Report Of The Final Project

BY:

LYM:5180309910393 ; ZKP:5180309910374 ; ZWT:5180309910405

Warnings

- The code contains the command lines for cmd, so please run all our projects on windows platforms.
- Please make sure your computer is equipped with the module pyqt5 and os

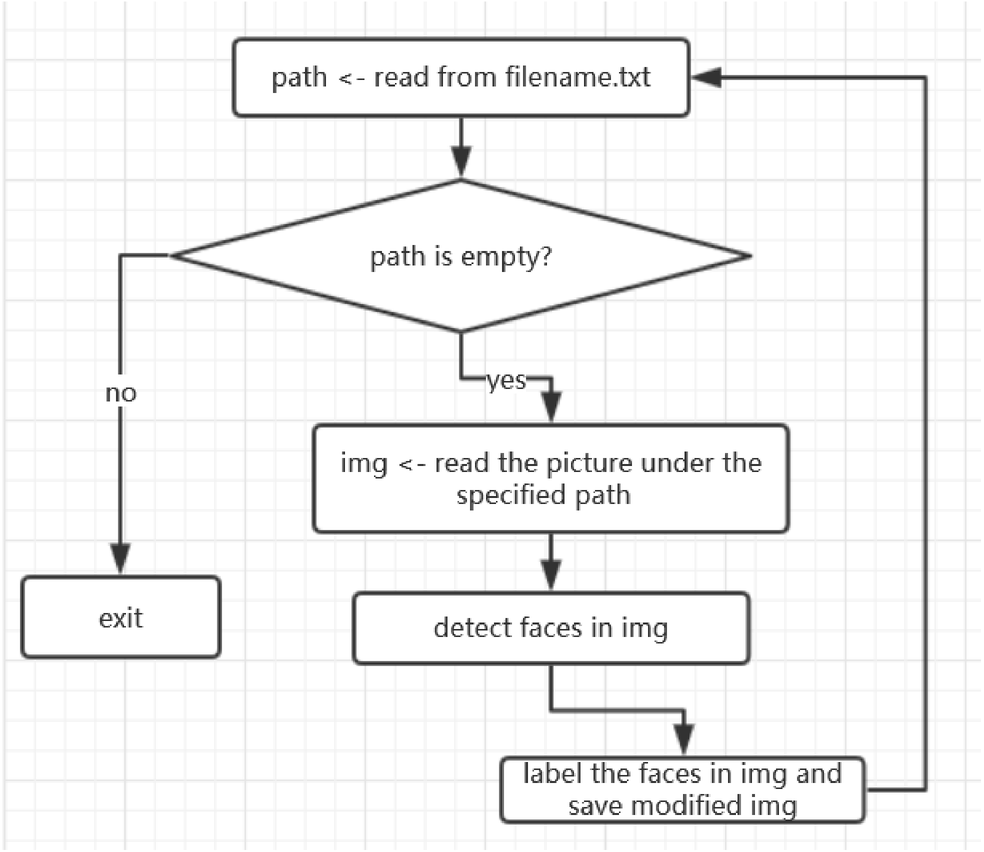
Problem Description

- Tell whether the person in the given picture is smiling.
- Make the computer learn smile detection
- Three subtasks:
  - 1.Face detection
  - 2.Smile classification
  - 3.Realize a real-time Smile Detection application using camera

Problem Analysis

- This is a typical classification problem. We can solve it using traditional machine learning approaches.
- We use LBP to deal with the images of human faces. LBP records the contrast information between the pixel and the surrounding pixels, and describes local texture features.
- We use the eigenvectors that we get from LBP as the training set to build an SVM model, a powerful binary classifier.
- SVM can efficiently perform a non-linear classification using the kernel trick, mapping the input into high-dimensional feature spaces. And using kernel tricks, the calculations in high-dimensional spaces are not so complex.

Subtask I - Face Detection



Important function:
cv2.imread()
cv2.cvtColor()
cv2.CascadeClassifier()
cv2.CascadeClassifier.detectMultiScale()
ImageDraw.Draw.rectangle()

Important resource:
haarcascade_frontalface_default.xml

Subtask I Note

cv2.imread(filename, flags=None):
Loads an image from the specified file and returns it.
The retval is a matrix.
(Empty matrix when the image cannot be read)

cv2.cvtColor(src, code):
Param src --- input image;
Param code --- color space conversion code
Here we use cv2.COLOR_BGR2GRAY as the conversion code to transform the BGR mode into the Gray mode.
And we judge the mode by the image's dimension.
When dim=2, it's gray image; when dim=3, it's BGR.

cv2.CascadeClassifier.detectMultiScale():
This function is used to detect the face. It invokes the classifier haarcascade_frontal-face_default.xml. And the retval is a list of ordered pairs (x,y,w,h). Each pair in the list shows a rectangle area whose top left corner is (x,y) , width is w and height is h.

Then for each rectangle area we get from the function above, we draw it on the image, and the face area in the image is labeled.

Subtask I Code Changes

```
drawFaces('files_of_faces/'+content)
```

```
try:
    drawFaces(in_catalog + '/' + content)
except ValueError:
    error.append(in_catalog + '/' + content)
except RuntimeError:
    error.append(in_catalog + '/' + content)
```

For robustness. When the image cannot be read or there are some other mistakes, we can continue to detect the remains.

```
faces = open('files_of_faces/faces.txt', 'r')
if not os.path.exists('the_frame_of_faces'):
```

```
in_catalog = "files_of_faces"
out_catalog = "the_frame_of_faces_new"

faces = open(in_catalog + '/faces.txt', 'r')
if not os.path.exists(out_catalog):
```

For readability. The paths above appears more than once in the code. So replace the string with variables, making it convenient to modify the input and output path.

Some addition

```
if len(faces):
    count+=1
else:
    nondetected.append(image_name)
```

Record images of faces that we didn't detect.

