计算机视觉实验三

EigenFace人脸识别算法

钱思忆

一、 作业已实现的功能简述及运行简要说明

最终实现运用命令行进行训练和测试的效果。

训练过程：

```
E:\OpenCVPRO2\x64\Release>train.exe 0.95 eigenValue.model mean.model
```

识别过程：

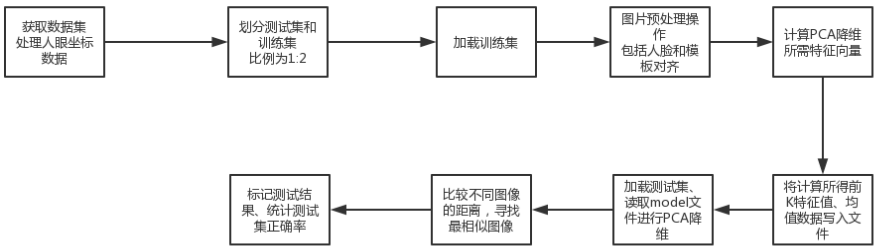
```
E:\OpenCVPRO\x64\Release>test.exe eigenValue.model mean.model
```

作业可以通过命令行来进行训练和测试。实现EigenFace人脸识别算法。本作业实现了对246张人脸的训练和对123张人脸的测试。其中测试集共41人，最后一人是自己，测试集共41人，最后一人是自己。

二、 作业的开发与运行环境

VS 2017+OpenCV 4.0+ Win X64

三、 系统或算法的基本思路、原理、及流程或步骤等



四、 具体实现方法

4.1 图片预处理、加载图像

图片预处理包括简单的resize、直方图均衡化操作。之后就是人脸对齐操作，主要通过两眼之间的角度和位置来校正人脸使得和模板对齐。

```
//preprocess.h
#pragma once
#include <opencv2/opencv.hpp>
#include <opencv2/opencv.hpp>
#include <iostream>
#include <string>
#include <vector>
#include<fstream>
#include<sstream>
#define TRAIN_IMG_PATH "facedatabase/trainimg/"
#define TEST_IMG_PATH "facedatabase/testimg/"
#define TRAIN_IMG_LIST "facedatabase/trainlist.txt"
#define TEST_IMG_LIST "facedatabase/testlist.txt"
#define TRAIN_IMG_EYE_POSE "facedatabase/traindata.txt"
#define TEST_IMG_EYE_POSE "facedatabase/testdata.txt"
using namespace std;
using namespace cv;



class FaceAlignment //人脸图像预处理操作
{
public:
    Mat origin_pic;
    Mat gray_pic;
    Mat transformed_pic;
    double x1, y1, x2, y2;
    Mat trans_mat;
    Mat equalized_mat;
    Mat vect;
    int label;

    void load(string imgpath)
    {
        gray_pic = imread(imgpath, IMREAD_GRAYSCALE);
        align();
        equalizeHist(gray_pic, gray_pic); //进行直方图均衡化，使程序对光照不敏感
        resize(gray_pic, gray_pic, Size(240, 320));//进行图片统一大小操作,这里resize
是因为原图计算量太大。
        vect = gray_pic.reshape(0, 1).t(); //将图片化为向量
    }

    void align() //人脸对齐函数，主要通过计算两眼位置和倾斜度来校正人脸位置
    {
        Point center((x1 + x2) / 2, (y1 + y2) / 2);
        double angle = atan((double)(y2 - y1) / (double)(x2 - x1)) * 180.0 /
CV_PI;
        trans_mat = getRotationMatrix2D(center, angle, 1.0); //使两眼处于水平状态
```

```
        trans_mat.at<double>(0, 2) += 285 - center.x; //平移图像使每张图像眼睛位置对齐
        trans_mat.at<double>(1, 2) += 349 - center.y;
        warpAffine(gray_pic, transformed_pic, trans_mat, gray_pic.size()); //进行几何变换操作


    }

};

class FaceLib //构建人脸识别库
{
public:
    int num_of_faces; //脸的总数
    int num_of_persons; //总共有几个对象
    int facepernum;//每人几张图像
    vector<FaceAlignment*> faces;
    vector<Mat_<double>> _samples;
    Mat_<double> samples;
    Mat_<double> graypic;
    vector<vector<double>>eye_pos;
    vector<string>filelist;
    string data_path;
    double x1, y1, x2, y2;
    void init(string eye_pose_path, int faces, int persons, int pernum)
    {

        num_of_faces = faces;
        num_of_persons = persons;
        facepernum = pernum;
        load_eye_pos(eye_pose_path, num_of_faces);
    }
    void load_eye_pos(string eye_pose_path, int num) //获取每张图像的眼睛位置数据，存储格式为左眼x,左眼y，右眼x，右眼y
    {
        ifstream in(eye_pose_path);
        for (int i = 0; i <num; i++)
        {
            vector<double>linedata;
            in >> x1 >> y1 >> x2 >> y2;
            linedata.push_back(x1);
            linedata.push_back(y1);
            linedata.push_back(x2);
            linedata.push_back(y2);
            eye_pos.push_back(linedata);


        }


    }
    void loadimg(string imglistpath, string rootPath) //加载图片
    {
        ifstream f(imglistpath); //先加载图片文件列表
        string filename;
        for (int i = 0; i < num_of_persons; i++)
        {
```

```
            for (int j = 0; j < facepernum; ++j)
            {
                f >> filename;
                filename = rootPath + filename + ".jpg"; //根据列表读取每张图片
                filelist.push_back(filename);
                FaceAlignment* face = new FaceAlignment();
                face->x1 = eye_pos[i*facepernum + j][0];
                face->y1 = eye_pos[i*facepernum + j][1];
                face->x2 = eye_pos[i*facepernum + j][2];
                face->y2 = eye_pos[i*facepernum + j][3];
                face->load(filename);
                face->label = i;
                faces.push_back(face);
                _samples.push_back(face->vect);

            }
        }
        hconcat(_samples, samples);
    }
};
```

