浙江大学

计算机视觉(本科)作业报告

作业名称：	HW#3: Eigenface 人脸识别算法
姓　　名：	高铭健
学　　号：	3210102322
电子邮箱：	1473760386@qq.com
联系电话：	19550212596
指导老师：	宋明黎

2022 年 12 月 25 日

一、 实验实现的功能简述及运行说明

二、 作业的开发与运行环境

三、 系统或算法的基本思路、原理、及流程或步骤等

四、 具体如何实现，例如关键（伪）代码、主要用到函数与算法等

五、 实验结果与分析

六、 结论与心得体会

七、 参考文献

1. 实验实现的功能简述及运行说明

1.1 实验实现功能

1. 每张人脸图像只有一张人脸，且两只眼睛位置已知。每张图像的眼睛位置存在相应目录下的一个与图像文件名相同但后缀名为txt的文本文件里，分别对应于两只眼睛中心在图像中的位置
2. 实现了两个程序过程，分别对应训练与识别
3. 训练程序为：“train.exe 能量百分比 model文件名”，用能量百分比决定取多少个特征脸，训练结果输出保存到model文件中 。同时将前10个特征脸拼成一张图像，然后显示出来
4. 识别程序为：“test.exe 人脸图像文件名 model文件名 ”，将model文件装载进来后，对输入的人脸图像进行识别，并将识别结果叠加在输入的人脸图像上显示出来，同时显示人脸库中跟该人脸图像最相似的图像。

1.2 运行说明

- 训练程序运行：

  在命令行中输入：`path to folder\exe\train.exe train_number ./model.txt`，

  `path to folder`表示到当前提交文件夹的路径；`train_number`使用特征向量的数量

  （例如D:\study\CV\3210102322_Gaomingjian\exe\train.exe 100 ./model.txt）表示运行训练程序保存100个特征向量到当前文件夹的model.txt文件

- 识别程序运行：

  在命令行中输入：`path to folder\exe\test.exe test_image ./model.txt`，

  `path to folder`表示到当前提交文件夹的路径；`test_image`表示测试的图片；`./model.txt`表示特征向量所在的文件

  （例如D:\study\CV\3210102322_Gaomingjian\exe\train.exe ./test/me9.jpg ./model.txt）表示运行识别程序，测试当前文件夹test文件夹中的me9.jpg文件特征向量位于当前文件夹model.txt文件中

- 运行的结果：

  人脸集存放在`train`文件夹中

  对齐后的图片存放在`output_align`文件夹中

  归一化后的图片存放在`output_normalization`文件夹中

  平均化的图片存放在`output_average`文件夹中

  要测试的图片存放在`test`文件夹中

  前10张特征脸存放在`train_result`文件夹中

2.作业的开发与运行环境

- win11 系统
- 使用python + Opencv库

3. 主要用到的函数与算法

3.1 Eigenface人脸识别算法

Eigenface人脸识别算法：一种常用的基于特征投影方法的人脸识别技术。主要思想是使用主成分分析（PCA）来提取人脸图像集的主要特征，并利用这些特征来表示和识别不同的人脸。算法主要步骤如下：

1. 构造训练库：用一组图像作为训练库，并且给出眼睛的位置坐标用于对齐。
2. 预处理：对每个图像进行预处理，首先确定模板，根据人脸两只眼睛的中心位置，进行缩放/平移/旋转等使所有训练人脸图像与模板对齐。然后根据模板，切出脸部区域，最后对灰度值做归一化，对图像进行直方图均衡化和直方图拉伸
3. 特征提取：计算图像协方差矩阵，根据协方差矩阵求处对应的特征值与相应的归一特征向量
4. 识别：对于待识别的人脸图像，将其投影到与训练图像相同的特征空间中，得到该图像的特征向量。利用之前训练得到的特征向量来判断这个特征向量所属的类别，从而实现人脸识别。