Subtask I - Running Result

```
3996
3997
3998
3999
4000
共接收4000张照片
共识别3720张照片
识别率：0.93
请按任意键继续. . .
```

Screenshot of console

FaceDetectionReport.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

received 4000 photos
detected 3720 photos
detecting rate: 93.00%

not detected:
files_of_faces/file0016.jpg
files_of_faces/file0019.jpg
files_of_faces/file0031.jpg
files_of_faces/file0065.jpg

Screenshot of the report file

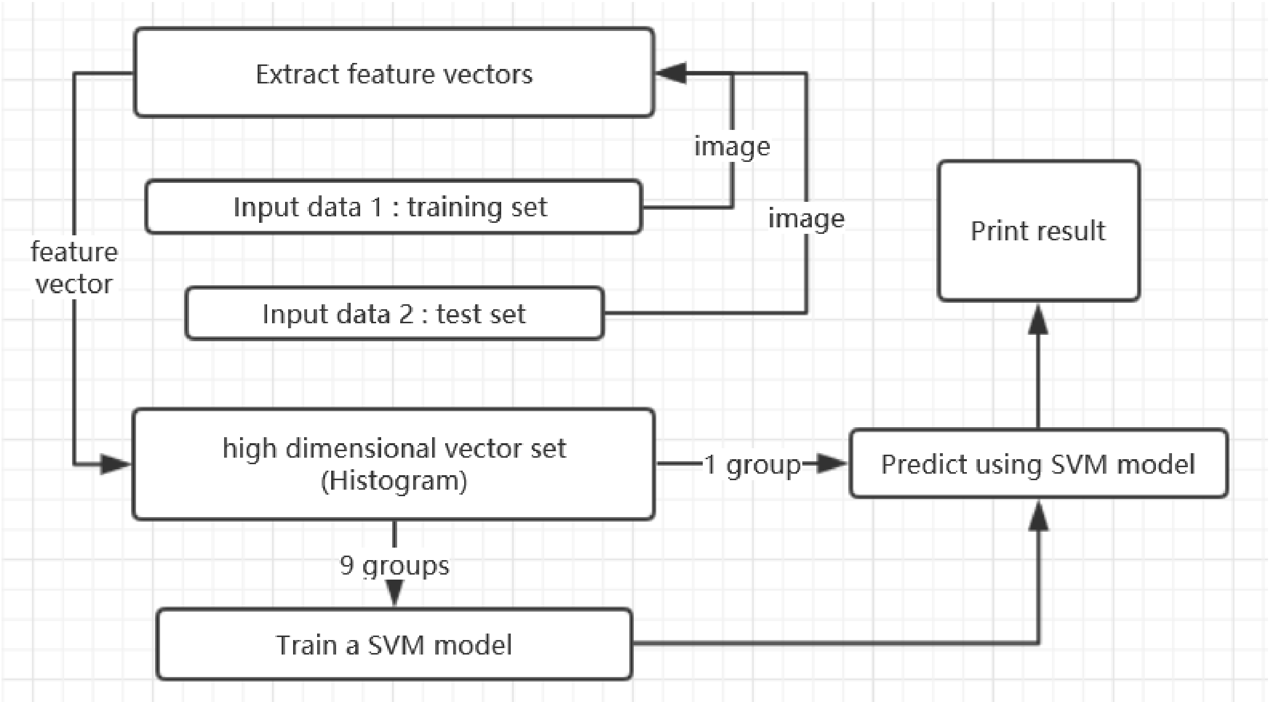
The second parameter of the function below is quite significant.

When it is 1.2 the recognition rate is relatively high.

When it is 1.1, the recognition rate is even higher, but some regions without faces are considered as face regions.

```
faces = face_cascade.detectMultiScale(gray, 1.2, 5)
```

Subtask II – Smile Detection



Important library:
cv2
numpy
sklearn.svm
skimage.feature

Important function:
SVC()
numpy.vstack()
local_binary_pattern()
skimage.feature.hog()

Subtask II - Note

Function	Module	Parameters	Notes
SVC	sklearn. svm	kernel, C, coef0, degree, gamma	1) The parameter “kernel” is the type of the selected kernel function. 2) The parameter “C” is the penalty factor. Generally, the bigger the value of “C” is, the better the classifier works. But when it becomes too large, it might cause over-fitting. 3) When “kernel” is “linear”, the parameters “degree”, “gamma”, “coef0” will be ignored.
fit	sklearn. svm		1) The function is a member function of the class “sklearn. svm. classes. SVC”. it can train a classifier using the given training set and labels. 2) The function has two parameters. The first one represents the training set and the second one represents the given labels.

Subtask II - Note

predict	sklearn. svm		1) The function is also a member function of the class "sklearn. svm. classes. SVC". It can use the trained classifier to predict the result. 2) The function has only one parameter, which represents the test histogram.
vstack	numpy		1) The function is to change a list into a class called numpy. ndarray, which is more suitable for training and predicting. 2) The function has only one parameter whose type is list.
local_binary_pattern	skimage. feature		1) The function is to get the local binary pattern of the picture which is a grayscale image 2) There are four parameters representing the image, the number of selected points, the radius of the selecting circle and the way to get the local binary pattern successively.

Subtask II – The Efficiency Analysis

We found a problem that the reference code is too slow. Why is it so low?

1) The process of extracting feature vectors is slow.
2) The process of merging high dimensional vectors is slow.
3) The process of training SVM model is slow.

The process 1) can be quickened by changing the feature extraction method, for example, if we use hog instead of using local_binary_pattern, it may be quicken largely.

