Real-Time Hand Gesture Recognition in Video Control Human-Machine Interaction Using Camera

Nguyen Thanh Long, Dang Hoang Phong, Nguyen Minh Duc, Chu Dang Huong
{longnthe160739, phongdhhe160286, ducnmhe160568, huongcdhe163236} @fpt.edu.vn

Advisor: Do Thai Giang

Abstract: Nowadays working with computers is a common task. In most cases, the keyboard and mouse are the primary input devices. However, people express emotions, act through gestures because it feels the most natural method to communicate. Compared to external devices like keyboards and mice, movements feel more natural. Thus, the idea and goal of the project is to use hand gestures to operate and control multimedia players and electronic devices. In this thesis, a hand gesture recognition system capable of controlling media players on computers, and controlling electronic devices is developed and provided. Hand gestures are important for interacting with our system. We used our hand gesture recognition to perform media player, such as volume down/increase, next music, etc.

Keywords: Hand gestures recognition, human-machine interaction, deep learning, neural network, video.

1. Introduction

Hand gestures recognition plays an important role in human-computer interaction. Hand gestures recognition taking note the fingers position, move, the shape constructed by the hand when interacting with the surroundings. Common gestures recognition based computer applications include those for interacting with translation sign language, driving detection, gaming control, home smart application, monitoring the behavior of disabled individuals.

There are two types of hand gestures: static and dynamic. The static gesture relates to the stable shape of the hand, whereas the dynamic gesture is made up of a sequence of hand motions like clapping or waving. There are two approaches are generally used to interpret gestures for human-machine interaction application. The first approach is based on data gloves and the second approach is based on computer vision. In the proposed method, the hand gesture is detected from the primary image using OpenCV and MediaPipe. Then used skin color (HSV color space) technique. After, the data is fed into CNN model MobileNetV2 for classification, the results return 6 numbers from 0 to 5 corresponding to 6 classes. And then we bind these numbers for various functionalities to control the media

player or YouTube.

2. Related Works

In 2022, Zheng, Zepei [1] proposed and studied a potential and concept to recognize human gestures, as well as increase the amount of information provided through image analysis to enhance human interaction. and computer through algorithms HHM, DTW, SVM, ANN. This study looks at computer vision-based gesture recognition, which has overcome the limitations of wearables after nearly two decades of research. However, issues, including about 30% poor uniformity, sensitivity to light changes as well as congestion, and unreliable real-time performance persist. Certainly, increasing the computing speed of the computer can alleviate the unreliable performance challenge of real-time detection.

In 2011, D. Kumarage, Sohan Fernando, P. Fernando, D. Madushankan [2] presented a less expensive approach to develop computer vision-based sign language recognition applications in a real-time context with motion recognition. They explored new concepts of breaking down motion gestures into sub-components for parallel processing and mapping of motion data to static data representations. This concept can be used to define sign language gestures without performing computationally intensive tasks for each and every frame captured. Alternatively, sign language gestures can be evaluated with minimal image processing and motion mapping into linear/non-linear equations using the functions proposed in this news. Index terms-Sign Language Recognition, American Sign Language, Motion Recognition, Computer Vision, Gesture Recognition. From this project, we can see the construction process, how it works and its results, the rate when using RF algorithm is 91~100%, the rate of using SVM and MLP also gives results. similar and quite high and accurate.

In 2022, Schlüsener, N. &Bücker, M. [3] studied few-short learning model by sign language, a group of researchers have researched and created this new learning model. They have developed models of Learning less often trained to recognizes five or ten different dynamic hand gestures that are, respectively, interchangeable at will by providing the model with one, two, or five examples for each hand gesture. All models are built in the Relational Network (RN) Less-Learning architecture, where Short-Term Long-Term Memory cells form the backbone. Models using hand reference points extracted from the RGB video sequence of the Jester dataset were modified to accommodate 190 different types of hand gestures. The results show an accuracy of up to 88.8% for year recognition and up to 81.2% for ten dynamic hand gestures. The study also sheds light on the potential for effort savings when using the Lesser Learning approach instead of the traditional Deep Learning approach to detect dynamic hand gestures. Savings are defined as the number of additional observations required when the Deep Learning model is trained on new hand gestures instead of the Learning model several times. The difference to the total number of observations required to achieve approximately the same accuracy suggests a potential savings of up to 630 observations for five and up to 1260 observations for ten recognized hand gestures. Since labeling hand gesture video recordings implies considerable effort,

these savings can be considered substantial.

In 2020, Dinh-Son, Tran. Ngoc-Huynh, Ho. Hyung-Jeong, Yang. Eu-Tteum, Baek, Soo-Hyun, Kim. & Gueesang, Lee [4] introduced a new method in real-time hand gesture recognition. The original method is to define the fingertips on the contour. They used Microsoft's Kinect RGB (RGB-D) depth camera combined with the 3D convolutional neural network. This system has overcome the confounding factors, overcome the limitation of distance difference. Tested in identifying six hand gestures in different environments. Delivers highly accurate results (more than 90%) in hand gesture recognition that surpasses basic methods: traditional machine learning models—support vector machines (SVMs) and bidirectional convolutional neural networks (2DCNN). However, it requires significantly higher training time.