3.2 其他函数

使用的其他库函数主要有：

- `cv2.warpAffine(src, M, dsize[, dst[, flags[, borderMode[, borderValue]]]])`：

  用于对图像进行仿射变换：

  主要参数如下：

  - `src`：输入图像
  - `M`：变换矩阵，包括移动的数量
  - `dsize`：输出图像的宽度和高度

- `cv2.equalizeHist(src[, dst])`：用于图像的直方图均衡化

  用于对图像进行仿射变换：

  主要参数如下：

  - `src`：输入图像
  - `dst`：输出图像

4. 实验步骤及代码具体实现

4.1 编写训练函数代码

- 将用到的 opencv 等库导入进来：

```python
import os
import cv2
import sys
import numpy as np
import matplotlib.pyplot as plt
```

- 读取模板图像和眼睛的坐标：

  使用`cv2.imread`函数读取模板图像，用`cv2.IMREAD_GRAYSCALE`参数将图像以灰度方式加载。

  使用`open`函数打开模板眼睛坐标文件，用`readlines()`读取文件内容。选取第二行进行处理计算模板图像中眼睛中心点的坐标

  最后，根据眼睛坐标计算出人脸矩形的位置和大小

```python
# 读取模板图像和眼睛的坐标
    template_image = cv2.imread(template_image_path, cv2.IMREAD_GRAYSCALE)
    with open(template_txt_path, 'r') as f:
        line = f.readlines()[1] left_eye_x, left_eye_y, right_eye_x,
        right_eye_y = map(int, line.split('\t'))

    template_eye_center = ((left_eye_x + right_eye_x) / 2,
                  (left_eye_y + right_eye_y) / 2)
    # 计算模板图像中人脸矩形的位置和大小
    face_rect = (left_eye_x - 200, left_eye_y - 100, 130, 170)
```

- 图片预处理：
  - 函数`align_face_with_template`对人脸图像进行对齐：

    读取待处理图片眼睛位置，计算水平和垂直方向上的平移量，以使待处理图片的眼睛中心与模板图像的眼睛中心对齐，使用`cv2.getRotationMatrix2D`生成旋转变换矩阵，使用`cv2.warpAffine`函数，对待处理的人脸图像进行仿射变换，得到对齐后的人脸图像。
  - 遍历训练图片：用`align_face_with_template`函数进行对齐，使用`cv2.equalizeHist`和`cv2.convertScaleAbs`函数函数和将对齐后的图片进行直方图均衡化和直方图拉伸，将处理后的图像重采样为指定大小

```python
def align_face_with_template(img, txt_path, template_eye_center):
    with open(txt_path, 'r') as file:
        txt_data = file.readlines()[1].split('\t')
        img_template_left_eye_x, img_template_left_eye_y,
        img_template_right_eye_x, img_template_right_eye_y = map(int,
txt_data)
    eye_center = ((img_template_left_eye_x + img_template_right_eye_x) //
2,
                  (img_template_left_eye_y + img_template_right_eye_y) //
2)
    dx = template_eye_center[0] // 2 - eye_center[0]
```

```
    dy = template_eye_center[1] // 2 - eye_center[1]
    scale = 1.0
    M = cv2.getRotationMatrix2D(eye_center, 0, scale)
    M[0, 2] += dx  # 在变换矩阵中添加平移量
    M[1, 2] += dy
    aligned_face = cv2.warpAffine(img, M, (img.shape[1], img.shape[0]))

    return aligned_face

    for image_file in os.listdir("./train"):
        if image_file.endswith(".pgm") or image_file.endswith(".jpg"):
            image_path = os.path.join("./train", image_file)

            image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
            if image_path.endswith(".pgm"):
                txt_path = image_path.replace(".pgm", ".txt")
            else:
                txt_path = image_path.replace(".jpg", ".txt")
            aligned_image = align_face_with_template(image, txt_path,
                                                     template_eye_center)
            output_path = os.path.join(output_folder,
f"aligned_{image_file}")
            template_face = extract_face_region(aligned_image, face_rect)
            cv2.imwrite(output_path, template_face)
            # print(f"Aligned image saved to: {output_path}")
            # 直方图均衡化和直方图拉伸
            equalized_image = cv2.equalizeHist(aligned_image)
            stretched_image = cv2.convertScaleAbs(equalized_image,
alpha=1.5,
                                                  beta=0)
            template_face2 = extract_face_region(stretched_image,
face_rect)
            output_path_nor = os.path.join(output_folder2,
f"nor_{image_file}")
            resized_img = cv2.resize(template_face2, (65, 85),
                                     interpolation=cv2.INTER_AREA)
            cv2.imwrite(output_path_nor, resized_img)
            images_array.append(resized_img)
```