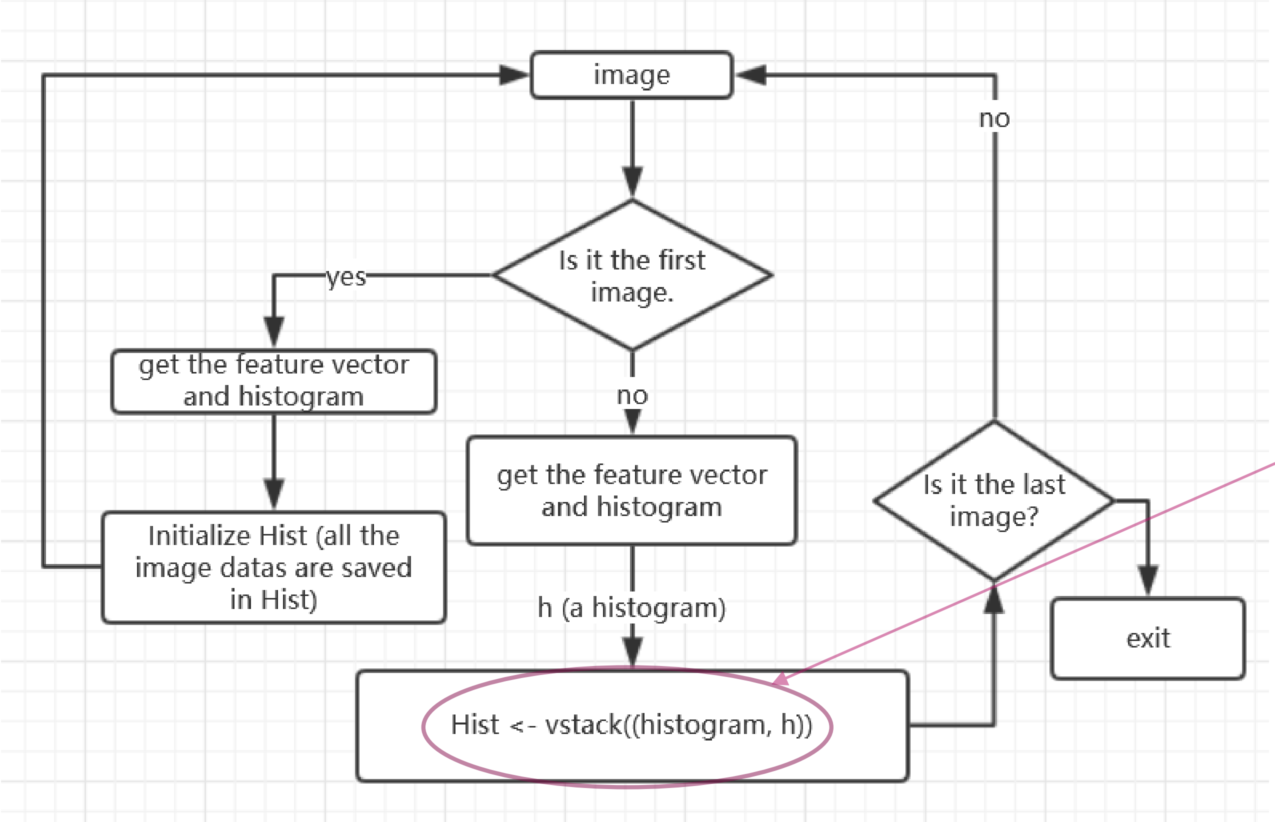
The process 3) can be quickened by reducing the dimensions of the feature vectors. For example, we can change the value of blocks in lbp from 5 to 4, and the process of training and prediction will be much quicker, but the value of F1 may be affected.

Luckily, the process 2) is relatively easy to quicken. The efficiency is limited mainly by the way we use the function “numpy.vstack()”. In the reference code, we do not use the vstack function in a efficient way. And we found that the time consumption of numpy.vstack() is in direct proportion to the sizes or lengths of the vectors that we need to collect.

We show the details below:

Subtask II – The Efficiency Analysis

the old version used by the reference code



The image datas are collected in "Hist".

We assume the number of images is n. Then we have:

$T(n) = T(n-1) + \Theta(n)$

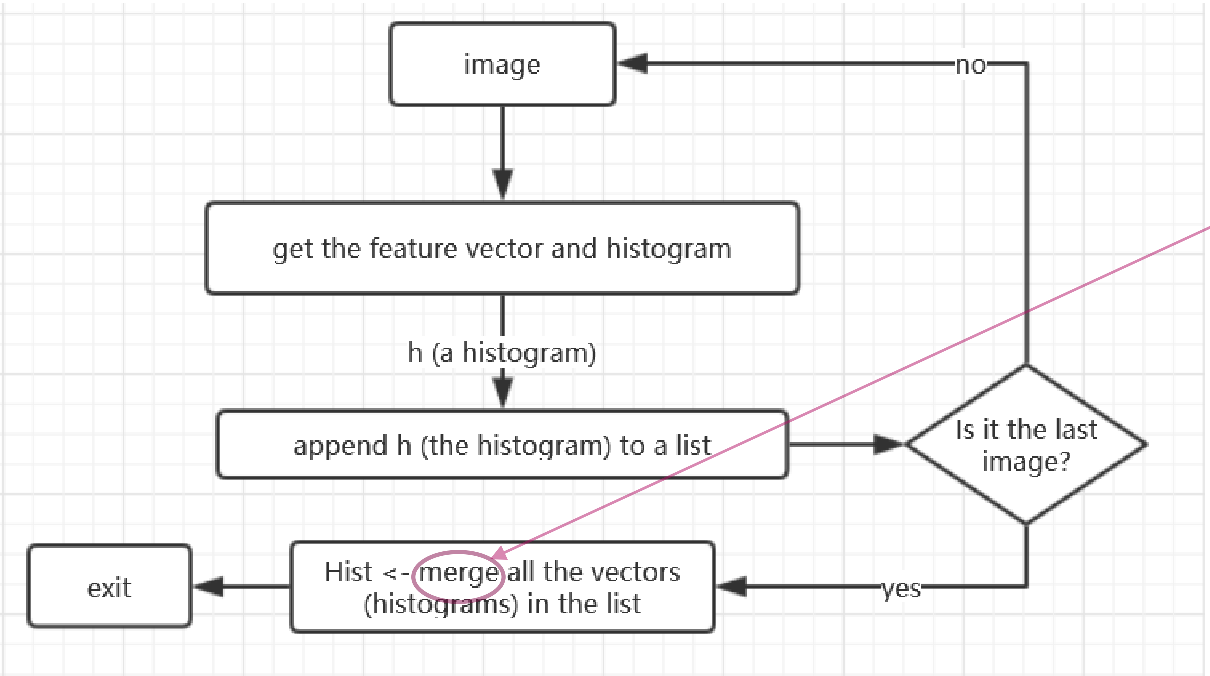
$\Rightarrow \Theta(n^2)$

There is another question: the judgement "Is it the first image?" seems to be not so necessary.

The additional judgement and the process of initializing make the program not so graceful.

Subtask II – The Efficiency Analysis

the improved version used by the our code



The image data set is collected in “Hist”.

The code of the function “merge”

```
def merge(lst):
    Len = len(lst)
    if Len == 1:
        return lst[0]
    M = Len >> 1
    return np.vstack((merge(lst[:M]), merge(lst[M:])))
```

The efficiency analysis:

$$T(n) = 2 \times T\left(\frac{n}{2}\right) + \Theta(n)$$

$$\Rightarrow T(n) = \Theta(n \log n)$$

The process of “merge” is similar to “merge sort” in some degree.

And in this version, the code is more concise and more efficient.
The robustness is surprisingly good.

Subtask II – The Efficiency Analysis

In last page we have mentioned that we use a function called “merge” to speed up the data collection and its efficiency is $\Theta(n \log n)$. The efficiency is great but in fact, the efficiency of the function numpy.vstack has linear relation with the size of the given list. So if we collect all the histograms in a list and use numpy.vstack to turn it into a numpy.ndarray, the efficiency of it is $\Theta(n)$, which is faster.

