Eye-blink Detection Based on SVM

Xiaoxing Wang*

Shanghai Jiao Tong University

figure1_wxx@sjtu.edu.cn

May 17, 2017

Abstract

My senior Sijie Xiong had already make a pretty completely smart glasses, and perform very good. In this semester, I achieve a new method to locate the eyes from an image of the whole face, and detect whether each eye is open or not.

Good as the iBlink performs, it still has some drawbacks. Firstly, the camera has to be fixed behind the glass, which makes the glass inconvenient to take. Secondly, their input image for eye-blink detection is the images of eyes, but in general, it is more easy to acquire images of faces. Finally, the algorithm they used for eye-blink detection is based on convolutional neural network(CNN), which is pretty difficult to compute on common devices.

My contributions are: First, I find a pretty good method to locate two eyes from the images of faces, which means that we can easily acquire face images from our mobile phones. Second, I achieve a new algorithm for eye-blink detection which need much less computation. Third, I simplify my algorithm after I add some hypothesis, which is also very reasonable in actual life. Moreover, the method I use are all achievable on a common mobile phone.

I. Introduction

The smart glasses, iBlink have brought great news for people suffering facial paralysis. In my seniors' paper [1] they have stated the universality and harmfulness of facial paralysis, So they created wearable device to improve the facial paralysis patient's quality of life. As their device performs pretty well, it still has several drawbacks: inconvenient to take, complicated computation.

In this paper, I propose to improve the algorithm for eye detection and eye-blink detection. In particular, I find a pretty effective method to locate the position of two eyes from an image of a whole face, and then use another algorithm other than CNN for eye-blink detection.

First, To locate the face and get landmarks of the eyes from an image, I use the frontal face detection model of dlib C++ Library to detect faces in an image. And use the face-alignment model of dlib to locate 68 landmark of a face, in which, there are 6 points to locate each eye. The details will be shown in section II

Second, for eye-blink detection, I use support vector machine(SVM) to classify the two pattern of eyes. To simplify the difficulty, I assume that there is only one user to use the model, that is, one model is only suitable to detect a specific user. Actually, the hypothesis above is reasonable, and the reason will be stated on the section III Facial paralysis is not contagious, and the device is pretty personal for a patient, so once there is a user using the device, he is certainly not willing to share the device with another patient. Based on this hypothesis, I can simplify the training process and make it much faster for one train on the premise of accuracy.

What's more, I experiment on each features and parameters of SVM, and list some points which can be improved for further research. Through the experiments, I can see the strength of LBP, the good characteristic of its improved versions(see in the subsection IV.2), and the

*Student ID: 5140309247

optimal training parameters for SVM models. In the section V, I list some drawbacks of the method of eye-location, and shows how can the accuracy improved.

II. Ocular Image Acquirement

II.1. Working Procedure

In reality, it is more easy to acquire facial images, rather than ocular images. What's more, if we want to acquire real-time images from our mobile phones, we have to locate the eyes' position and than deal with the ocular images. So the working procedure of ocular images acquirement is shown in Figure 1.

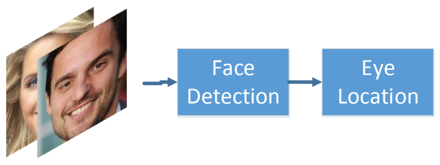

Figure 1: *Procedure of Ocular Image Acquirement*

The input image is a frontal face, and we detect the face position in the image and get the bounding box. Afterward, we acquire 68 facial landmarks according to the bounding box, and extract the 12 points for eyes(6 for each one). Finally, we crop the ocular image based on the 12 points, and resize the image for further training(discuss in section).

II.2. Face Detection and Eye location

To detect the face and locate the eyes, I use the model provided by dlib C++ Library. At first, I choose the "haarcascade_frontalface" model provided by OpenCV to achieve face detection. However, the accuracy is bad. After checking more information, I find dlib C++ Library, whose model performs much better and the speed is suitable for the classification.