4.2 人脸训练过程

最主要的过程是将图像转化为一个列向量，并进行PCA降维。先利用***calcCovarMatrix***函数计算协方差矩阵的特征值，并根据能量值选择保存的特征向量的个数。在计算协方差矩阵的时候加入CV_NORMAL是为了同时计算均值和协方差。

```
//train.cpp
#include "preprocess.h"
Mat Vec2Img(Mat vect, int width, int height); //用于将向量还原为图片


int main(int argc,char**argv)
{

    double energy = atof(argv[1]);
    string model_name = argv[2];
    string model_name1 = argv[3];


    FaceLib*facelib=new FaceLib();
    facelib->init(string(TRAIN_IMG_EYE_POSE), 240, 40, 6);//加载训练集
    facelib->loadimg(string(TRAIN_IMG_LIST),string(TRAIN_IMG_PATH));
    Mat samples, cov_mat, mean_mat,eigenValues,eigenVector;
    facelib->samples.copyTo(samples);
    FileStorage model(model_name, FileStorage::WRITE); //写入前K的特征向量
    FileStorage model1(model_name1, FileStorage::WRITE);//写入人脸均值数据
    calcCovarMatrix(samples, cov_mat, mean_mat, COVAR_COLS | COVAR_NORMAL); //计
算协方差矩阵
    cov_mat = cov_mat / (samples.rows - 1);
    eigen(cov_mat, eigenValues, eigenVector);
    vector<Mat> Top10; //保存前10的人脸特征值数据
    for (int i = 0; i < 10; i++)
    {
        Top10.push_back(Vec2Img(eigenVector.row(i), 240, 320));
    }
    Mat result;
```

```
    hconcat(Top10, result);//将10张人脸连在一起

    result.convertTo(result, CV_8U, 255); //imshow不支持CV_64FC1做此转换

    imshow("Top10EigenFace", result);
    imwrite("Top10EigenFace.png", result); //将前10图像写入文件

    double totalValue = sum(eigenValues)[0];
    double upEnergy = totalValue * energy;
    double energySum = 0;
    int k = 0;
    for (k = 0; k < eigenValues.rows; k++) //计算需要保存的特征值数目
    {
        energySum += eigenValues.at<double>(k, 0);
        if (energySum >= upEnergy) break;
    }

    eigenVector = eigenVector.rowRange(0, k); //保存前K的特征向量
    model << "eigenvector" << eigenVector;
    model1 << "mean_mat" << mean_mat;
    model.release();
    model1.release();


    return 0;
}

Mat Vec2Img(Mat vect, int width, int height)
{

    Mat result(Size(width, height), CV_64FC1);
    int i;
    for (i = 0; i < height; ++i)
    {
        vect.colRange(i*width, (i + 1)*width).convertTo(result.row(i),
CV_64FC1);
    }
    normalize(result, result, 255, 0.0, NORM_MINMAX); //做归一化处理
    result.convertTo(result, CV_8UC1);
    return result;

}
```