Hist <- merge all the vectors (histograms) in the list

Version II:

```
[Finished in 735.5s]
```

Version III:

```
[Finished in 732.7s]
```

The two pictures on the right is the time consuming of two versions, but we may find that there is no significant difference.

Subtask II – Parameter Adjustment

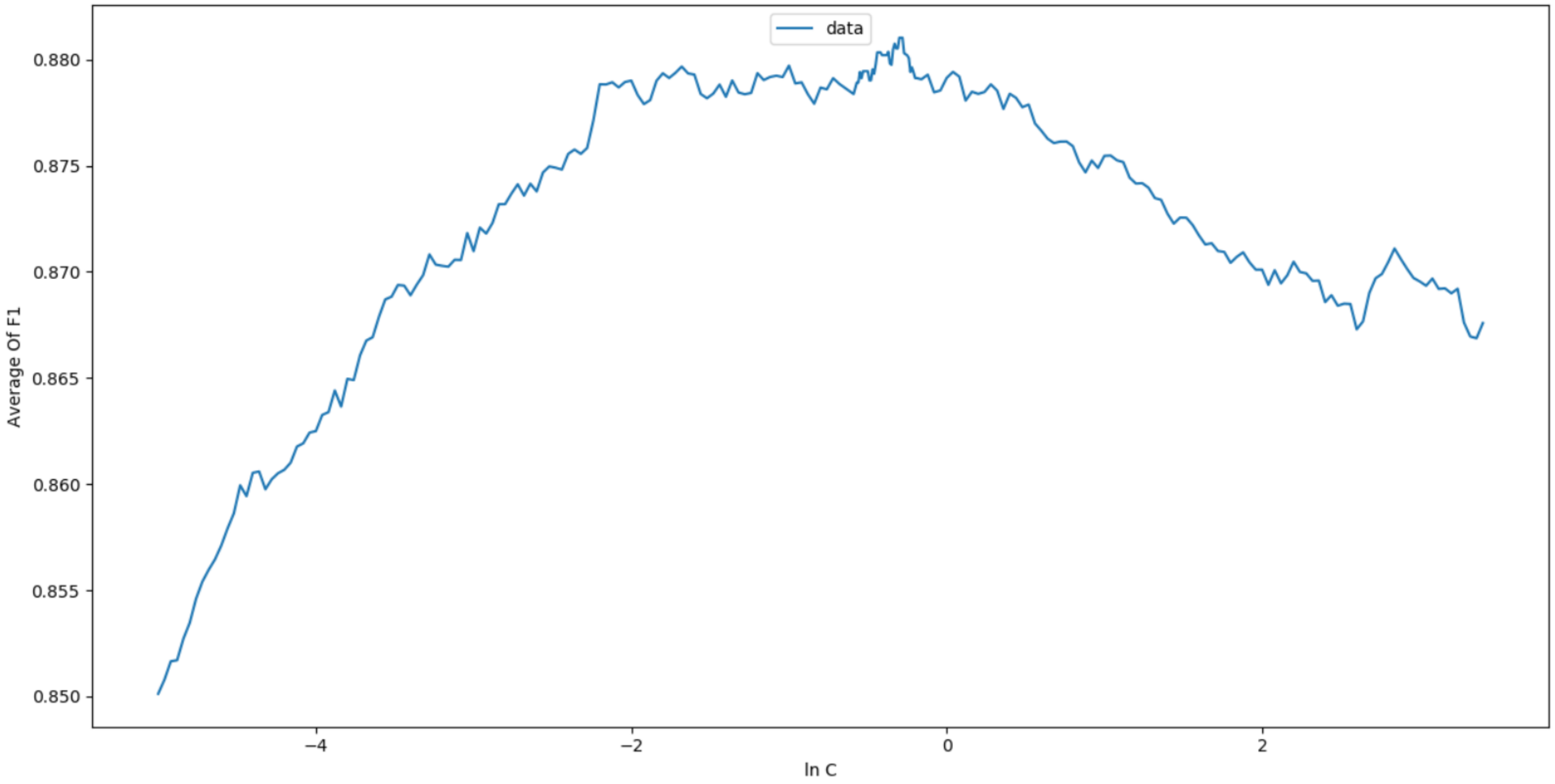
We use local-binary-pattern to extract the feature vectors. We calculate the different values of F1 when the parameter blocks changes. (The blocks means how many blocks we divide the image into.) The datas are shown below:

blocks = 3	blocks = 4	blocks = 5	blocks = 6
group 0, F1: 0.828375	group 0, F1: 0.853012	group 0, F1: 0.872549	group 0, F1: 0.855769
group 1, F1: 0.813559	group 1, F1: 0.855746	group 1, F1: 0.882793	group 1, F1: 0.892768
group 2, F1: 0.848921	group 2, F1: 0.875598	group 2, F1: 0.876238	group 2, F1: 0.904523
group 3, F1: 0.803738	group 3, F1: 0.853081	group 3, F1: 0.836272	group 3, F1: 0.849383
group 4, F1: 0.858491	group 4, F1: 0.893720	group 4, F1: 0.884058	group 4, F1: 0.897059
group 5, F1: 0.864486	group 5, F1: 0.874109	group 5, F1: 0.905569	group 5, F1: 0.910843
group 6, F1: 0.838407	group 6, F1: 0.891509	group 6, F1: 0.883495	group 6, F1: 0.898058
group 7, F1: 0.864608	group 7, F1: 0.906921	group 7, F1: 0.889952	group 7, F1: 0.901205
group 8, F1: 0.838554	group 8, F1: 0.882927	group 8, F1: 0.888350	group 8, F1: 0.888889
group 9, F1: 0.825553	group 9, F1: 0.870416	group 9, F1: 0.891139	group 9, F1: 0.878788

blocks = 6 is relatively good, but the dimension of the vector is higher, and the process of training and prediction is slower