In 2019, Adam Ahmed Qaid, MOHAMMEDORCID. Kien Thanh, Lv & MD. Sajjatul, Islam [5] publish a deep learning-based design architecture to incorporate the detection and recognition of static hand gestures. By training a single phase loss of focus dense object detector for hand detection, namely Retina Net, and using Mobile Net-based light convolutional neural network (CNN) for gesture recognition. Helped to reduce resource consumption and increase the performance of the architecture. Experiments compare the accuracy of hand detection and the high accuracy of hand gesture recognition results compared to previous methods, especially in highly cluttered environments, while still being able to maintain high speed. 12 FPS on GPU, meeting basic needs for gesture applications.

In 2023, Mahmoud, Elmezain. Majed, M. Alwateer. Rasha, El-Agamy. Elsayed, Atlam & Hani, M. Ibrahim [6] published research on the integration of Hidden Markov Models (HMMs) and Deep Neural Networks (DNNs), providing an automated technique that simultaneously detects and predicts meaningful hand gestures without there is a delay. By processing and recognizing by determining the start and end points of continuous hand movements, this model does not need to use training data. According to the experimental results, the proposed method can successfully detect and forecast essential fluctuations with a confidence level of 94.70%. However, this model has limitations on distance and reduces the accuracy with subjects to direct noise.

3. Data Preparation

After fine-tuning pre-trained models, we generate the dataset ourselves (**IHS** – the *Image Hand Sign* dataset) using OpenCV to capture the video and MediaPipe to detect the hand from the video. When hand is detected, image frames are taken from the video then the hand is cropped from the image to reduce noise information from the background. Then the image is converted to the HSV color space. The steps of data preparation are,

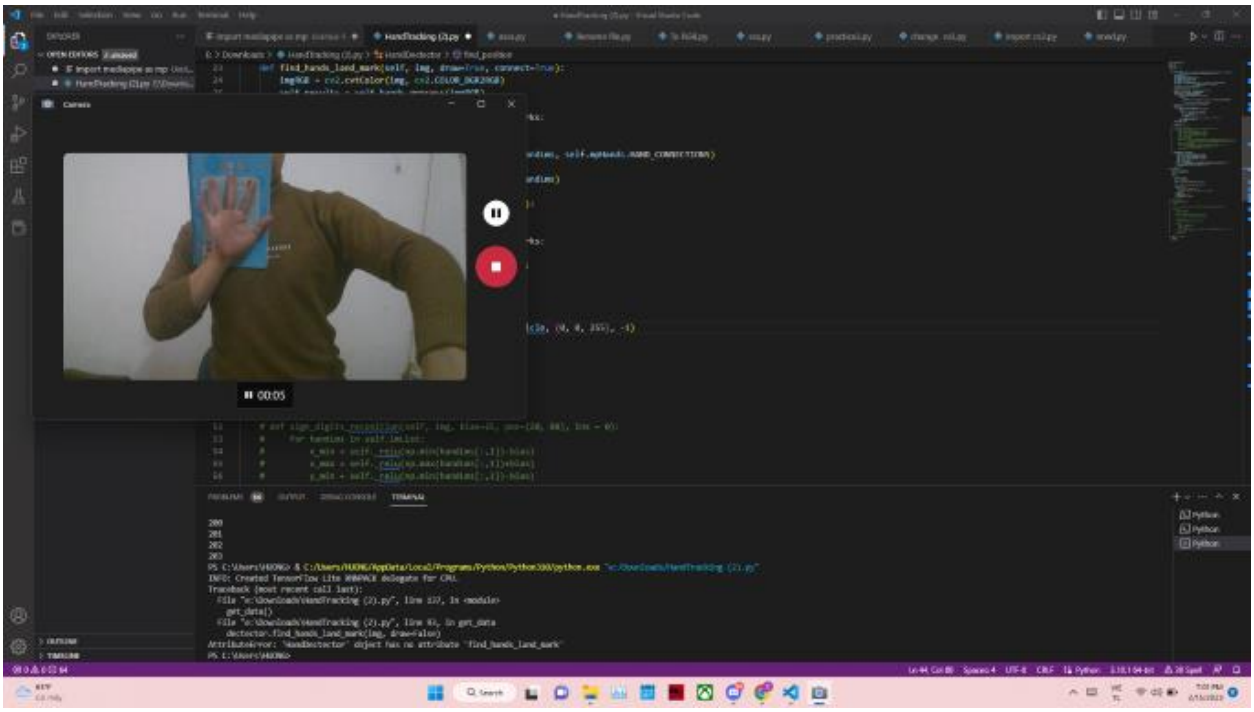

Figure 2.1. Capture video with OpenCV.

