Project report

Yuyichen ID：5140309200

EE327-G12 May 17,2017

My project title is Software development for iblink. Iblink is a project which contains a wearable computing and its software development. This wearable computing is designed to be a pair of glasses with a camera. Facioplegia were will benefit from this project. However, we should slove some software problems about blink detect.

Our work aims at finding pervasive algorithms for blink detection working in iblink. Both the accuracy and resource utilization should be taken into account.

1、Hardware foundation

Iblink is a pair of glasses with a camera . This wearable computing will help facioplegia were express their ideas by eye action. We use adafruit's 3D Printed design, which turns a pair of 'private display glasses' into a "google glass"-like form factor. It easily clips to your prescription glasses, and can display any kind of device with Composite Video like a Raspberry Pi.[1]

The camera is designed to catch the small act of one eye. As a result, we can only get an image containing only one eye.

open closed

Figure 1: Acquired Image

At the same time, due to the limited operating capacity of the processor, the existing program can not process the image in time, and can not meet the design requirements.

2、Existing program

The program we used is a program system containing real time eye tracking and blink detection. This program is designed to check a picture with whole face ,which is not suitable for our situation.

The system presented here detects the user’s eye blinks and analyzes the pattern and duration of the blinks, using them to provide input to the computer in the form of a mouse click. After the automatic initialization of the system occurs from the processing of the user’s involuntary eye blinks in the first few seconds of use, the eye is tracked in real time using correlation with an online template.[2]

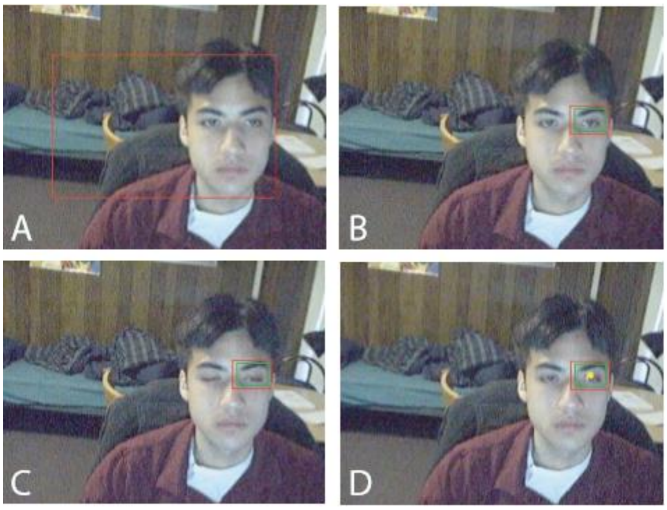

Figure 2: Real Time Eye Tracking and Blink Detection

This program consists of four parts, initialization, template creation ,eye tracking and blink detection. However, we don't need so many steps because the hardware we use have already given us an image of the eye area. Moreover, the image we get is not adapted to existing facial recognition programs. Such a redundant program will waste a lot of computing resources. The result is that our blink detection program becomes sluggish and low cost performance.

3、Advanced design

Due to the fact that the existing blink detection program is not suitable enough for our need, we should design a new program which is about blink detect for an eye-only image with low resource use.

I try an algorithm based on eye templates. This algorithm will firstly load a number of eye images which we have already known if they're open or closed. For example, we generally load one hundred open-eye images and the same number closed-eye images. We'll convert them into grayscale images and create an array to save the average value of one hundred samples. The two arrays we get will be the templates and standards to represent closed eye and open eye.

We can translate the array into gray images to understand calibration results:

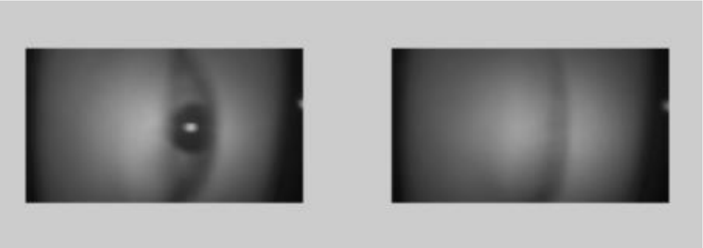

Figure 3: Graphical standards

After get the standards, we can compare the immediate image due to an compare algorithm.

The image to be measured should be input as grayscale information in array as well. To calculate easily, we should do a histogram equalization (In test program with matlab ,histqe() function can easily do this thing). Then we get the array img(i,j) containing each pixel points' grayscale information. Using following formula will give us the similarity between the image to be measured and the standards. (std(i,j) is the pixel points' information of standard)

$$\mathrm{sim} = \frac{\sum \mathrm{img}(\mathrm{i},\mathrm{j}) * \mathrm{std}(\mathrm{i},\mathrm{j})}{\sum \mathrm{img}^2(\mathrm{i},\mathrm{j}) + \sum \mathrm{std}^2(\mathrm{i},\mathrm{j})}$$

Due to the formula ,we can get sim1 and sim2 which is about the similarity of open-eye standard and closed-eye standard. By comparing them, we can easily know if the eye image we get is open.

The advanced program give up the face recognition part. The content of eye tracking is given to the hardware acquisition section. This greatly reduces the complexity of the software part and the use of resources. However, the template section needs pre-added and extra storage.

Unless the wearable computing can provide steady eye images, we should add trace program in order to get an image that is easy to handle.

4、Future work

(1) Solving standards storage issues

(2) Translating existing matlab program into c++ program

(3) Test programs on hardware and solving stability problems

5、Abbreviated program

```
ia=720;ib=1280;
summ=zeros(ia,ib);
for i=1:100
    str=strcat('F:\ed\EyeData\test\0\lux0\',int2str(i),'.jpg');
    img1=imread(str);
    img1=rgb2gray(img1);
    summ=double(summ)+double(img1);
end
stdopen=summ/100;
summ=zeros(ia,ib);
```

```
for i=1:100
    str=strcat('F:\ed\EyeData\test\1\lux0\',int2str(i),'.jpg');
    img1=imread(str);
    img1=rgb2gray(img1);
    summ=double(summ)+double(img1);
end
stdclose=summ/100;
fid=fopen('F:\ed\EyeData\test\2\result.txt','wt');
   for k=31:50
temp1=0;
temp2=0;
temp3=0;
tempa=0;
tempc=0;
     str=strcat('F:\ed\EyeData\test\2\',int2str(k),'.jpg');
    img=imread(str);
    img=rgb2gray(img);
    img=histeq(img);
    for i=1:ia
        for j=1:ib
        temp1=temp1+double(img(i,j))*double(stdopen(i,j));
        temp2=temp2+double(img(i,j))^2;
        temp3=temp3+double(stdopen(i,j))^2;

        tempa=tempa+double(img(i,j))*double(stdclose(i,j));
        tempc=tempc+double(stdclose(i,j))^2;
        end
    end
sim1=temp1/(sqrt(temp2*temp3));
sim2=tempa/(sqrt(temp2*tempc));
if(sim1<sim2)
    fprintf(fid,'the %d eyes close\n',k);
else
    fprintf(fid,'the %d eyes open\n',k);
end
   end
```

6、References

[1] https://learn.adafruit.com/diy-wearable-pi-near-eye-kopin-video-glasses

[2] Michael Chau and Margrit Betke, Real Time Eye Tracking and Blink Detection with USB Cameras, Abstract