- 计算人脸图像集的平均脸和差分脸：
  - 使用 `flatten()` 方法将图像转换为转换为一维数组构造所有图像的一维数组矩阵
  - 用 `np.mean` 函数计算 `all_faces` 矩阵的均值，得到平均脸 `average_face`
  - 遍历 `all_faces` 矩阵中的每个人脸向量 `face`。计算与平均脸的差值，得到差分脸

```python
# 获得视频的属性（长、宽、帧率）
video_path = os.path.join(input_folder, video_file)
video = cv2.VideoCapture(video_path)
video_width = int(video.get(cv2.CAP_PROP_FRAME_WIDTH))
video_height = int(video.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = video.get(cv2.CAP_PROP_FPS

# 遍历图像调整图像的尺寸
resized_images = []
for image_file in image_files:
    image_path = os.path.join(input_folder, image_file)
    img = cv2.imread(image_path)
    resized_images.append(cv2.resize(img, (video_width, video_height)))
```

- 计算协方差矩阵，计算其中的特征向量和特征值，展示了前10个特征向量的图像。
  - 遍历每个差分脸向量，得到协方差矩阵，使用`np.linalg.eig`计算协方差矩阵的特征值`eigenvalues`和特征向量`eigenvectors`。
  - 对特征值排序降序排列，基于特征值的实部进行排序，保存前N个特征向量：
  - 遍历前10个特征向量。将每个特征向量转为2D图像，并进行归一化处理，最后显示这些图像

```python
# 计算协方差矩阵
    cov_matrix = np.zeros((height * width, height * width))
    for delta_face in difference_faces:
        cov_matrix += np.outer(delta_face, np.transpose(delta_face))
    cov_matrix = cov_matrix / len(difference_faces)
    eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)
    # 选择特征值最大的前N个特征向量作为主要成分
    eigenvalues_sort = []
    for i in range(height * width):
        eigenvalues_sort.append((eigenvalues[i].real, i))
    eigenvalues_sort.sort(reverse=True, key=lambda e: e[0])
    v = []
    # 保存前N个特征向量
    v = [eigenvectors[:, eigenvalues_sort[i][1]].real for i in range(0, N)]
    np.savetxt('./model.txt', np.array(v), fmt="%s")
    v_img = []
    # 取前十个特征向量
    plt.figure()
    for i in range(0, 10):
        v_img.append(np.array(v[i]).reshape(height, width))
        cv2.normalize(v_img[i], v_img[i], 0, 255, cv2.NORM_MINMAX)
        plt.subplot(2, 5, i + 1)
        plt.imshow(v_img[i].astype('uint8'), cmap=plt.cm.gray)
        plt.xticks(())
        plt.yticks(())
    plt.savefig("./train_result/top_eigenvectors.jpg")
    plt.show()
```