Furthermore, dlib C++ Library also provides the model of facial landmark detection, which can be used for eye location, whose accuracy is also suitable for this project.

Actually, The functions I use in this project is two fundamental ones, if there is another simply but more accurate method to achieve above goal, we can change the method.

II.3. Geometric Approach for Eye-blink Detection

A teammate who is also doing the same project told me a new geometric approach for eye-blink detection. The steps are shown as follows

- Locate the key points of each eyes. Usual landmarks of a face have 6 points to calibrate one eye, and record the position of each point.
- Calculate the width$W$ and height$H$ of each eyes, and divides W with H, that is,

$$R = \frac{W}{H} \tag{1}$$

- Set a threshold for R, suppose is R_t. For those eyes whose $R \leq R_t$ are open, and close on the opposite.

This method is fast and easy to achieve, and will have relatively high accuracy is the approach of face alignment is accurate. But it relies to much on the accuracy of face alignment. If the key points change a little, the ratio R will change a lot.

III. SVM

To get rid of the limitation of face alignment, which hasn't had a pretty good way to achieve, I propose to use support vector machine(SVM) for eye-blink detection. There are several advantages for SVM:

- SVM can get pretty good results even if the number of training data is small. As my hypothesis, if the model will be trained for every new users, we have to update the training database from camera. To satisfy the convenience of users, we can't shoot too much time, at most 2 minutes either for open eyes or close eyes, which means that for each user, we have a small database to train the model.

Table 1: *Comparison among three versions of LBP*

Pattern	Dimension of vectors	Characteristic
Original LBP	256	Illumination Invariance
Rotation Invariance Pattern	39	Illumination & Rotation Invariance
Uniform Pattern	59	Illumination & Rotation Invariance

- SVM has much less parameters than convolutional neural network(CNN) does, so the speed of detection is faster and the necessary memory space for model is also very small.
- Instead of put the whole image to the model, we can extract the features in advance, which can represent the image and have some pretty good characters.

From the above statement, I find that SVM approach is very suitable for my purpose. And the following sections will show the process of feature extraction and training in detail.

III.1. Feature Extraction

III.1.1 choice

At first, I just use the most naive features: put all the pixels value into an 1-D array. As expected, the results are very poor, although, this kind of features can get pretty good outcome on handwritten-detection. Afterwards, I use the histogram as the features to represent the image, but the histogram will drop many details and as a result, the effect is not very good.

Afterwards, I find another useful feature: LBP features. After looking for some information and papers, I find two effective features for object detection: Histogram of Oriented Gradient(HOG) and Local Binary Patterns(LBP). From the [2],I know that HOG is more suitable for pedestrian detection, and LBP is more suitable for face detection. So I choose LBP feature to train the model.

III.1.2 Strength of LBP

From the information, research, and my test, I find LBP feature has the following advantages.

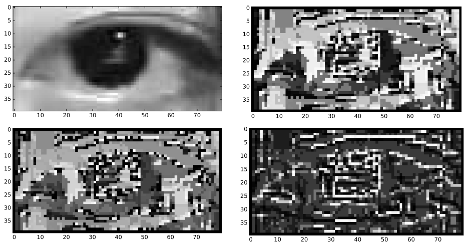

Figure 2: *(a)the cropped eyes. (b)the original LBP of (a). (c)the Uniform Pattern LBP of (a). (d)the Rotation Invariance Pattern LBP of (a).*

- The processing is much simple than HOG, and has a good performance.
- LBP features is insensitive to illumination. This characteristic has two strength: on the one hand, it means we don't care too much about illumination when taking photos. On the other hand, it means that we don't need to take in more data or train an independent model for low light levels.
- There are two more improved version of LBP, and they are insensitive to the rotation of images, which can suit more situations.