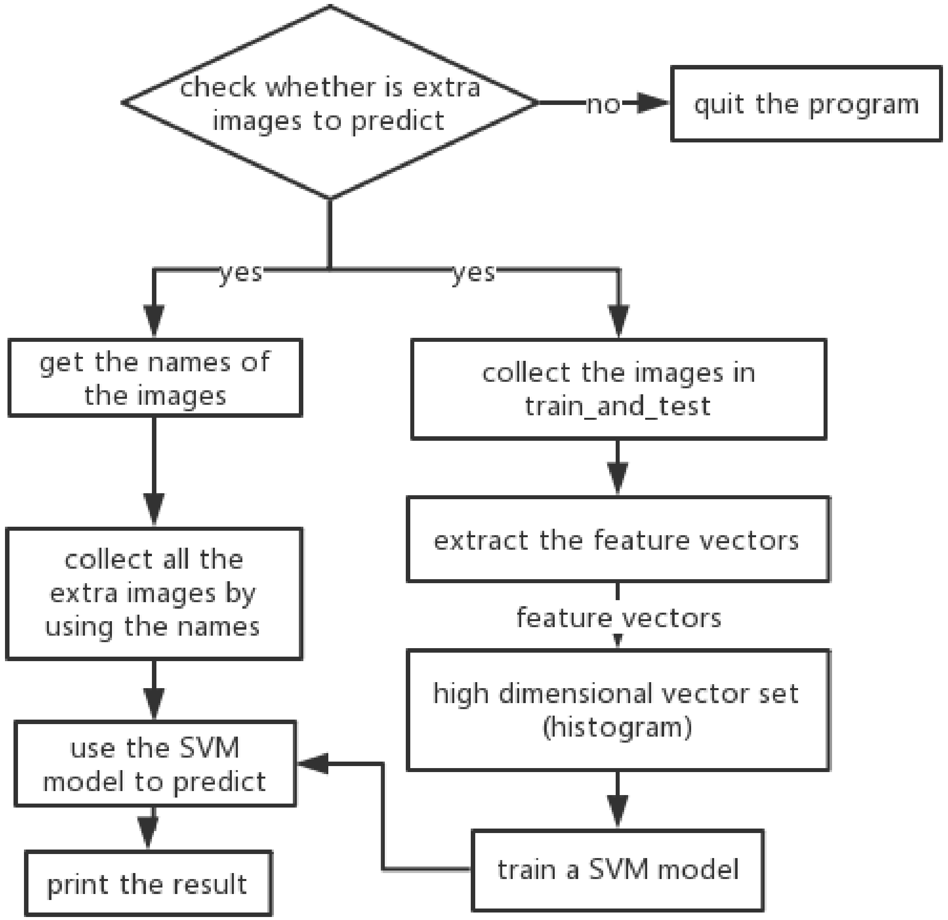
Subtask II – Parameter Adjustment

We also attempt the HOG feature, and we find that using HOG, the dimension of the feature vectors is lower and the process of training and predicting is faster.
We change "C" (penalty factor), and get the average value of F1 under different "C".
The datas are shown below: when $C = e^{-0.28} \approx 0.755784$, the average F1 is the highest.



Subtask II – Additional function

We may find a interesting thing that the program can collect feature vectors from the given pictures, so we can use the collected feature vectors to train the classifier and used the classifier to predict whether a image is a smiling face. Then we have an additional function which can predict your own images. The picture on the right is the progress of the additional function.



Subtask II – Notes of Additional function

To realize the function, we might need the following functions

Function	Module	Parameters	Notes
dirname()	os. path	a string	The parameter is the path of a file, and the function will return the path of the folder which the file belongs.
absphat()	os. path	a string	The parament is a string which is a name of a file. And the function will return the absolute path of the file.
system()	os	a string	The string is a command line, and function will run the command line.
chdir()	os	a string	The given the string is a absolute path of a folder. The function will change the working directory into the given path

Subtask II – Running Result

Comparing with the original version, we have removed all the output of F1 and store then in a text file. Also, we print the running process on the screen which might make the user clear about how the program works

```
*REPL* [python]
Test_time: 0  Train_group: 1  DATA COLLECTION START
Test_time: 0  Train_group: 1  DATA COLLECTION FINISHED
Test_time: 0  Train_group: 2  DATA COLLECTION START
Test_time: 0  Train_group: 2  DATA COLLECTION FINISHED
Test_time: 0  Train_group: 3  DATA COLLECTION START
Test_time: 0  Train_group: 3  DATA COLLECTION FINISHED
Test_time: 0  Train_group: 4  DATA COLLECTION START
Test_time: 0  Train_group: 4  DATA COLLECTION FINISHED
Test_time: 0  Train_group: 5  DATA COLLECTION START
Test_time: 0  Train_group: 5  DATA COLLECTION FINISHED
Test_time: 0  Train_group: 6  DATA COLLECTION START
Test_time: 0  Train_group: 6  DATA COLLECTION FINISHED
Test_time: 0  Train_group: 7  DATA COLLECTION START
Test_time: 0  Train_group: 7  DATA COLLECTION FINISHED
Test_time: 0  Train_group: 8  DATA COLLECTION START
Test_time: 0  Train_group: 8  DATA COLLECTION FINISHED
Test_time: 0  Train_group: 9  DATA COLLECTION START
Test_time: 0  Train_group: 9  DATA COLLECTION FINISHED
Test_group 0 : TEST DATA COLLECTION START
Test_group 0 : TEST DATA COLLECTION FINISHED


Training...


Training finished...

Start detecting...


Test_group 0 Finished!!

Line 157, Column 1                                   Tab
```

Subtask II – Running Result

And all the result is stored in a text file, instead of showing them on the screen. You might find the report by the given path.

```
smile_detection_v2.py    *REPL* [python]

Train_group 2 DATA COLLECTION FINISHED
Train_group 3 DATA COLLECTION START
Train_group 3 DATA COLLECTION FINISHED
Train_group 4 DATA COLLECTION START
Train_group 4 DATA COLLECTION FINISHED
Train_group 5 DATA COLLECTION START
Train_group 5 DATA COLLECTION FINISHED
Train_group 6 DATA COLLECTION START
Train_group 6 DATA COLLECTION FINISHED
Train_group 7 DATA COLLECTION START
Train_group 7 DATA COLLECTION FINISHED
Train_group 8 DATA COLLECTION START
Train_group 8 DATA COLLECTION FINISHED
Train_group 9 DATA COLLECTION START
Train_group 9 DATA COLLECTION FINISHED


Training...


Training finished...

Start detecting...


The Extra Group Finished!!


The test result is stored in "E:\sjtu\Introduction_to_computation\homework\project_2018\project_2018\Smile_Detection\report.txt".

***Repl Closed***
```

Subtask II – Running Result

The following is the report:

report.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

Test_group 0:
F1: 0.874704
TP: 185 ; TN: 137 ; FP: 36 ; FN: 17 ;

