Development of Memory Support System by Smart Glasses

Zijun Yang^{a,*}, Seiichi Serikawa^a

^a Kyushu Institute of Technology, Kitakyushu, 8048550, Japan

* Corresponding Author: zijun.yang529@mail.kyutech.jp

Abstract

This paper mainly introduces how to use intelligent glasses to realize the memory of human behavior. We use smart glasses to record human behavior and the environment in which it occurs. People with cognitive and memory impairment can use smart glasses to help them acquire non-invasive memories. The data obtained through smart glasses will be processed by the system in the background while the user is sleeping. Data can be awakened by voice. What is currently being done is the part of image recognition, after which a speech processing system will be further established.

keywords: smart glasses, Python, OpenCV.

1. Introduction

With the development of electronic technology, electronic products are gradually inputting into every corner of our lives. They can help us solve many trivialities and inconveniences. For example, the sweeping robot can help us with housework, support robots can help people with mobility problems, and companion robots can accompany us to do some simple things to relax. But in fact, in our lives, there are some people with cognitive impairment and memory impairment, they can not properly distinguish between the people they encounter in life, resulting in problems in life.

So there are smart glasses and other products to help them identify people in their lives.

The foundation of smart glasses is based on face recognition technology. On the basis of face recognition, terminals can be replaced. Other forms of portable terminals can be made to facilitate people to carry around. They can also expand services, such as increasing the range of image recognition, to carry out landscape recognition; increase voice recognition function, input and output voice; increase infrared control function, remote control for home.⁽¹⁾⁽²⁾ This paper mainly realizes that the smart glasses recognize the face at the back end, and feedback the user at the front end in

the form of a window. The user can see the image of the person and the image of the face in the window, and can remark the name and other information. More terminal functions mentioned above will be implemented later.

The program involves the project mainly divided into three parts. The first part mainly captures the face, forms the training set and trains, the second part mainly realizes the recognition of the face, and the third part mainly realizes the window display picture and related information, supplement the relevant information. The face recognition program is mainly based on python + OpenCV. The relevant content will be explained in turn below. And the construction of user window is mainly realized by Pygame.

2. The establishment of face capture and recognizer

This chapter mainly introduces the content of capturing faces, forming training sets and training, and the recognition of faces.

2.1 Face to capture

The first is the capture of the face. We take a screenshot of the camera shot at a certain frequency, we name it "frame", and then we will process the image. This is a real-time image of the relevant, we can also use the existing video to take a screenshot based on a certain frequency. The two main implementation functions are the same, but the instructions opened when executing the OpenCV instruction are different (the former is Capture but the latter is video).

First of all, we need to use OpenCV for grayscale conversion of the obtained images. The purpose of this is to reduce the amount of computation at run time and meet the requirements of subsequent algorithms. After that, we need to detect the face of the grayscale image. The algorithm used by the program for detection is Dlib's `get_frontal_face_detector` algorithm for obtaining positive faces. The picture of the

detected face is further processed, and the undetected is no longer processed.

2.2 The establishment of recognizer

Due to the initial of the program, the data is empty, so we need to collect the face as the basis for information collection and training, so we need to collect multiple photos for a single person, and use the input command to enter the name corresponding to the face or someone with a number. Code different people to name them. For ease of use we need to standardize the way we name it, such as "username + num + sequence number". The "serial number" can be numbered either sequentially or with a timestamp. At this point, we use the program to perform face detection and create several face folders. These folders will hold several photos of the corresponding person. These folders are the data sets we built.

As shown in Fig. 1. File storage.

Fig. 1. File storage

Next, we need to train the data in the dataset. The recognition program we use is OpenCV's own LBPH face recognition algorithm to process images. LBPH means Local Binary Patterns Histograms, with each pixel as the center, the relationship between the center point and the surrounding 3×3 pixel gray value is judged, the binary processing is carried out, and a list is established to store the feature vectors of different faces. (B), (U)

As shown in Fig. 2. LBPH encoding mode.

At this point, we have completed the establishment of the data set. In addition to this algorithm, we have many other known face recognition algorithms, such as FisherFaces and EigenFaces under OpenCV, and a library of known algorithms for 68 and 128 face symbolic value extraction under Dlib.

EigenFaces optimizes just train themselves by extracting

Fig. 2. File storage

the main components of a face, and they keep track of which components belong to which people. So every time we introduce a new image to the algorithm, it repeats the same process: extract the main components from the new image, compare those features with a list of elements stored during the training, and find the face that best matches. Returns the "USER" tag associated with the best-matching component. However, the feature surface algorithm takes illumination as an important feature. Light will have a great impact on the recognition results. (S), (S)

FisherFaces is an updated version of the EigenFaces algorithm. By adjusting the feature face, it can extract useful features from each person's face, rather than from all the combined faces. In this way, even if one person's illumination varies greatly, it will not affect the feature extraction process of others. Specifically, the FisherFaces face recognition algorithm extracts the main components that are differentiated from other faces. In this sense, the individual components do not dominate (become more useful) the other components. (U), (S)

Dlib is a face recognition algorithm of C++, which extracts 68 key points of a face, and the information of the key points is shown in the Fig. 3. Dlib key points.