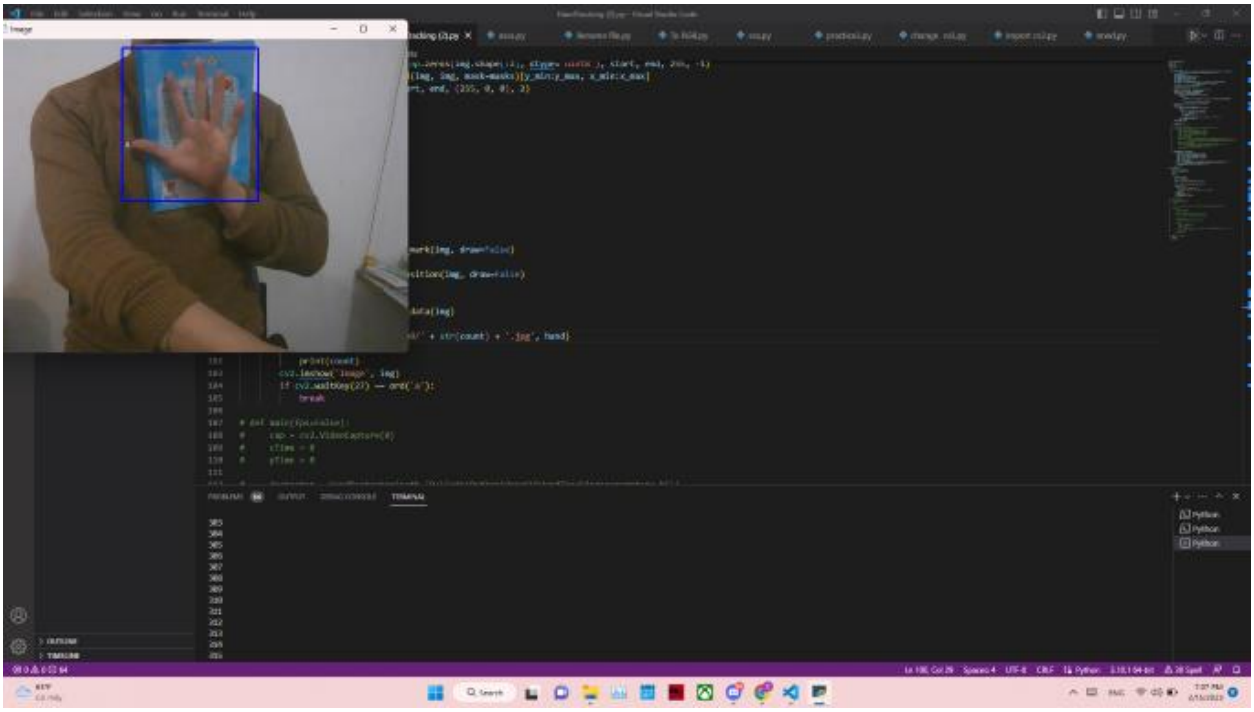

Figure 2.2. Detect hand with MediaPipe.

Figure 2.3. Crop the hand and change to the same size.

The result of our data preparation process is a dataset with 6600 samples, containing 6 different classes. Each class combines two original hand gestures (left hand and right hand) with ratio are 50% - 50%.

Table 1. Overview on our data

Type	Samples	Left - Right (Hand)
Class 0	1100	50% - 50%
Class 1	1100	50% - 50%
Class 2	1100	50% - 50%
Class 3	1100	50% - 50%
Class 4	1100	50% - 50%
Class 5	1100	50% - 50%

4. System Design

We proposed system architecture, the real-time hand gesture input from the user through the integrated webcam is used by the hand gesture recognition system on controlling media player, and it works when the user's inputted gesture matches one that can control the media player. The model offers the fundamental controls for the media player, including play, pause, volume adjustment. To develop the application, we employ a variety of Python tools and modules, including PyAutoGUI, OpenCV, MediaPipe, Tensorflow and subprocess. The system architecture diagram of a proposed model is,

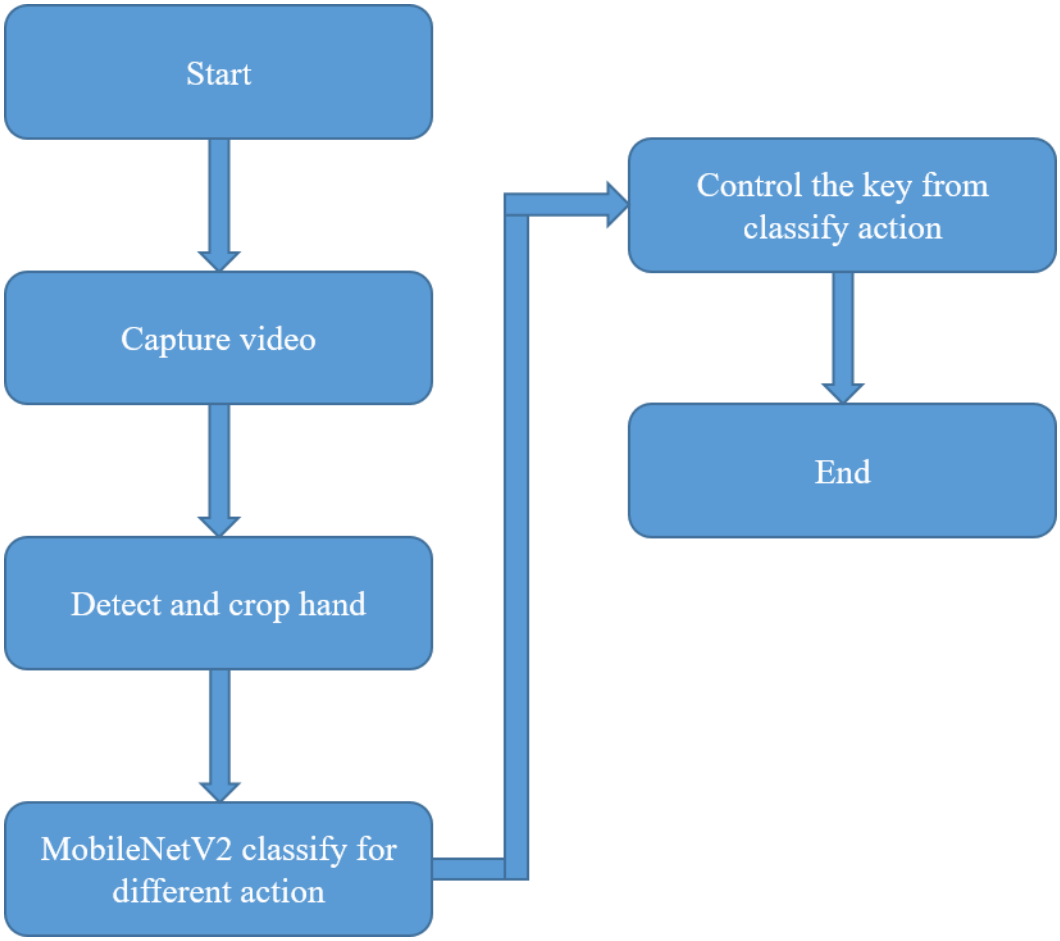

Figure 4.1. System architecture.

