专业：计算机科学与技术
姓名：卢佳盈
学号：3180103570
日期：2020/12/29

浙江大学实验报告

课程名称：计算机视觉 指导老师：宋明黎 成绩：

实验名称：EigenFace 人脸识别算法

一、实验目的和要求

自己写代码实现 Eigenface 人脸识别的训练与识别过程：

1. 假设每张人脸图像只有一张人脸，且两只眼睛位置已知（即可人工标注给出）。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，文本文件中用一行、以空格分隔的 4 个数字表示，分别对应于两只眼睛中心在图像中的位置；
2. 实现两个程序过程（两个执行文件），分别对应训练与识别；
3. 自己构建一个人脸库（至少 40 人，包括自己），课程主页提供一个人脸库可选用；
4. 不能直接调用 OpenCV 里面与 Eigenface 相关的一些函数，特征值与特征向量求解函数可以调用；
5. 训练程序格式大致为："mytrain.exe <*能量百分比*> <*model 文件名*> <*其他参数*>..."，用能量百分比决定取多少个特征脸，将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；
6. 识别程序格式大致为："mytest.exe <*人脸图像文件名*> <*model 文件名*> <*其他参数*>..."，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。

二、实验内容和原理

2.1 开发环境

- Windows X64
- Visual Studio 2017
- opencv-3.4.0

2.2 运行方式

【训练】mytrain.exe <*能量百分比*> <*model 文件名*> <*数据库路径*>

如 `mytrain.exe 0.95 train_model ORLdatabase`

【测试】mytest.exe <*人脸图像文件名*> <*model 文件名*> <*数据库路径*>

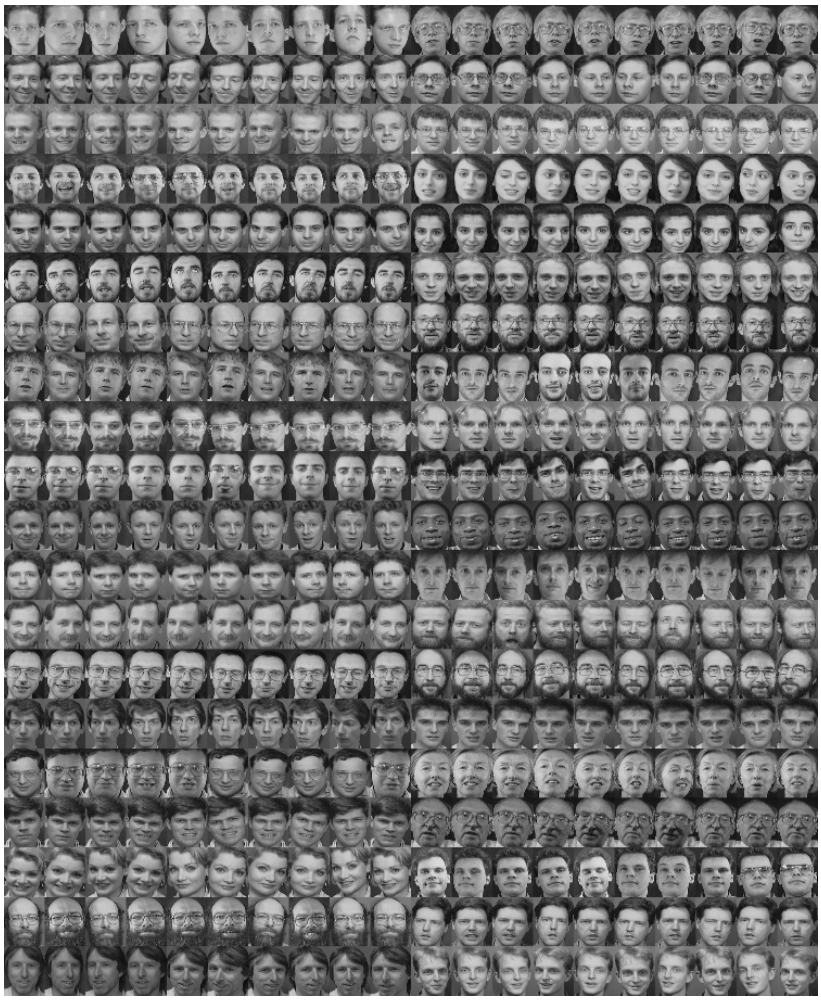
如 `mytest.exe input train_model ORLdatabase`

2.3 原理介绍

【ORL 人脸图像库】

ORL 人脸数据集共包含 40 个不同人的 400 张图像，是在 1992 年 4 月至 1994 年 4 月期间由英国剑桥的 Olivetti 研究实验室创建。

此数据集下包含 40 个目录，每个目录下有 10 张图像，每个目录表示一个不同的人。所有的图像是以 PGM 格式存储，灰度图，图像大小宽度为 92，高度为 112。对每一个目录下的图像，这些图像是在不同的时间、不同的光照、不同的面部表情(睁眼/闭眼，微笑/不微笑)和面部细节(戴眼镜/不戴眼镜)环境下采集的。所有的图像是在较暗的均匀背景下拍摄的，拍摄的是正脸(有些带有略微的侧偏)。



