commun. ACM 54, 2 (February 2011), 60–71.

[2] *Learning Chinese - Numbers in Chinese.* China Highlights. https://www.chinahighlights.com/travelguide/learning-chinese/chinese-number.htm
[3] Chai D, Ngan K.N, “Locating Facial Region of a Head-and-shoulders color image,” In :Proceedings of the 3rd International Conference on Automatic Face and Gesture Recognition. Nara, Japan, pp.124~129, 1998.
[4] Rogalla, O., Ehrenmann, M., Zöllner, R., Becher, R., and Dillmann, R. Using gesture and speech control for commanding a robot assistant. In *Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication* (Berlin, Sept. 2527, 2002), 454459.
[5] Duan, Hong & Luo, Yang. (2013). A Method of Gesture Segmentation Based on Skin Color and Background Difference Method. 10.2991/iccsee.2013.90.
[6] Rajam, Subha & Ganesan, Balakrishnan. (2011). Real time Indian Sign Language Recognition System to aid deaf-dumb people. International Conference on Communication Technology Proceedings, ICCT. 10.1109/ICCT.2011.6157974.
[7] Xu, Y., Park, D., & Pok, G. (2017). Hand Gesture Recognition Based on Convex Defect Detection.
[8] C. Fiorenza, S.K. Barik, A. Prajapati, S. Mahesh.
. (2019).Hand Gesture Recognition using Convexity Defect. International Journal of Innovative Technology and Exploring Engineering (IJITEE)
[9] Soto, Carlos & Murphy, Robin. (2012). Hand gesture recognition with depth images: A review. Proceedings - IEEE International Workshop on Robot and Human Interactive Communication. 411-417. 10.1109/roman.2012.6343787.
[10] Joslin, C., El-Sawah, A., Chen, Q., & Georganas, N. (2005). Dynamic Gesture Recognition. *2005 IEEE Instrumentationand Measurement Technology Conference Proceedings, 3*, 1706-1711.
[11] Wang, R.Y.; Popovi´c, J. Real-time hand-tracking with a color glove. ACM Trans. Graph. 2009, 28, 1–8.
[13] Q. De Smedt, H. Wannous, J.-P. Vandeborre, J. Guerry, B. Le Saux, and D. Filliat. 2017. 3D hand gesture recognition using a depth and skeletal dataset: SHREC'17 track. In Proceedings of the Workshop on 3D Object Retrieval (3Dor '17). Eurographics Association, Goslar, DEU, 33–38.
[14] O. Köpüklü, A. Gunduz, N. Kose and G. Rigoll, "Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks," *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*, Lille, France, 2019, pp. 1-8
[15] Michael Beyeler. (2015). OpenCV with Python Blueprints. Packt Publishing
[16] Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *ArXiv, abs/1704.04861*.
[17] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4510-4520.
[18] Tan, M., & Le, Q.V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *ArXiv, abs/1905.11946*.
[19] Wikipedia contributors. (2021, March 10). Kinect. In *Wikipedia, The Free Encyclopedia*. Retrieved 06:48, March 20, 2021, from https://en.wikipedia.org/w/index.php?title=Kinect&oldid=1011388970