A Contact-Free Human-Computer Interaction Interface Based on Embedded System

You-Rong Chen[1], Pei-Yin Chen[2], and Yu-Ting Lee[3]

[1, 2]Digital Integrated Circuit Design Laboratory, Department of Computer Science and Information Engineering, National Cheng Kung University, No.1, University Road, Tainan, Taiwan [3]MediaTek Inc., No.1, Dusing 1st Road, Hsinchu, Taiwan
lt2es.93039@gmail.com[1], pychen@mail.ncku.edu.tw[2], qweszxc12345@gmail.com[3]

Abstract

In this paper, a contact-free human-computer interaction interface is developed. The system recognizes the hand gestures from the user and transmits corresponding control signals to the end-user side. Subsequently, the end-user side manipulates cursor movements or clicks based on the received control signals. To ensure the operability on the embedded system development board, the system adopts low-complexity methods and integrates an external vision processing unit. The experiment results demonstrate the successful implementation of the system on Raspberry Pi 4.

Key words: Computer Vision, Embedded Systems, Hand Gesture Recognition, Human-Computer Interactions

Introduction

The development of human-computer interaction interfaces has led to the successive creation of devices such as mice, keyboards, and touch screens. However, all these devices still require manipulation through direct hand contact. During outbreak of infectious diseases, hand contact with shared devices such as mice, keyboards, or touch screens on automatic ordering machines and automatic teller machines (ATMs) can increase the risk of disease transmission. For instance, the COVID-19 virus can survive on plastic surfaces for more than three days [1]. Disinfecting and cleaning the shared devices not only increases the labor burden, but also leads to additional resource wastage.

To address this problem, we developed a contact-free human-computer interaction interface based on embedded system. With the developed system, users can control the cursor movements and clicks using different hand gestures, thereby interacting with computers without physical contact.

System Architecture

The architecture of the system is illustrated in Fig. 1. The system comprises a Raspberry Pi 4 single-board computer, an external vision processing unit (VPU), a camera, and an end-user device. The Raspberry Pi 4 features two USB 3.0 ports with a transmission speed of 500MB/s and two USB 2.0 ports. The camera is used to capture real-time hand gesture images, while the VPU assists the Raspberry Pi 4 with neural network computation. Therefore, the camera and the VPU are connected to the Raspberry Pi 4 via the USB 3.0 ports to ensure system responsiveness. When the host of the system (i.e. the Raspberry Pi 4) receives real-time images from the camera, it detects the type of hand gesture using a low-complexity convolutional neural network. Corresponding control signals

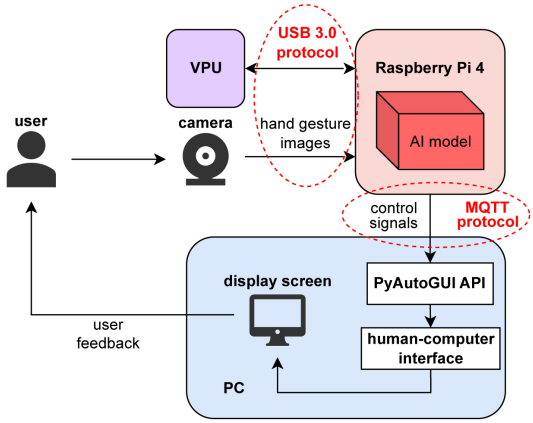

Fig. 1 The architecture of the designed system.

are then transmitted to the end-user device in the system (i.e. personal computer) according to the detected gestures via the MQTT protocol [2]. Finally, the end-user device utilizes PyAutoGUI application programming interface (API) [3] to control the cursor movements and clicks and enable contact-free interactions. Fig. 2 shows the practical implementation of the system. The system can be easily installed on a standard display screen, ensuring the portability of the system.

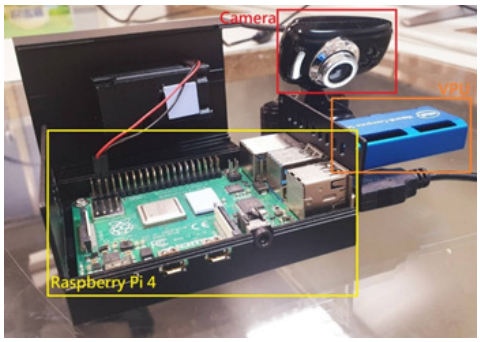

Fig. 2 Practical implementation of the system.

Experimental Methods

In the experiment, we use an AI model to detect the user's hand gestures and introduce low-complexity methods to ensure that the AI model can operate on the Raspberry Pi 4.

A. Model Structure

The designed hand gesture detection model adopts an AutoEncoder architecture [4], which is constructed with an encoder and a decoder. The encoder side extracts deep features from the input images and generates compressed low-

2024 10th International Conference on Applied System Innovation (ICASI) | 979-8-3503-9492-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICASI60819.2024.10548002

dimensional representations of the input. The decoder side then decompresses the low-dimensional representations to obtain output results, which are the keypoints of user's hand in this task. Fig. 3 illustrates the structure of the designed AI model. The encoder side adopts the architecture of ShuffleNetV2 [5], while the decoder side utilizes two transposed convolution branches to predict the keypoints of the hand and the correlations between the keypoints, respectively. The type of hand gesture is then determined according to the relative position of each keypoint.