【特征脸方法 Eigenface】

- 训练流程
    1. 将训练集的每一个人脸图像都拉长一列，将他们组合在一起形成一个大矩阵 M。
    2. 将所有的 K 个人脸在对应维度上加起来，然后求个平均，得到了一张“平均脸”AverageFace。
    3. 将 N 个图像都减去平均脸图像，得到差值图像的数据矩阵 M。
    4. 计算协方差矩阵，再对其进行特征值分解，得到想要的特征向量(特征脸)了。
    5. 将步骤 4 中得到的 K 个特征向量中的前 P（P=100）个最大的特征值对应的向量取出，作为 PCA 子空间的一组基。
- 识别过程 训练过程保存协方差矩阵的前 100 个特征向量，然后将测试图片读入预处理后，也在这个降维后的 P 维 PCA 子空间中找到对应的坐标。这里用这个坐标与训练

过程得到的 K 个坐标分别求欧氏距离，即可比对出与测试图片欧氏距离最小的一张人脸，此时完成识别过程。

2.4 实现过程（train）

1. 读取图片

调用方式：第一个参数为图像库路径，第二个参数为训练集与测试集比例

```
trainer t("ORLdatabase", 70);
```

读取方式：以灰度图形式读取图片，然后对每张图片都调整大小为 30*30，方便特征向量的提取

```
for (int i = 1; i <= num_people; ++i)
{
    for (int j = 1; j <= (int)(img_per_person * trainPercent * 0.01); ++j)
    {
        fileName = 's' + std::to_string(i)+"/" +std::to_string(j) + ".bmp";
        trainFiles.push_back(fileName);
        cv::Mat gray, imgResized;
        cv::cvtColor(cv::imread(datasetDirent + fileName), gray, cv::COLOR_BGR2GRAY
);
        cv::resize(gray, imgResized, cv::Size(width_resize, height_resize));
        dataset.push_back(imgResized);
    }
}
```

2. 求样本平均脸

```
std::vector<cv::Mat> img_samples;
cv::Mat img_cat;
for (int i = 0; i < 10; ++i)
{
    cv::Mat tmp(cv::Size(width_resize, height_resize), CV_64F);
    cv::Mat tmp_int(cv::Size(width_resize, height_resize), CV_8UC1);
    for (int j = 0; j < width_resize * height_resize; ++j)
    {
        tmp.at<double>(j / width_resize, j % width_resize) = eigenVector.at<double>
(i, j);
    }
    cv::normalize(tmp, tmp, 255, 0, cv::NORM_MINMAX);
    tmp.convertTo(tmp_int, CV_8UC1);
    img_samples.push_back(tmp_int);
}
```

3. 计算协方差矩阵

```
cv::eigen(covar, eigenValue, eigenVector);
A_T = eigenVector.colRange(0, eigenVector.cols * energyPercent * 0.01);
```

2.5 实现过程（test）

```
cv::Mat img_read, img;
img_read = cv::imread(fileName);
```

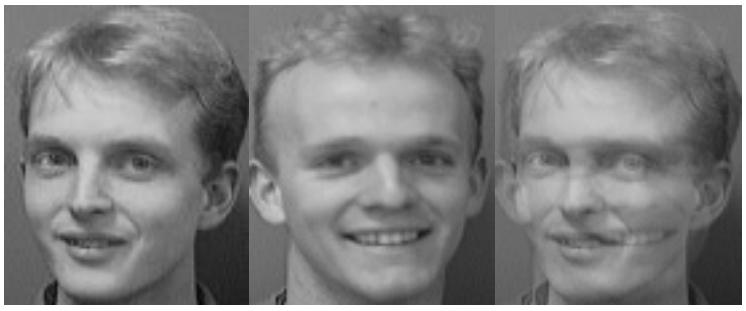
```
cv::cvtColor(img_read, img, cv::COLOR_BGR2GRAY);
cv::resize(img, img, cv::Size(30, 30));
img.reshape(0, 1).convertTo(img, CV_64F);
cv::Mat coordinate = img * A_T;

double min = -1;
int index = 0;
for (int i = 0; i < dataset.size(); ++i)
{
    cv::Mat diff = coordinate - dataset[i];
    double mo = cv::norm(diff);
    if (mo < min || min < 0)
    {
        min = mo;
        index = i;
    }
}
```

四、实验结果

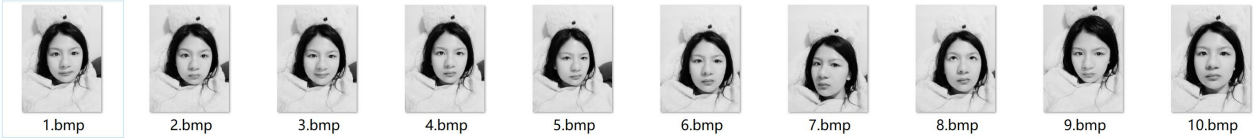
4.1 样本平均脸

4.2 测试样例

原图　　识别结果　　原图叠加



加入自己的照片后识别：



原图　　识别结果　　原图叠加