Test_group 1:
F1: 0.922306
TP: 184 ; TN: 154 ; FP: 12 ; FN: 19 ;

Test_group 2:
F1: 0.895522
TP: 180 ; TN: 139 ; FP: 18 ; FN: 24 ;

Test_group 3:
F1: 0.859951
TP: 175 ; TN: 136 ; FP: 27 ; FN: 30 ;

Test_group 4:
F1: 0.899263
TP: 183 ; TN: 150 ; FP: 19 ; FN: 22 ;

Test_group 5:
F1: 0.902913
TP: 186 ; TN: 150 ; FP: 21 ; FN: 19 ;

Test_group 6:
F1: 0.883495
TP: 182 ; TN: 146 ; FP: 25 ; FN: 23 ;

Test_group 7:
F1: 0.905213

The following part is the test result of the extra images:
The following images are considered as smiling faces:
0e248ab1c31c7111f7f6a7fbd94ca73e.jpg
267f9e2f07082838ac3a8f1bb899a9014c08f18e.jpg
7b0e46686f2c30723ae93a469752b40c.jpg
c0596680fe0734170ee64d79c854a835.jpg
file0022.jpg

The following images are not considered as smiling faces:
031a930f87f5ecf85a1e8055c69f2270.jpg
084e2b8e3d223839bbe04cc2867bbe7e.jpg
1606529304d066525783775b12564f86.jpg
3c81a8c4aead4a09ebb713d0cd692004.jpg
69853cc9c5d1a31e1005d8075bcc9e9a.jpg
ccb3239d0ba9c4119bea04157a5e06f7923b6661.jpg
d8d9602cd1208103c7170c40ec68bd5d.jpg

But in fact, the result is no good. Because the feature vector is extracted from the whole picture, not the area of faces, so it might cause lots of mistakes.

Subtask II – Questions and Answers

What is training dataset? What is test dataset?

Sol: training dataset is a set of data with labels, the label will be considered as a stander and help to adjust the parameters while training the SVM. The test dataset is also a set of data with labels. But the labels will not change the parameters of a SVM. It’s used to test the accuracy of the SVM model and the labels are just used for statistics and to value the SVM model.

Subtask II – Questions and Answers

What is the feature of an image?

Sol: the feature of a image represent the characteristics of the image in some respects, such as colors, Image gradient or textures. For example, to get the characteristic of texture, people use LBP. It works on the gray level image. The computer divide the image into several parts and get the LBP values of each part by comparing the values of some points in the area. Then it draw a histogram of the appearance frequency of every LBP values and get the feature vectors by concatenating the histogram.

Subtask II – Questions and Answers

How to train SVM?

Sol: We use 9 of the 10 given data groups as the training datas to train the SVM classifier.
The training dataset can be expressed as:
$(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \dots, (\boldsymbol{x_n}, y_n)$ where n $\mathbf{x_i}$ is a real vector, $\mathrm{y_i}$ is the label.
And we consider function: $\left[\frac{1}{n}\sum_i^n \max\left(0, 1 - y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} - b)\right)\right] + \lambda \|\boldsymbol{w}\|^2$
The parameter λ determines the tradeoff between increasing the margin-size and ensuring the $\boldsymbol{x_i}$ lie on the correct side of the margin.
The process of training the SVM is the process of minimizing the function we mentioned above.

Subtask II – Questions and Answers

How to choose kernel functions

Sometimes the situation we facing is not linear classifiable. We map the vectors into higher-dimensional space. And SVM use the kernel trick to simplify the calculation.

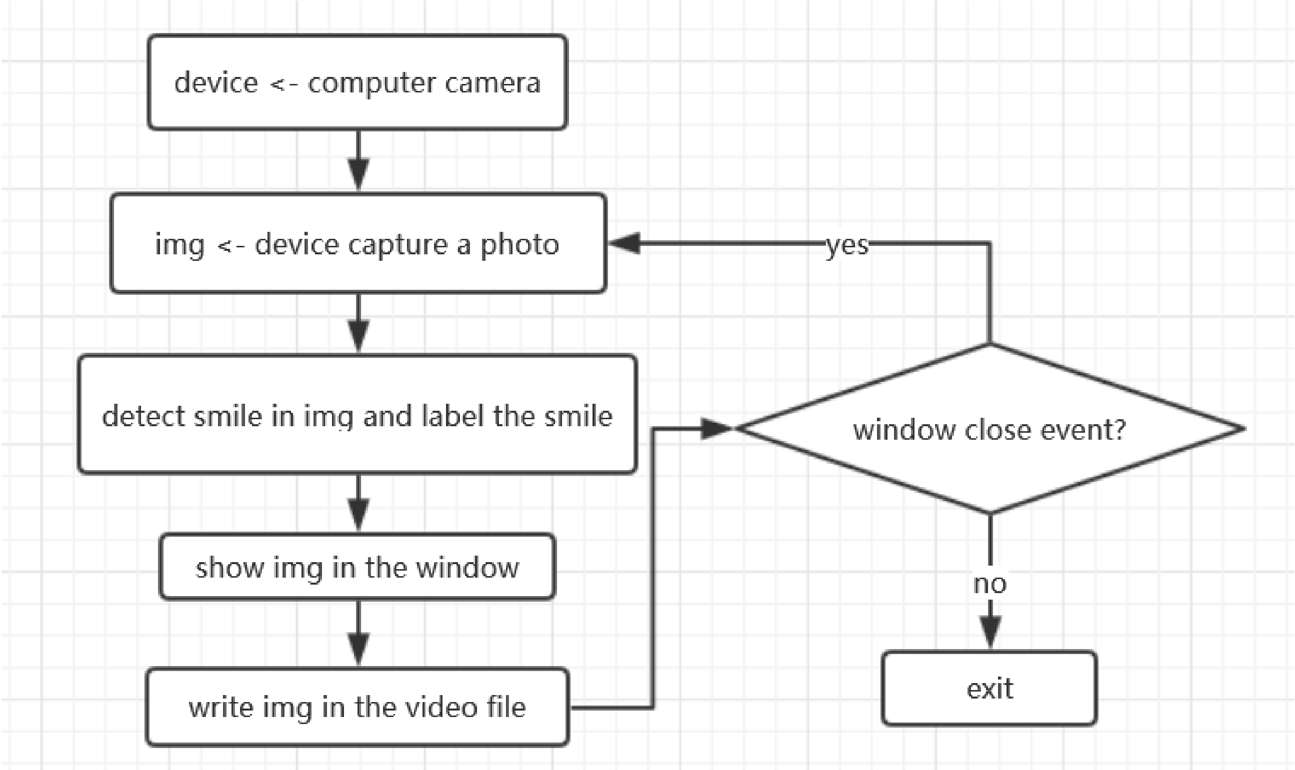
The kernel function $\boldsymbol{k}$ satisfies : $\boldsymbol{x} \rightarrow \varphi(\boldsymbol{x}), \boldsymbol{k}(\boldsymbol{x_i}, \boldsymbol{x_j}) = \varphi(\boldsymbol{x_i}) \cdot \varphi(\boldsymbol{x_j})$

There are several common kernel functions like linear kernel, polynomial kernel, radial basis function kernel (rbf).