4.2 编写识别函数代码

- 识别函数：
  - 图像预处理：将输入人脸图像转换为灰度图像，然后进行直方图均衡化，调整图像到指定的`(65, 85)`
  - 计算特征向量：将经过预处理的图像变为一维数组，与模型数据进行矩阵乘法，得到输入图像的特征向量
  - 查找最相似图像：遍历人脸库，对于每个人脸图像，将预处理后的人脸图像映射到模型数据得到特征向量计算其和输入特征向量`input_vector`之间的欧氏距离。
  - 返回结果：返回欧式距离最小的图片，同时返回其和与原图片叠加后的图像

```
def recognize_face(face_image, model_data, face_database_folder):
    # 将输入人脸图像与模型数据进行点积计算，得到特征向量
    input_gray = cv2.cvtColor(face_image, cv2.COLOR_BGR2GRAY)
    input_equalized = cv2.equalizeHist(input_gray)
    input_image_stretched = cv2.convertScaleAbs(input_equalized,
                                                alpha=1.5, beta=0)
    resized_img = cv2.resize(input_image_stretched, (65, 85),
                             interpolation=cv2.INTER_AREA)
    height = 85
    width = 65
    test_array = resized_img.reshape((height * width), 1).astype('float64')
    input_vector = np.matmul(model_data, test_array)

    # 初始化最小距离和最相似图像路径
    min_distance = float('inf')
    most_similar_image_path = ''

    # 遍历人脸库中的每个人脸特征向量
    for face_path in glob.glob(face_database_folder + '/*.pgm') +
    glob.glob(face_database_folder + '/*.jpg'):
        face = cv2.imread(face_path)
        face_gray = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
        face_equalized = cv2.equalizeHist(face_gray)
        face_image_stretched = cv2.convertScaleAbs(face_equalized,
                                                   alpha=1.5, beta=0)
        resized_img = cv2.resize(face_image_stretched, (65, 85),
                                 interpolation=cv2.INTER_AREA)
        face_array = resized_img.reshape((height * width),
1).astype('float64')
        face_vector = np.matmul(model_data, face_array)
        distance = np.linalg.norm(face_vector - input_vector)
        # 如果找到更近的距离则更新最小距离和最相似图像路径
        if distance < min_distance:
            min_distance = distance
            most_similar_image_path = face_path
            add_image = cv2.addWeighted(input_gray, 0.5, face_gray, 0.5, 0)
    print(most_similar_image_path)
    most_similar_image = cv2.imread(most_similar_image_path)
```

```
38      return most_similar_image, add_image
```

4.3 结果测试

- 创建.spec配置文件，输入`pyinstaller -F test.py`命令生成识别程序.spec配置文件和可执行文件，如下图所示：

```
PS D:\study\CV\hw3_3210102322> pyinstaller -F test.py
1042 INFO: PyInstaller: 6.2.0
1042 INFO: Python: 3.11.5
1052 INFO: Platform: Windows-10-10.0.22621-SP0
```

```
113863 INFO: Embedding manifest in EXE
113947 INFO: Appending PKG archive to EXE
115681 INFO: Fixing EXE headers
120346 INFO: Building EXE from EXE-00.toc completed successfully.
```

- 创建.spec配置文件，输入`pyinstaller -F train.py`命令生成训练程序.spec配置文件和可执行文件，如下图所示：

```
PS D:\study\CV\hw3_3210102322> pyinstaller -F train.py
537 INFO: PyInstaller: 6.2.0
537 INFO: Python: 3.11.5
550 INFO: Platform: Windows-10-10.0.22621-SP0
551 INFO: wrote D:\study\CV\hw3_3210102322\train.spec
```

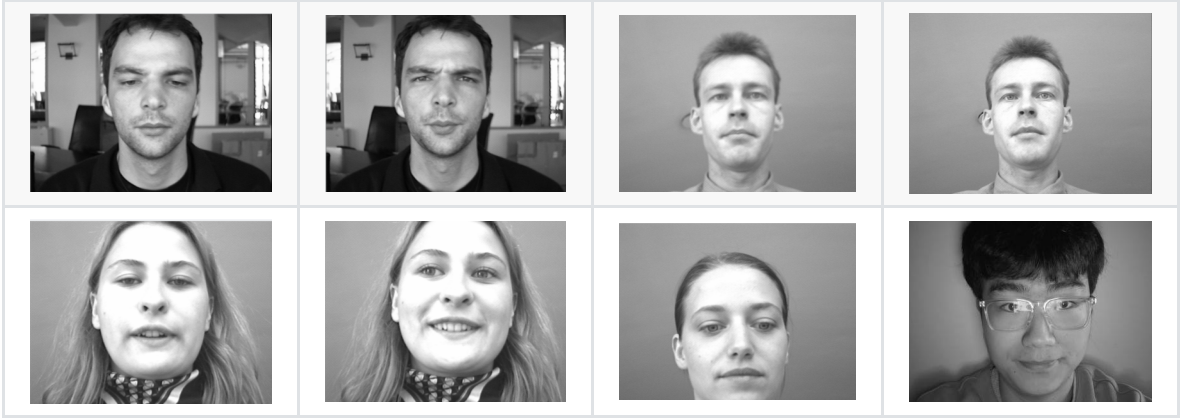
```
113947 INFO: Appending PKG archive to EXE
115681 INFO: Fixing EXE headers
120346 INFO: Building EXE from EXE-00.toc completed successfully.
```