III.1.3 LBP

LBP is a kind of method to reconstruct the original image. For every pixel, we compare it with other 8 pixels around it, and get a 8-bit binary array, which is the representation of the original pixel. The output of LBP processing is a new way to express the image.

Based on the original LBP features, there are several improved LBP features, and I also make several test on original LBP feature, Rotation Invariant Pattern of LBP and Uniform Pattern of LBP. The characteristic for each pattern is list on Table 1. From the experiment results

Table 2: *Parameters For SVM*

Parameter	Value
Kernel Function	Linear/Gaussian
C	default 1.0
k	number of classification
γ	default $\frac{1}{k}$

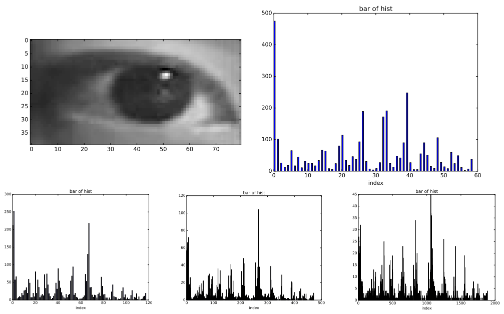

Figure 3: *(a)an open eye. (b)the histogram of (a). (c)the histogram of (a) with 1×2 blocks. (d)the histogram of (a) with 2×4 blocks. (e)the histogram of (a) with 4×8 blocks.*

discussed in section IV, I find Uniform Pattern of LBP is the most effective. The examples are shown in Figure 2and Figure 4

In order to extract the features, I use Multi-block LBP [3]. First, I cut the original image to several blocks, calculate the histogram for every block and concatenate them as the final feature. As for the number of the blocks, I make several test, and find that it will get the perfect effect if we divide the output of LBP into 4×8 blocks. The bar of histogram of blocks are shown in Figure 3, from which we can see more blocks expand the size of feature, and it shows more details.

III.2. Training

Firstly, the choice of kernel function is important. For support vector machine(SVM), one of the most important parameters is the choice of kernel function. There are 3 most general kernel functions: Linear Kernel, Gaussian Kernel and Polynomial Kernel. Though Linear Kernel is the usual choice because of its fast speed, but Gaussian Kernel usually performs better for simple features.

Secondly, the penalty factor C for C-SVC is also important. Penalty factor can be seen as the tolerance for misclassified samples, the higher the penalty factor C, the less the misclassified samples are, but the worse the generalization ability is.

For Gaussian Kernel, there is another important parameter γ. If γ is set very high, the accuracy will be improved, but the model will only suitable for samples which are very close to the training samples, that is, the generalization ability is very poor.

The three free parameters are list in the Table 2. There is a pretty useful and convenient function in the svm library of OpenCV, called train_auto, which can search for the optimal value for each parameter. However, there is a trade-off in the choice of penalty factor and γ, so we have to change the parameters around the given optimal value manually.

IV. Experiment and Results

IV.1. experiment on features

To decide which feature can describe the eyes perfectly, I experiment on 3 features: naive feature, original LBP(O-LBP), Rotation Invariance pattern(RI-LBP) and Uniform Pattern(U-LBP). The training data is the from the database Sijie Xiong's team use. For this experiment, I use the eyes shot under 1000 luminous intensity. The parameters I choose is 4×8 blocks, SVM parameters: auto-train and Linear kernel. Results are shown in the Table 3. We can see the LBP performs much better than naive feature. However, because of the training data is little,