To choose the kernel functions , we might follow these tips:

1) If the amount of the features is large and the problem can be considered as an approximate linearly separable problem ,we choose “linear”.
2) If the problem is not linearly separable, and the feature vectors are not high in dimension, we might choose “rbf”
3) If we have no idea about the type of the problem, we can choose “Gaussian kernel”, “Laplace RBF kernel”.
4) “Hyperbolic tangent kernel” and “Sigmoid kernel” are used in neural networks, and “ANOVA radial basis kernel” is used to solve Regression problems.

Subtask III - Real-time Detection



Important library:
cv2

Important function:
cv2.VideoCapture()
cv2.VideoWriter()
cv2.imread()
cv2.imshow()
cv2.waitKey()

Important resource:
haarcascade_smile.xml

Subtask III Note

cv2.VideoCapture(para1):
When para1 is 0, it invokes the built-in camera.
When para1 is a path of a video file, it invokes the file.

cv2.imshow(para1,para2):
para1(a string):Name of the window.
para2(a matrix):Image to be show.
It will show the specified image in the assigned window.
This function should be followed by cv2.waitKey().
Otherwise, it won't display the image.

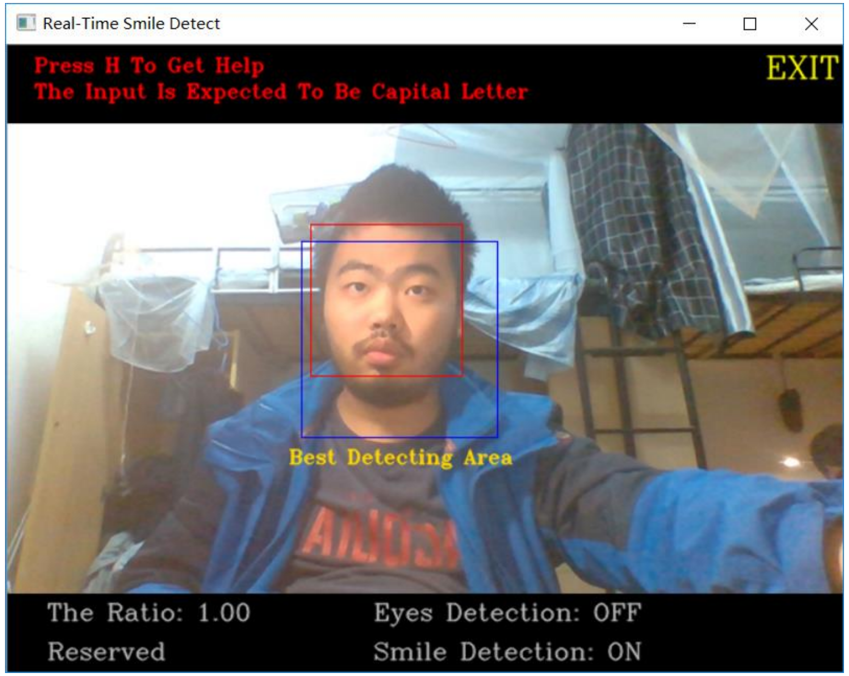
cv2.waitKey(para):
When pata is 0, it will display the image infinitely until pressing a key.
When pata is greater than 0, it will display the image for para ms, after which the display will be automatically closed.
If we put this function into a loop, we can show the video frame-by-frame.

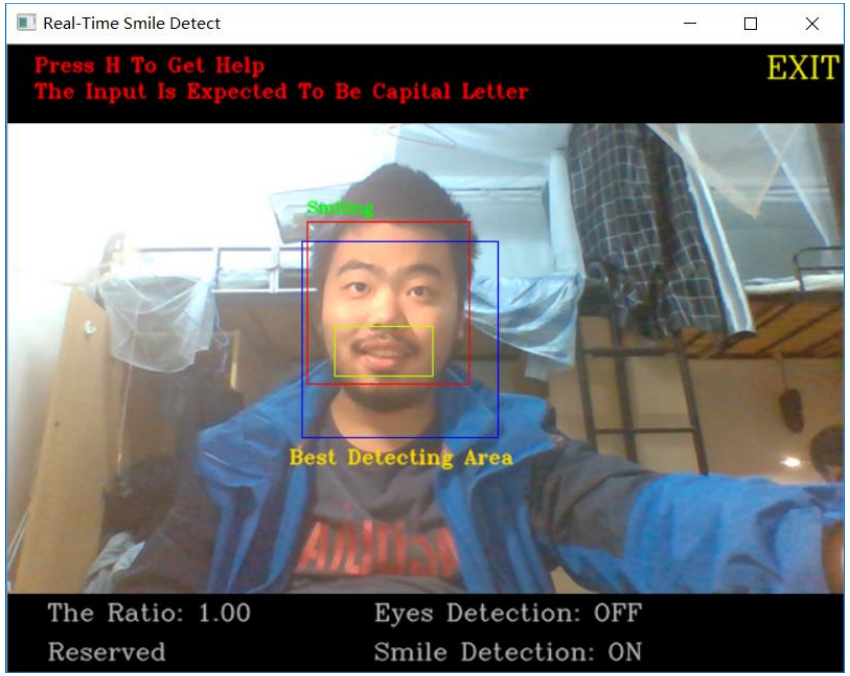
The retval of cv2.waitKey():

When we do not press any key, it returns -1 (or we can consider it as 255, the complement code of -1)

When we press a key, it returns the ASCII code (0~255).