The image frames are obtained from the video feed, then the hand is detected from the image and cropped to convert from BGR to HSV image using library function called cvtColor. The image is then fed into the MobileNetV2 [7] model for classification.

MobileNetV2

MobileNetV2 [7] is a CNN architecture (Figure 4.2) designed for mobile devices with limited computational resources. It is an improvement over the original MobileNet architecture, with better performance and fewer parameters. There are 5 main components of MoblieNetV2 (Depthwise Separable Convolution, Inverted Residuals, Linear Bottlenecks, Global Depthwise Pooling, Squeeze-and-Excitation Module). The input to MobileNetV2 is a 3-channel RGB image with a fixed size. The most commonly used input size is 224x224 pixels, but other sizes can also be used. For example, larger input sizes such as 256x256 or 299x299 pixels may be used for higher resolution images or when more fine-grained details need to be captured. On the other hand, smaller input sizes such as 128x128 pixels may be used to reduce the computational cost and memory usage of the

model, especially when running on mobile or embedded devices. The input image is fed into the network as a tensor of shape (batch_size, 224, 224, 3), where batch_size is the number of images being processed at once. The output of MobileNetV2 is a tensor of shape (batch_size, num_classes), where num_classes is the number of output classes for the specific task. The values in this tensor represent the network's confidence in each class for each input image in the batch.

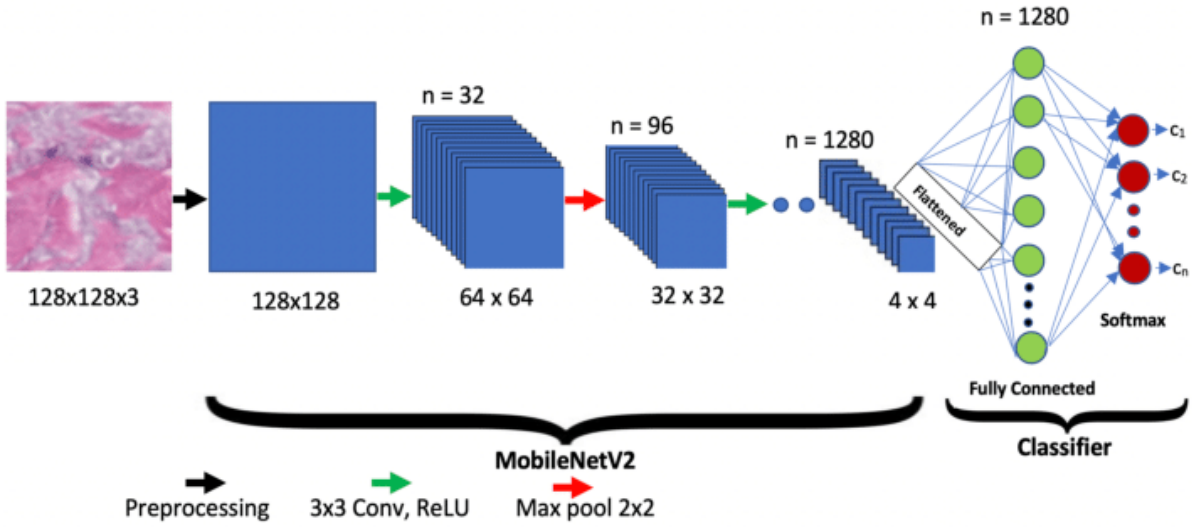

Figure 4.2. MobileNetV2 network architecture.

MobileNetV2 is trained to be able to predict 1000 different classes. So to use MobileNetV2 for our task we need to make some changes on top layer. We removed the top 2 layers Dense (predict 1000 classes) and GlobalAveragePooling2D. And we add 3 layers GlobalMaxPooling2D, Dropout, Dense (predict 6 classes). Then we started to fine-tune the model on dataset.

After classify the defects for different actions, we have used PyAutoGUI to control the keys, map gestures to keys. i.e., to automatically press the keys according to conditions.

- Prepare to receive orders - 0 defect
- Skip forward - 1 defect
- Skip back - 2 defect
- Volume up - 3 defect
- Volume down - 4 defect
- Play/Pause - 5 defect

5. Implementation

We use TensorFlow(2.11.0) for model training and data processing, Keras(2.11.0) for defining model architecture, model evaluation. After 9 epoch of training in 1 hour on

Google Colab with 12GB RAM, we get 98.66% accuracy – 0.0536 loss in validatiton dataset and 97.53% accuracy – 0.0691 loss in training dataset.