- 输入`./exe/train.exe 100 ./model.txt`运行生成出的训练程序可执行文件
- 输入`./exe/test.exe 100 ./model.txt`运行生成出的识别程序可执行文件

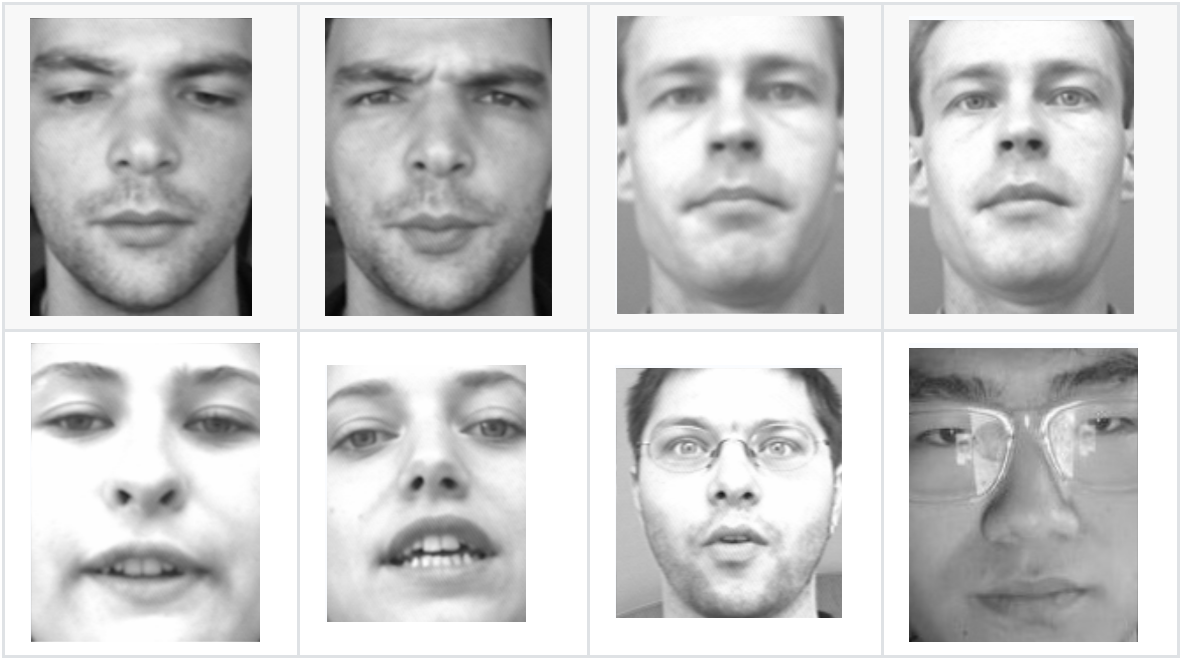
5. 实验结果与分析

5.1 训练程序结果

- 训练集中的部分人脸图片：



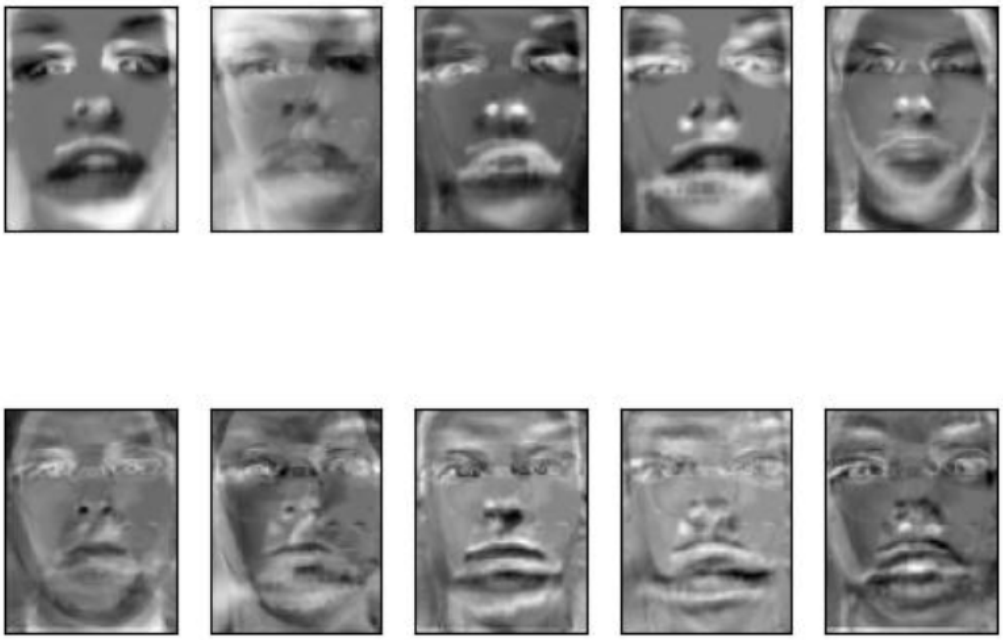
- 对齐后的部分图片：



- 归一化后的图片效果：

- 平均的人脸：

- 前10个特征向量的人脸图像：



- 保存的特征向量（model.txt，设置为100个）

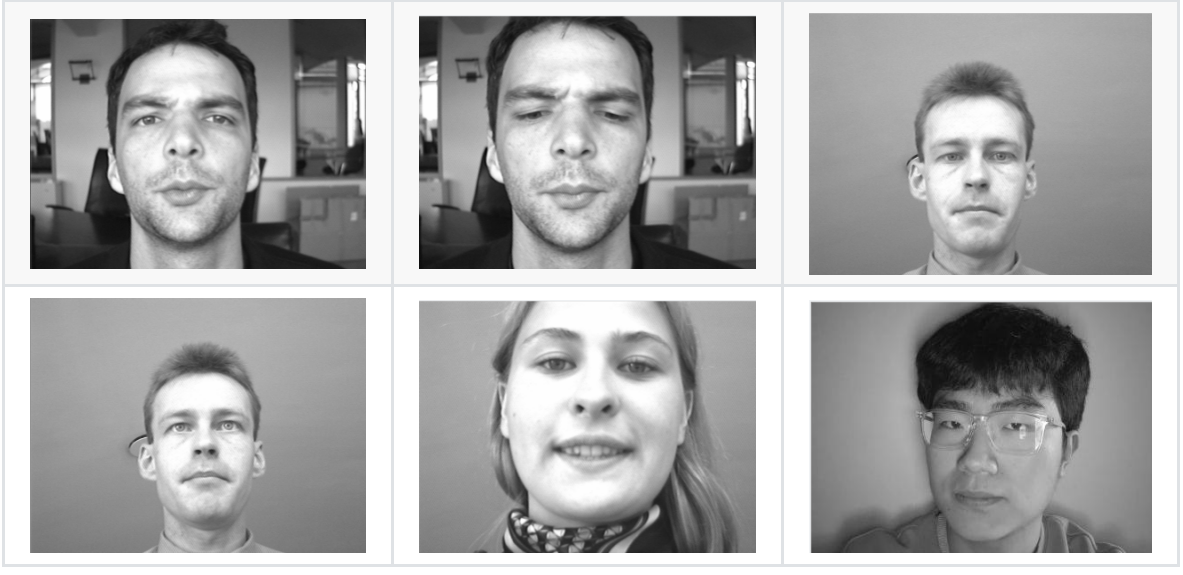
test.py × train.py × model.txt ×

The file size (11.92 MB) exceeds the configured limit (2.56 MB). Code insight features are not available.

```
1  0.019844336619358114 0.024221354343935183 0.024022453933951058 0.020199318085827457 0.021903132107376936 0.018960978722568992
   0.012524668685645541 0.005303600053134907 0.004445237855659932 0.0022337301665493596 -0.0001616130490579761 -0
   .0003414777462369375 -0.0003576956946646873 -0.0003586198838996755 -0.00038643840446116884 -0.00040183504086149286 -0
   .00047859586067084797 -0.0004690146678677401 -0.00047778202892040215 -0.000370407034850458 -0.00030540995813153744 -0
   .0002515854075796414 -0.00018359175418525297 -0.00010026588028278661 -5.669573950114483e-05 -2.815739159279992e-05 -3
   .5904118487393264e-05 -3.5923923200162017e-05 -1.98264381387907 44e-05 -4.457430860615585e-05 2.88061155958233e-06 1
   .5489648140864377e-05 1.5124623568475733e-05 8.07965937608899e-06 1.0375988406523821e-05 4.4200562581430995e-05 4
   .005676714252734e-05 2.8548162506583485e-05 -2.7775697313855127e-05 -0.00014131824872378409 -0.00023260690072618987
   -0.00018206667714040618 -0.00019273676114697525 -0.00033037443623376133 -0.0003334822853584136 -0.0005547827884873804
   -0.0005166885467436707 -0.0004629473365760733 -0.0003948926366141227 -0.0002751053032152841 -0.00016904809606995498
   -0.00017262736643083984 -0.00018815073742310226 0.00015651183418737368 0.0006315994318795305 0.001812688576028829 0
```