4.3 人脸识别过程

人脸识别过程主要对应于PCA降维的过程，对应的公式如下：

$$z_i = w^T \cdot x_i$$

其中w矩阵为降维矩阵（即前k个特征值（列向量）构成的矩阵），x为高维空间数据，z为投影到低维空间的数据。对训练集合测试集图像都进行到低维空间的投影操作，寻找在低维空间中句离最近的两张图片（采用L2距离）来判断测试图片属于哪个人。在进行投影操作前图片需减去均值。注：在实验中特征向量存储的方式为行向量。

```
//test.cpp
#include"preprocess.h"
Mat Vec2Img(Mat vect, int width, int height)
{

    Mat result(Size(width, height), CV_64FC1);
    int i;
    for (i = 0; i < height; ++i)
    {
        vect.colRange(i*width, (i + 1)*width).convertTo(result.row(i),
CV_64FC1);
    }
    normalize(result, result, 255, 0.0, NORM_MINMAX);
    result.convertTo(result,CV_8UC1);
    return result;

}

int main(int argc,char**argv)
{

    FaceLib*facelib=new FaceLib(); //加载测试集
    facelib->init(TEST_IMG_EYE_POSE, 123, 41, 3);
    facelib->loadimg(string(TEST_IMG_LIST),string(TEST_IMG_PATH));
    ofstream f("record111.txt"); //用来记录每张图片的预测值
    string model_name = argv[2];
    string model_name1 = argv[3];

    Mat e_vector_mat;
    Mat mean_mat;
    FileStorage model(model_name , FileStorage::READ); //读取提取的主要特征值
    FileStorage model1(model_name1, FileStorage::READ); //读取均值图像
    model["eigenvector"] >> e_vector_mat;
    model1["mean_mat"] >> mean_mat;
    Mat distance;
    Mat sample;
    FaceAlignment face;
    int idx = 0;

    FaceLib*src = new FaceLib(); //加载训练集
    src->init(TRAIN_IMG_EYE_POSE, 246, 41, 6);
    src->loadimg(string(TRAIN_IMG_LIST), string(TRAIN_IMG_PATH));
    int correctPre = 0;
    f << "person    " << "predicted" << endl;

    for (idx = 0; idx <123; idx ++)
    {
        Mat curValue = facelib->samples(Rect(idx, 0, 1, src->samples.rows));
        Mat curV1 = e_vector_mat * (curValue - mean_mat); //将测试集当前图像减去人脸
均值
        Mat libValue = src->samples(Rect(0, 0, 1, src->samples.rows));
        Mat curV2 = e_vector_mat * (libValue - mean_mat); //将训练集当前图像减去人脸
均值，进行降维
        string fpath = facelib->filelist[idx];
        Mat testMat = imread(fpath);
        double min_d = norm(curV1, curV2, NORM_L2); //计算两张图像间的距离，采用L2范
式
        int min_idx = 0;
```

```
        int min_pic = 0;
        double dis;
        for (int pic = 0; pic < 246; pic++)
        {
            libValue = src->samples(Rect(pic, 0, 1, src->samples.rows));
            curV2 = e_vector_mat * (libValue - mean_mat);
            dis = norm(curV1, curV2, NORM_L2);
            if (dis < min_d) //更新最相似图像
            {
                min_d = dis;
                min_idx = pic / 6; //获取该图像属于哪个人
                min_pic = pic;  //更新最相似图片下标
            }

        }

        Mat most_smiliar=  src->samples(Rect(min_pic, 0, 1, src-
>samples.rows)).t();
        Mat re = Vec2Img(most_smiliar,240,320);
        imshow("Most similar:", re);//展现最相似的原图

        Mat re1 = Vec2Img(curValue.t(), 240, 320);
        imshow("current:", re1);

        Mat dst;
        addWeighted(re, 0.5, re1, 0.5, 0, dst);
        imshow("MergePic:", dst); //展现融合后的图像

        string text = "Person" + to_string(min_idx);
        putText(testMat, text, Point(10, 580), FONT_HERSHEY_COMPLEX, 0.5,
(255,0,255)); //原图片（彩色图片）打上识别的人物结果标记
        string text2="Most similar:No." + to_string(min_pic);
        putText(testMat, text2, Point(10,610 ), FONT_HERSHEY_COMPLEX, 0.5, (255,
0, 255));//图片打上每张图片最相似的图片序号
        imwrite(fpath,testMat);
        int person = idx / 3;
        if (person == min_idx) //真实结果和识别结果一致
            correctPre += 1;
        f << person <<"    "<<min_idx<< endl;
        waitKey(0);
    }

    double ratio = 1.0*correctPre / 123; //计算正确率
    f<<"ratio:   "<<ratio<<endl;


}
```