Figure 5.1. Training and Validation loss

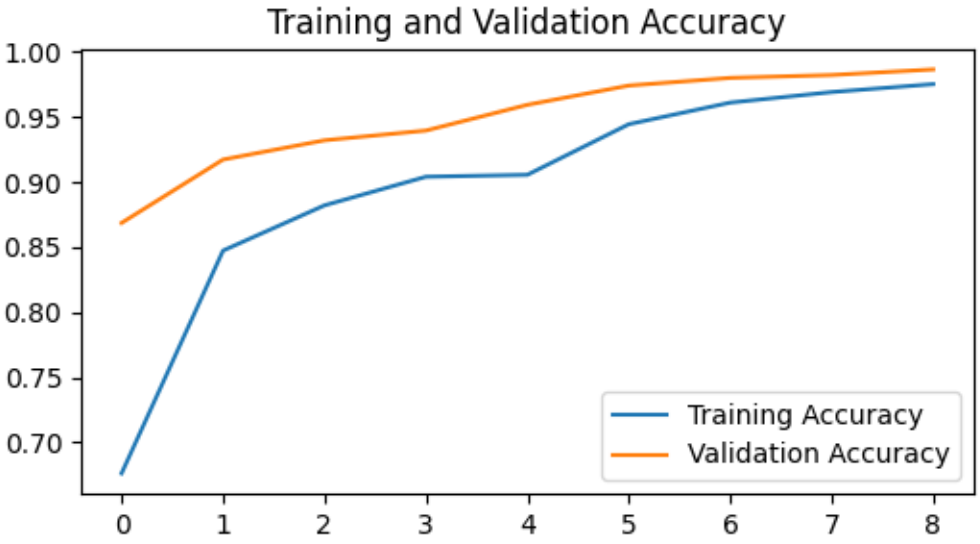

Figure 5.2. Training and Validation accuracy

A real time video is captured using OpenCV library and camera. The process involves selecting a rectangular portion of the captured image which is then filtered by a mask to extract the image of the hand. Then model is use to detect the number of fingers show by user then we can bind the count of number defect with the key to control media.

Resultant outputs:

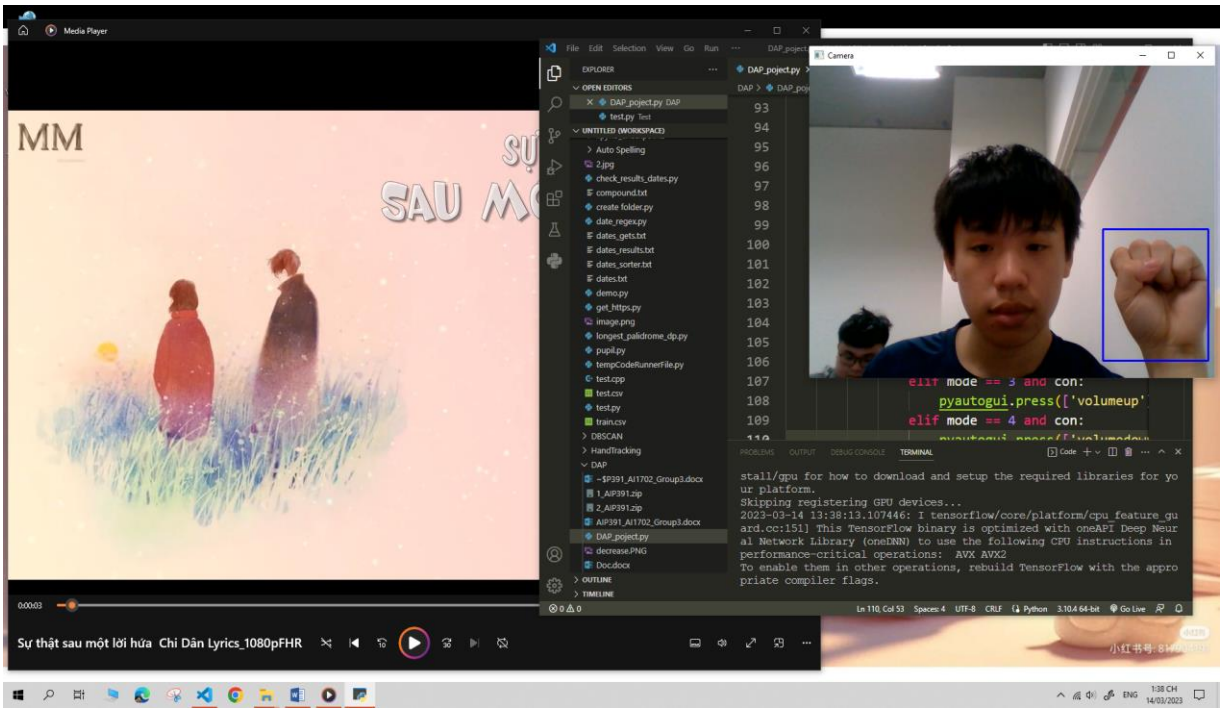

Figure 5.1. Prepare to receive orders

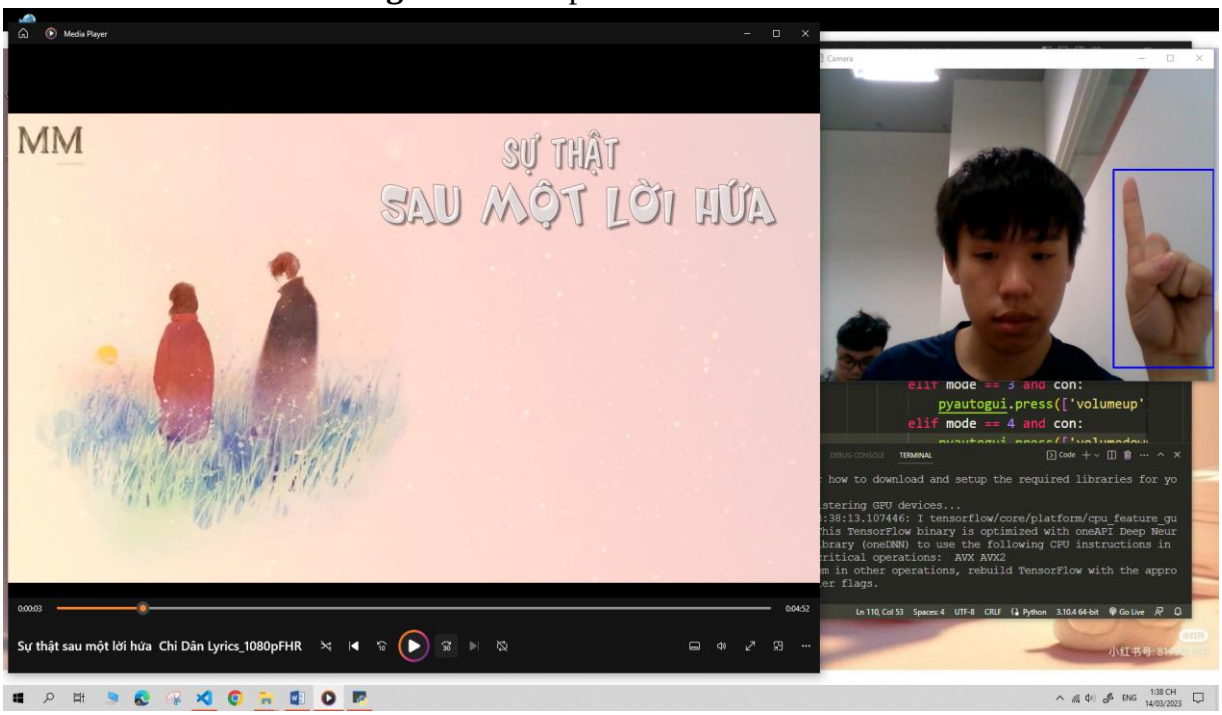

Figure 5.2. Skip forward

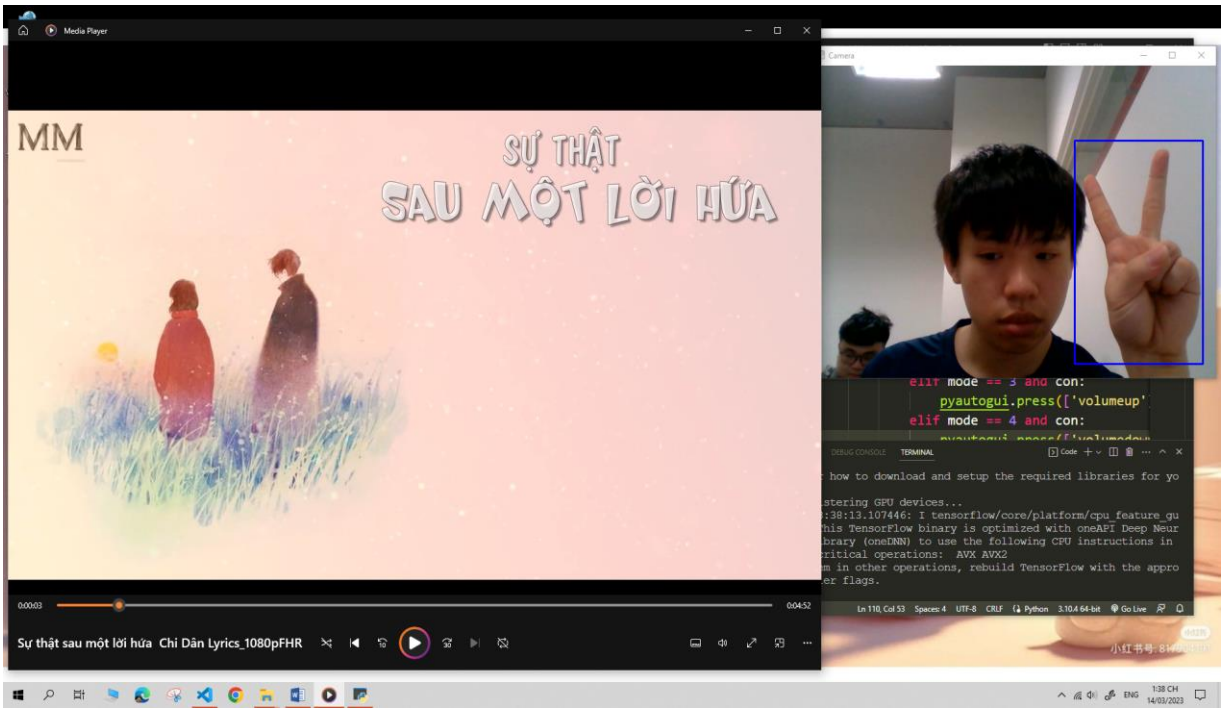

Figure 5.2. Skip back

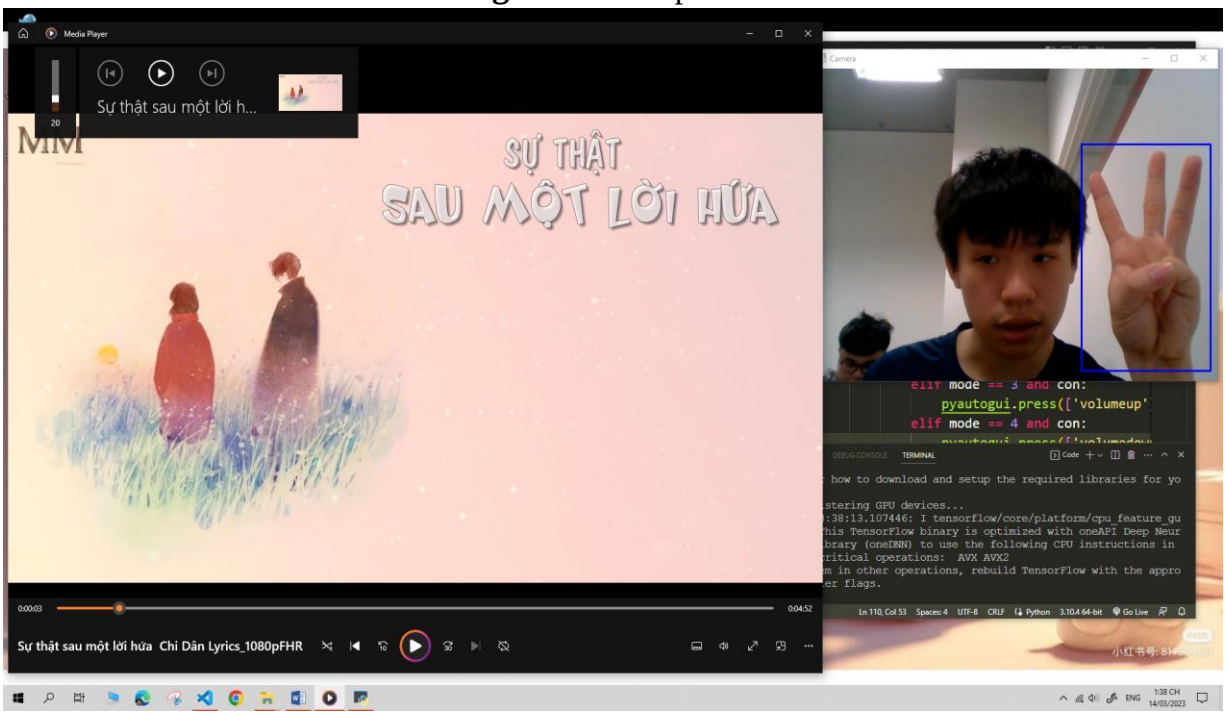

Figure 5.3. Volume up

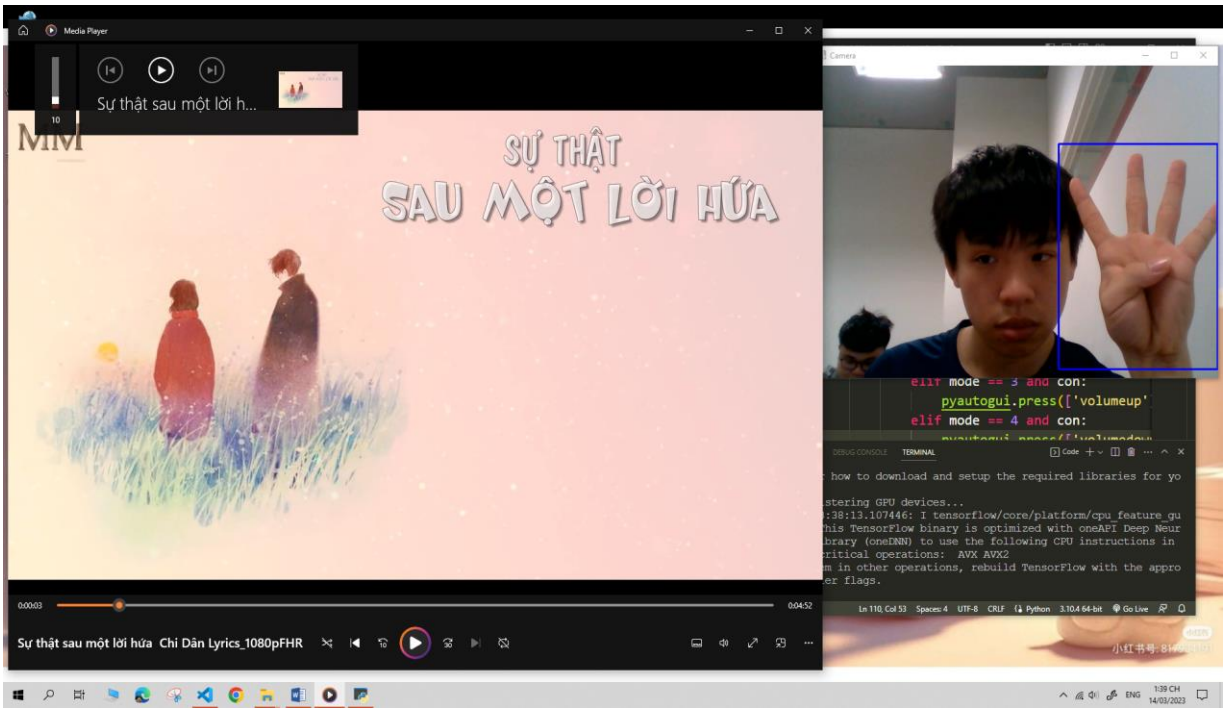

Figure 5.4. Volume down

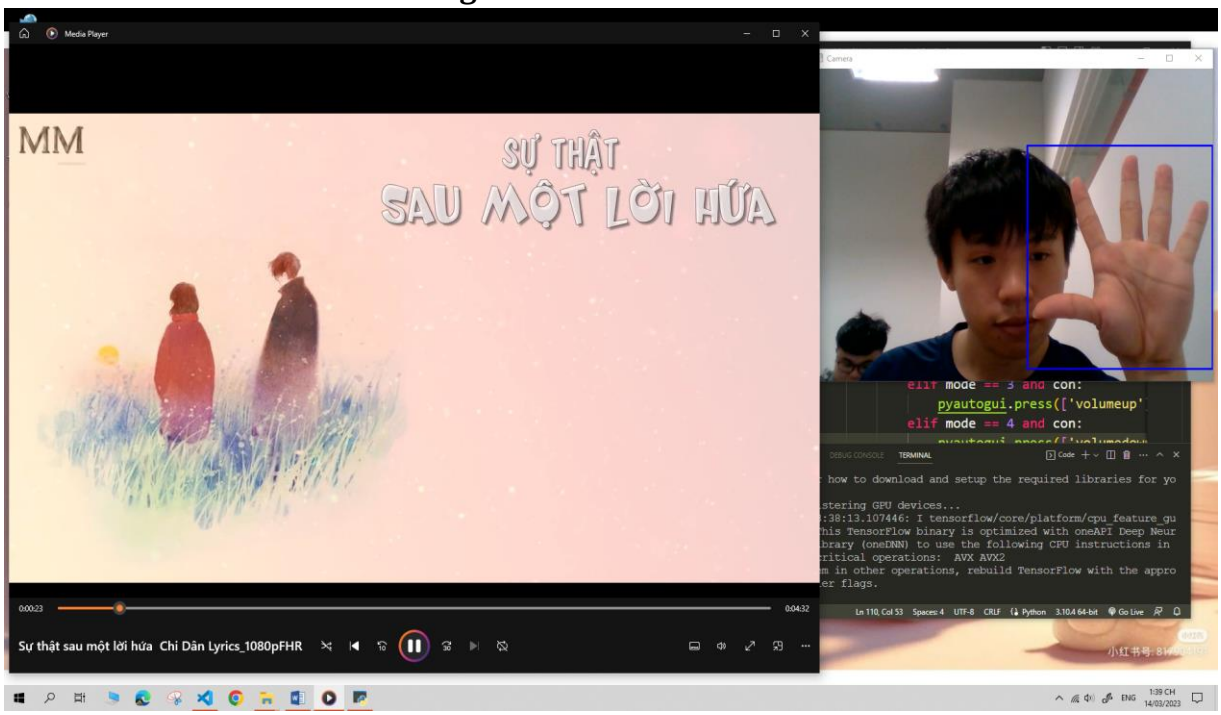

Figure 5.5. Play/Pause

The above-mentioned examples demonstrate how the created system functions well against a plain background (white background).

However, this also serves as a drawback. In order for the model to determine the displayed fingers a plain background is necessary. Noise backgrounds might cause false detections, rendering the technique worthless. In the future, hand detection can be enhanced by using

a stronger and more reliable technique. Such a system would depend on the other characteristics of a human hand in addition to the skin tone. The finger detection mechanism is also affected if the hand is not properly detected. A good hand detection mechanism ensures that he finger counting mechanism works correctly.