5.2 识别程序结果

- 测试集中的部分人脸图片：



- 测试的结果：
  - 如下图，命令行中输出的是对应训练集中图片的路径，也就是当前图片“0012”对应的是图片“0003”

    ```
    PS D:\study\CV\hw3_3210102322> ./exe/test.exe ./test/BioID_0012.pgm ./model.txt
    ./train\BioID_0003.pgm
    ```

    如下图，最左边是输入图像，中间是叠加后的图像，右边是寻找到的训练集中的图片

  - 如下图，命令行中输出的是对应训练集中图片的路径，也就是当前图片“0037”对应的是图片“00034”

    ```
    PS D:\study\CV\hw3_3210102322> ./exe/test.exe ./test/BioID_0037.pgm ./model.txt
    ./train\BioID_0034.pgm
    ```

    如下图，最左边是输入图像，中间是叠加后的图像，右边是寻找到的训练集中的图片

  - 如下图，命令行中输出的是对应训练集中图片的路径，也就是当前图片“0061”对应的是图片“0058”

    ```
    PS D:\study\CV\hw3_3210102322> ./exe/test.exe ./test/BioID_0061.pgm ./model.txt
    ./train\BioID_0058.pgm
    ```

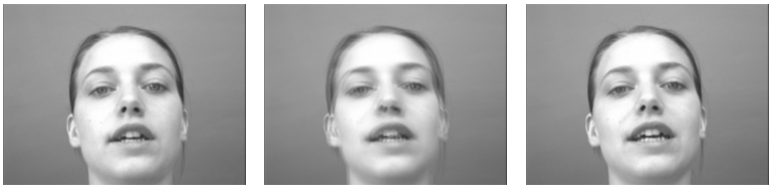
    如下图，最左边是输入图像，中间是叠加后的图像，右边是寻找到的训练集中的图片

  - 如下图，命令行中输出的是对应训练集中图片的路径，也就是当前图片“0087”对应的是图片“0084”

    ```
    PS D:\study\CV\hw3_3210102322> ./exe/test.exe ./test/BioID_0087.pgm ./model.txt
    ./train\BioID_0084.pgm
    ```

    如下图，最左边是输入图像，中间是叠加后的图像，右边是寻找到的训练集中的图片



- 如下图，命令行中输出的是对应训练集中图片的路径，也就是当前图片“me9”对应的是图片“me8”

```
PS D:\study\CV\hw3_3210102322> ./exe/test.exe ./test/me9.jpg ./model.txt
./train\me8.jpg
```

如下图，最左边是输入图像，中间是叠加后的图像，右边是寻找到的训练集中的图片

6. 心得体会

在本次实验中，我遇到了很多问题，在反复学习和调试后解决了一些问题，比如：

- 在进行协方差矩阵的计算时，由于训练集的图像很大，一开始我总是无法创建这么大的矩阵，最后只能不断缩小图像找到能创建的最大程度
- 在特征向量和特征值的计算时，总是会遇到矩阵计算形状不匹配，在反复修改后才调整好，所以一定要注意矩阵计算的参数（尤其是使用python没有明显的变量类型）

7. 参考文献

[1] 特征值和特征向量：https://zhuanlan.zhihu.com/p/95836870

[2] Python杂谈 | (15) 使用Pycharm执行带命令行参数的脚本：https://blog.csdn.net/sdu_hao/article/details/104326116

[2] 如何计算方阵的特征值和特征向量np.linalg.eig()：https://blog.csdn.net/qq_38048756/article/details/112543075