五、实验结果与分析

5.1 数据集说明

训练集：240+6张图片，共41人，前40人图像来自于MUCT Data Sets人脸数据库，最后6张为自己图像。

测试集：120+3张图片，对应于训练集。

MUCT数据集：

MUCT 数据集是一个人脸数据库，主要用于身份鉴定，该数据集由 3755 个人脸图像组成，每人对应15张图像。每个人脸图像有 76 个点的地标（landmark），其中31号为左眼中心标记，36号为人脸右眼标记。该图像库在种族、关照、年龄等方面表现出更大的多样性。

5.2 测试结果说明：

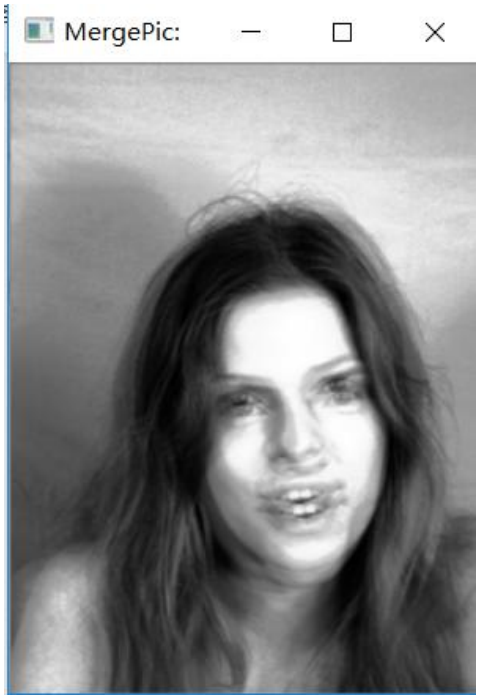
每张图片在测试后均在左下角打上测试结果（属于哪个人）和与之对应最接近的图像标号。

测试集随机选择的一张图片效果显示：

原图：

最相似图片：

叠加显示的照片：



分析：根据直观观察，这一组图片两张照片间也确实非常相似，从叠加结果可以发现两张照片除了人脸没有对齐，其余部分具有高度相似性。在实验进行对齐操作的预处理后，PCA降维能较好地在更低的维度保留主要信息。

测试集预测正确率：

record - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
29      29
29      29
30      30
30      30
30      30
31      3
31      3
31      3
32      32
32      22
32      22
33      33
33      22
33      29
34      34
34      34
34      34
35      7
35      35
35      35
36      36
36      36
36      36
37      37
37      37
37      37
38      38
38      38
38      23
39      37
39      39
39      39
40      40
40      40
40      40
ratio:    0.731707
```

结论：实验测试准确率较高，PCA降维应用于人脸识别具有一定准确性。

前10的人脸特征向量：