Following are some key observations and results of our project:

- The implementation works well with plain backgrounds.
- Sometimes the application fails to detect the hand gesture due to poor image quality.

6. Conclusion and Future work

Conclusioin

In this project, we performed hand and gesture recognition in python using OpenCV. Using the recognized gestures, we controlled various UI elements in the screen, hence making it easier for users to interact with the system. Hand recognition is based on the classification model. Based on which the number of fingers is found out, which becomes the gesture. This is then used to control UI elements. This implementation works well with a suitable background.

Future work

In the future, we can improve upon many aspects of the project. Some of them are:

- Removing the dependency on background.
- Adding more UI control with the recognized gestures specific to a particular application
- Improving the gesture recognition, possibly by using more data.

References

[1] Zheng, Zepei. (2022). Human Gesture Recognition in Computer Vision Research. SHS Web of Conferences. 144. 03011. 10.1051/shsconf/202214403011

[2] Kumarage, D. & Fernando, Sohan & Fernando, P. & Madushanka, D. & Samarasinghe, Raveendra. (2011). Real-time sign language gesture recognition using still-image comparison & motion recognition. 10.1109/ICIINFS.2011.6038061

[3] Schlüsener, N. &Bücker, M. (2022). Fast Learning of Dynamic Hand Gesture Recognition with Few-Shot Learning Models. 10.48550/arXiv.2212.08363

[4] Dinh-Son, Tran. Ngoc-Huynh, Ho. Hyung-Jeong, Yang. Eu-Tteum, Baek, Soo-Hyun, Kim. & Gueesang, Lee. (2020). Real-Time Hand Gesture Spotting and Recognition Using RGB-D Camera and 3D Convolutional Neural Network. 10.3390/app10020722

[5] Adam Ahmed Qaid, MOHAMMEDORCID. Jiancheng, Lv & MD. Sajjatul, Islam. (2019). A Deep Learning-Based End-to-End Composite System for Hand Detection and Gesture Recognition. 10.3390/s19235282

[6] Mahmoud, Elmezain. Majed, M. Alwateer. Rasha, El-Agamy . Elsayed, Atlam & Hani, M. Ibrahim. (2023). Forward Hand Gesture Spotting and Prediction Using HMM-DNN Model. 10.3390/informatics10010001

[7] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen. (2019). MobileNetV2: Inverted Residuals and Linear Bottlenecks. 10.48550/arXiv.1801.04381