In this experiment, the fist gesture corresponds to moving the cursor, while the open-palm gesture represents clicking the cursor. To ensure that the model can correctly detect the user's hand gesture, we collect 11113 and 1812 images with different hand gestures as the training data and the testing data, respectively. Each image is annotated with the positions of 21 hand keypoints and their connectivity relationships. The loss function during training adopts the mean squared error (MSE).

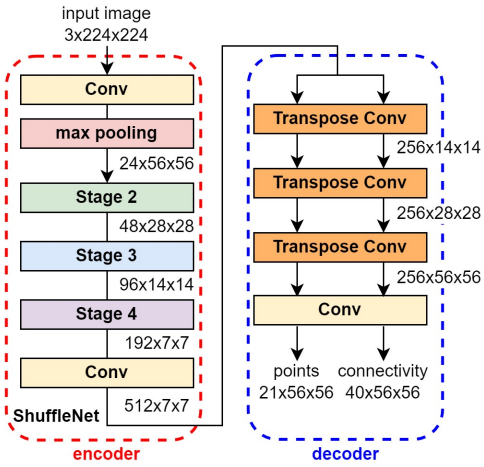

Fig. 3 Model structure.

B. Low-Complexity Methods

In order to implement the designed model on the Raspberry Pi 4, two low-complexity methods are introduced. The first method is group convolution [6], which divides the input feature maps into several groups and performs convolution on each group separately. Group convolution effectively reduces the number of parameters required by the convolutional layers. Furthermore, separating the feature maps at the convolutional layer level can prevent the features of different keypoints from being confused with each other. The second method is depth-wise separable convolution [7], which separates the process of convolution to depth-wise convolution and point-wise convolution. The depth-wise convolution process utilizes dedicated filters for each channel of input feature maps to perform convolution calculations. The point-wise convolution process then uses 1x1 convolution kernels to fuse the information from the results of the depth-wise convolution.

C. Results

Actual usage results of the designed system are shown in Fig. 4. In Fig. 4(a) and Fig. 4(b), the user controls the movement of the cursor with a fist gesture. In Fig. 4(c) and Fig. 4(d), the user controls the cursor to click with an open-palm gesture and successfully opens the folder.

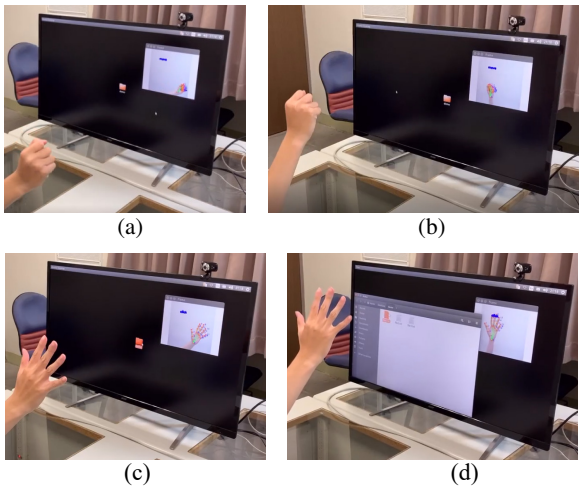

Fig. 4 Actual usage results.

Conclusion

In this paper, we designed a contact-free human-computer interaction interface and implemented it on the Raspberry Pi 4 single-board computer. With the assistance of the VPU and low-complexity methods, the designed AI model can successfully operate on the resource-constrained device. The experimental results show that users can control the movement and clicking of the cursor without physical contact. Future work includes adding more recognizable gestures and further improving computing efficiency to enhance the user experience.

Acknowledgement

This work was supported in part by Qualcomm through a Taiwan University Research Collaboration Project.

References

[1] V. Doremalen, N. *et al*., "Aerosol and Surface Stability of SARS-CoV-2 as Compared with SARS-CoV-1," *New England jornal of medicine*, vol. 382, no. 16, pp. 1564-1567, 2020.

[2] MQTT: The Standard for IOT Messaging. Accessed: 2022. [Online]. Available: https://mqtt.org/

[3] PyAutoGUI. Accessed: 2019. [Online]. Available: https://pyautogui.readthedocs.io/en/latest/

[4] D.E. Rumelhart, G.E. Hinton, and R.J. Williams: Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chap. Learning Internal Representations by Error Propagation, pp. 318-362, 1986.

[5] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, “ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design,” in *Proceedings of the European Conference on Computer Vision* (*ECCV*), pp. 116-131, 2018.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems* (*NIPS*), pp. 1097-1105, 2012.

[7] A. G Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “MobileNets: Efficient convolutional neural networks for mobile vision applications,” *arXiv*, p. 04861, 2017.

ISBN 979-8-3503-9492-4