Fig. 3. Dlib key points

Dlib points relatively complete, the use of these point information can be selected from the frame facial contour, glasses, mouth and other features. (U), (UU)

The main reason we chose LBPH is that it has the advantage of being immune to illumination, scaling rotation and other factors. Now we pass the image in the dataset to the recognizer, which generates a list of feature vectors for each

person. At this point, our first step is complete.

The program flow chart as shown in Fig. 4. Program_1 flow.

Fig. 4. Program_1 flow

3. Face recognition and User Windows

The second step, what we need to do is face recognition.

We only need to upload the face we want to detect to the program and repeat the part of the first step, that is, the video information is captured at a certain frequency; the face detection is performed on the cut image, and the detected face is input to the recognizer. Contrast matching is performed in the recognizer. If a face is recognized, the face is framed by a rectangle and its name tag is attached.

We hope this is a more useful application, so we hope that the system can give us some feedback when we need to know the information about the identified object. In the later stage, we will realize this function through speech recognition. Now

we will show it in the form of a normal program. First, we have created a picture browser that can display a character image.

As shown in Fig. 5. User Windows.

Fig. 5. User Windows

For convenience, we will also design the mobile client to implement the function of image browser, as shown in Fig. 6. Mobile phone User Windows.

Fig. 6. Mobile phone User Windows

We can frame the face through the first part of the content, and display the information we want to know on the other side and we want Supplementary information. Information will be related to each other in a dictionary. The picture browser of this program is made with Pygame, we can see the characters and face close-up in each picture (we need to deal with the size of all faces in a unified way). We can switch between the previous picture or the next picture by keyboard operation. You can choose whether to keep this picture by different buttons “S” and “D”, or you can choose to delete one picture or delete all pictures of this person.

The program flow chart as shown in Fig. 7. Program_2 flow.

As a specific application, in order to use convenient, in the late we have reconstructed the whole program, we select some can be user-defined variables as separate modules, such as: capture the frequency of images, video saved path, picture save path and so on, we will also implement different func-

Fig. 7. Program_2 flow

tions of the program in the different modules, such as face capture module, recognizer module, face recognition module. In the same way, we separate the back-end from the front-end. Program refactoring makes it easy to modify variables and functions intuitively. It is also more convenient for us to optimize the follow-up program and expand the function.

And the final result as shown in Fig. 8 . Face recognition result and Fig. 9 User windows result.

Fig. 8. Face recognition result

Fig. 9. User windows result

4. Conclusions

This thesis enriches the function of smart glasses through the research of face recognition function, realizes the record of human glasses on human behavior and the function of helping human memory. The article is adopted. The algorithm will also be more convenient to use in the future by adding a speech recognition system. The direction of image memory also adds to the memory of the surrounding environment.

References

- (1) N. Kaushik and T. Sasaki and Y. Takahashi and T. Nakazawa and K. Hane : "Smart Glass Type Retinal Imaging System Using MEMS Scanner", 2018 International Conference on Optical MEMS and Nanophotonics (OMN), pp. 1-5, 2018
- (2) N. Sumi and V. Moshnyaga : "A novel face recognition for smart glasses", 2016 IEEE Region 10 Symposium (TENSYP), pp. 115-118, 2016
- (3) G. Xiang and Z. Qiuyu and W. Hui and C. Yan : "Face recognition based on LBPH and regression of Local Binary features", 2016 International Conference on Audio, Language and Image Processing (ICALIP), pp. 414-417, 2016
- (4) X. Zhao and C. Wei : "A real-time face recognition system based on the improved LBPH algorithm", 2017 IEEE 2nd International Conference on Signal and Image Processing (ICSIP), pp. 72-76, 2017
- (5) Wei Wang and Jiatao Song and Zhongzhiu Yang and Zheru Chi : "Wavelet-based Illumination Compensation for Face Recognition using Eigenface Method", 2006 6th World Congress on Intelligent Control and Automation, pp. 10356-10360, 2006
- (6) H. Zhang and W. Liu and L. Dong and Y. Wang : "Sparse eigenfaces analysis for recognition", 2014 12th International Conference on Signal Processing (ICSP), pp. 887-890, 2014
- (7) TShiguang Shan and Bo Cao and Wen Gao and Debin Zhao : "Extended Fisherface for face recognition from a single example image per person", 2002 IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No.02CH37353), Vol. 1, No. 2, pp. II-II, 2017
- (8) S. Jin and Y. Lin and H. Wang : "Automatic Modulation Recognition of Digital Signals Based on Fisherface", 22017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), pp. 216-220, 2017

- (9) S. Sharma and K. Shanmugasundaram and S. K. Ramasamy : “FAREC — CNN based efficient face recognition technique using Dlib”, 2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), pp. 192-195, 2016
- (10) N. Boyko and O. Basystiuk and N. Shakhovska : “Performance Evaluation and Comparison of Software for Face Recognition, Based on Dlib and Opencv Library”, 2018 IEEE Second International Conference on Data Stream Mining Processing (DSMP), pp. 478-482, 2018