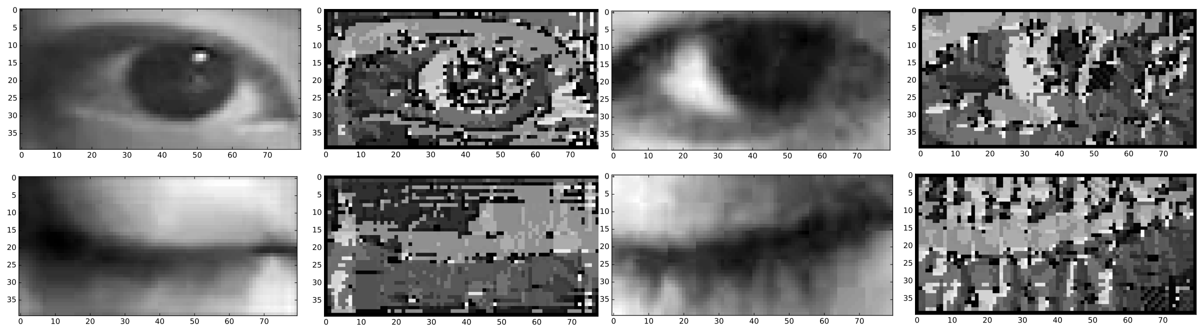

Figure 4: *(a)open eye with high resolution. (b)the Uniform Pattern LBP of (a). (c)open eye with low resolution. (d)the Uniform Pattern LBP of (c). (e)close eye with high resolution. (f)the Uniform Pattern LBP of (e). (g)close eye with low resolution. (h)the Uniform Pattern LBP of (g)*

the strength of U-LBP cannot indicated. If I augment the data, and add some interfere to the image, U-LBP performs much better than O-LBP because of the Rotation Invariance characteristic. The reason for the poor performance of RI-LBP is the few dimensions.

Table 3: *Experiment on features*

Feature	Training Accuracy	Val Accuracy
naive	55.87%	26.20%
O-LBP	100%	93.57%
RI-LBP	100%	74.10%
U-LBP	100%	90.36%

IV.2. experiment on characteristic of LBP

- **experiment on Rotation Invariance**
  To test the rotation invariance characteristic, I first draw the LBP before the rotation. Second, rotate the ocular image by 90°, and draw the LBP after the rotation. Finally, compare the two LBP feature maps. The results are shown in the Figure 5, from which we can see that the rotation invariance of RI-LBP shows clearly, and the characteristic exists in U-LBP
- **experiment on Illumination Invariance**
  To test the illumination invariance characteristic, I first decrease the pixels of an ocular image by 50. Second draw the LBP map, and the histogram of LBP feature. Comparing the results with the histogram of original image. The results are shown in the Figure 6. We can see though the gray level histogram of original image changed a lot, the histogram of U-LBP still keeps the original details.

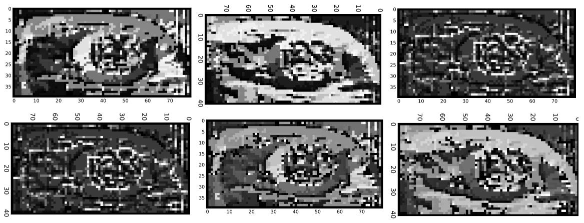

Figure 5: *(a)O-LBP before rotation (b)O-LBP after rotation. (c)RI-LBP before rotation (d)RI-LBP after rotation. (e)U-LBP before rotation. (f)U-LBP after rotation*

IV.3. experiment on parameters

- **Experiment on number of blocks**

Table 4: *Experiment for blocks*

Blocks	Training Accuracy	Val Accuracy
1×2	81.30%	66.58%
2×4	99.97%	91.23%
4×8	100%	93.33%
8×16	100%	90.16%