Subtask III – Real-time Detection(1)





[source file]

Subtask III – Code Changes

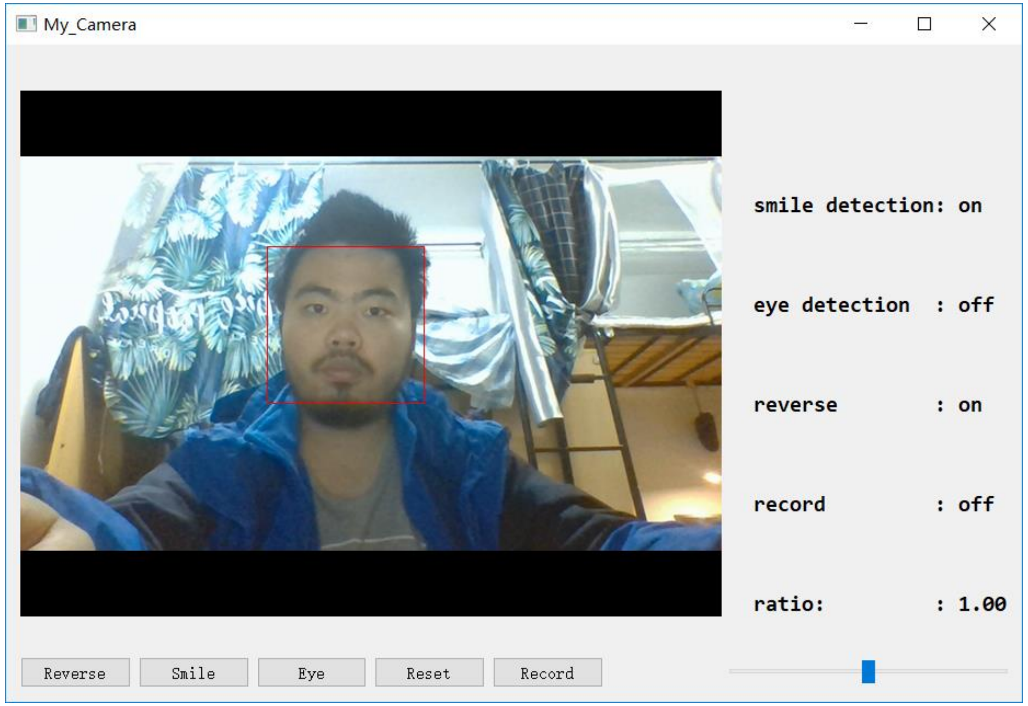
We made some small changes on the basis of the reference code:

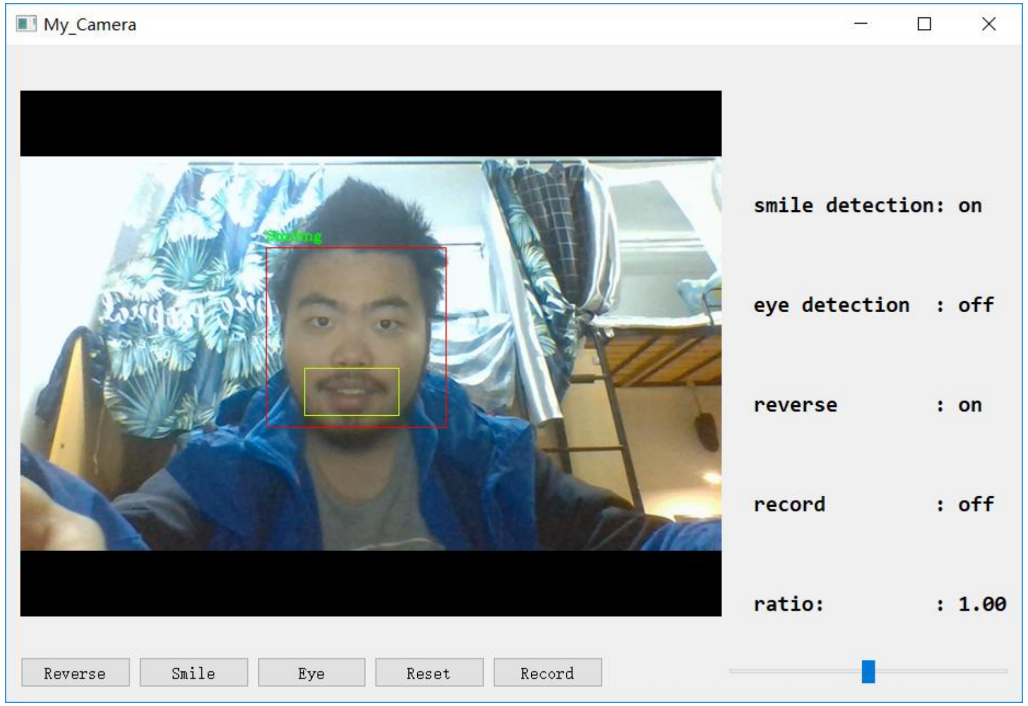
1. We changed the video format from .mp4 to .avi, and changed MP4V into XVID correspondingly, just to avoid the warnings reported by the interpreter.
2. We added a line of codes “out.release()” at the end of the code. Just to avoid such situation that we cannot delete or invoke the output file “output.avi” after running the real-time detection script.
3. We added a function to detect eyes. And it can be switched on/off by pressing ‘E’.
4. We added a function to scale the image. We realize this function by cv2.warpAffine() using the transfer matrix.
5. We added a function to reverse the image, making our camera more similar to a mirror, and bringing better using experiences to users.

Subtask III – Some Problems For Version1

- The program cannot be stopped by clicking the quit botton. It can only be closed by pressing keys.
- We can use “mouse_event” offered by cv2 to receive the message sent by the mouse.
- And we can set an area in the window. When the user click this area, the program ends.
- But there is still a problem, when we click the outside edge of the window, the “mouse_event” will not be triggered.
- So we come up with an idea: we can use “pyqt5”.

Subtask III – Real-time Detection(2)





source file

In this version, the interface is more friendly, and more pleasant to see. And we can click the botton instead of pressing keys.

Subtask III – Version2

Function(or Improvement)	Realize(codes and details)
Click the cross to quit	Use the pyqt5 widget
Elegant window layout	Use the grid layout to design
Switch on/off functions by clicking buttons	Use Qt's signals and slots
Inquire whether save the video or not when quitting	Rewrite the close_event, and use the QMessageBox
Scale the image and reverse the image	Use transfer matrix. Use cv2.warpAffine() and cv2.flip()
Improve the accuracy rate of smile detection	Adjust the parameters. Change the fixed minimum detecting size into a variable which changes with the size of detected faces.