More blocks can augment the number of

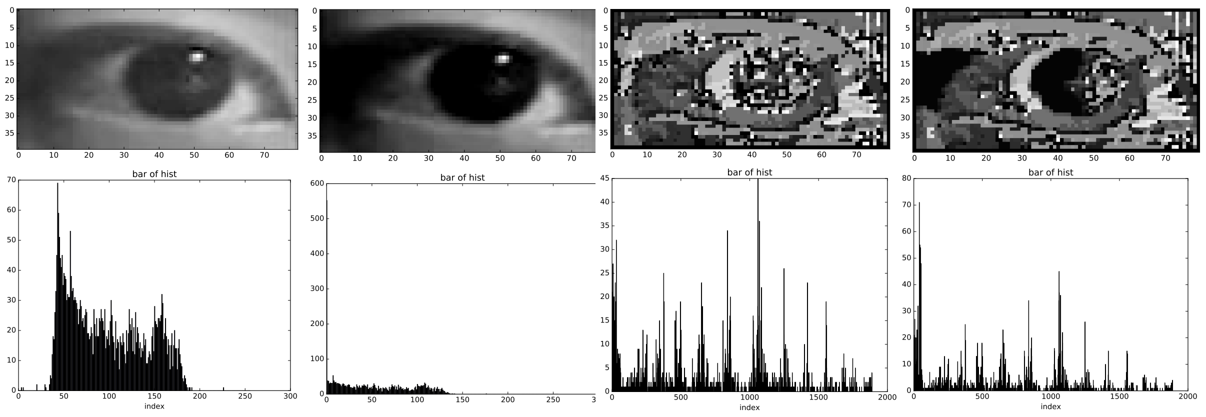

Figure 6: *(a)the original ocular image (b)the dark ocular image. (c)the U-LBP of (a). (d)the U-LBP of (b). (e)the histogram of gray level of (a). (f)the histogram of gray level of (b). (g)the U-LBP histogram feature of(a). (h)the U-LBP histogram feature of (b).*

features, and describe more details. But too many blocks will cut the original image into very small pieces, whose message in the useless area will affect the final results. The results of this experiment are list in the Table4. We can see that 4×8 blocks perform best. From the experiment, 8×16 blocks also have the lowest speed. So I choose 4×8 blocks to extract features.

- **Experiment on kernel functions**
  For LBP features, I find the accuracy on the same validation data for Linear Kernel and Gaussian Kernel is very close 5. Gaussian Kernel performs little better than Linear Kernel, but the selection of the parameter γ is very important. What's more, the speed of Gaussian Kernel is much lower than Linear Kernel. If the number of dimension is suitable for the number of training data, Linear Kernel can suit the case. For 4×8 blocks and U-LBP feature, we can get 1888-D feature.

To get the optimal parameters, such as the number of blocks, LBP pattern and kernel function, from experiments, I use variable-controlling approach, changing one parameter and keeping others stable. Besides the accuracy on validation data, the sense of result on video and camera is also taken into account.

Table 5: *Experiment for kernel function*

Kernel	Parameters	Val Accuracy
Linear(auto)	$C = 0.1$	93.33%
Linear	$C = 0.5$	92.77%
Linear	$C = 1$	92.77%
Gaussian(auto)	$C = 2.5, \gamma = 10^{-4}$	95.55%
Gaussian	$C = 2.5, \gamma = 0.5$	49.9%
Gaussian	$C = 1, \gamma = 10^{-4}$	95.81%
Gaussian	$C = 0.5, \gamma = 10^{-4}$	95.08%

V. Improvement

Though my approach does perform good, there are also several points need to be improved.

First, The method of locating eyes need to be improved. Lacking in training database, I can't train my own model for eye-location. The method I use is discussed in the subsection II.2. It has a drawback: the accuracy of eye-blink detection relies on the accuracy of eye-location, which may get poor performance in the dark situation, though LBP has the characteristic of illumination invariance.

Second, My result relies on the quality of front-facing camera. The camera of my cell phone has very low resolution, and the results on validation videos are not very good. But the cameras with high resolution can get pretty good performance.

Third, every user has to shoot 2 training videos for open eyes and close eyes, one minutes for each video. The following goal of my approach is to train a model suitable for most people.

References

[1] S. Xiong, "iblink: Smart glasses for facial paralysis patients,"

[2] "Three magic keys for object detection: Hog,lbp,haar." `http://www.open-open.com/lib/view/open1440832074794.html`.

[3] "Local binary patterns." `https://en.wikipedia.org/wiki/Local_binary_patterns`.