计算机视觉

实验三　EigenFace

黄昭阳 3130000696- 1/2/16

个人信息

3130000696
黄昭阳

开发软件说明

提供的可执行文件在Mac下编译运行，共有mytest和mytrain两个文件，其中：

1. mytrain使用方法

```
“./mytrain 能量百分比 训练数据放置文件夹 训练集”
eg.
 ./mytrain 0.95 Model
```

程序使用训练集指定目录（默认为当前目录下的trainSet目录）下的图片进行学习，学习结果数据放在指定的训练数据放置文件中（上方为、Model文件）。

trainSet目录中共54个目录，每个目录对应一个人，程序在每个目录下取七张脸进行训练，作者的脸放在第54个目录中。每个目录中都不止7张图片，剩下的图片可以作为测试图片使用。

2. mytest使用方法

```
“./mytest 指定识别图片 训练数据来源目录 训练集”
./mytest trainSet/51/11.jpg Model
```

由于最终需要输出最相似的图片，而按照标准过程每张脸向量都需要做归一化，因此显示出来的脸效果不好，这里程序使用训练集指定目录（默认为当前目录下的trainSet目录）来获取对应图片显示到屏幕。

由于提供的可执行文件在Mac下编译的，提供源代码与makefile文件，可以自行编译。

3. 所有人的图片都只取了前七张进行训练，多余的图片可以做测试用，作者的图片在54文件夹

算法具体步骤及实现要点

一、算法步骤：

1. mytrain实现步骤

```
    string ModelFolder = “Model/";
    string inputPath = "trainSet/";
    if(argc<3) return -1;
    EnergyPropotion = atof(argv[1]);  //能量比例
    ModelFolder = string(argv[2]);
    ModelFolder = ModelFolder+'/';    //训练数据结果放置目录
    if(argc>=4)                       //训练数据来源目录
    {
      inputPath = string(argv[3]);
      inputPath = inputPath+'/';
    }
//接受输入数据，分别如上所示
    namedWindow("images");
    InputFaces();  //从inputPath中读入所有脸的数据
    GetMeanFace(); //获取平均脸
    GetEigenBase(); //获取所有的特征脸
    ShowEignFace(); //按照实验要求展示前十章特征脸，并将其输入到ModelFolder目录中
    CalcNewCoordinate();  //使用获得的特征脸计算新坐标
    ExportData();         //导出数据到ModelFolder目录中
    destroyWindow("images");
```

2. mytest实现步骤

```
    if(argc<3) return -1;
    inputImgName = string(argv[1]);  //需要匹配的脸的文件名
    ModelFolder = string(argv[2]);   //训练数据结果目录
    if(argc == 4) trainSetPath = string(argv[3]);  //训练数据来源目录
    namedWindow("Result");
    namedWindow("Input");
    GetInput();      //从ModelFolder中将mytrain得到的结果读入
    MatchFace();     //使用数据匹配脸
```

二、实现要点：

1. mytrain实现要点：

mytrain的关键之处在于获取特征脸和计算新坐标这两部，分别对应上面InputFaces、GetEigenBase()、CalcNewCoordinate()这三个函数，其他函数比较简单，这里不一一展开了。

```
void InputFaces()
{
    char number[10],name[20];
    char eye[20];
    FILE* fp;
    number[0] = number[1] = number[2] = '\0';
    vector<Mat> bgr;
    int lx,ly,rx,ry,x,y;
    for(int i=0;i<IMAGENUM;i++)  //根据宏读入图片，并转灰度图和直方图均匀化
    {
        sprintf(number, "%d",i+1);
        for(int j=0;j<SAMPLENUM;j++)
        {
            sprintf(name, "/%d.jpg",j+1);
            dataset[i*SAMPLENUM+j] = imread(inputPath+number+name);
            split(dataset[i*SAMPLENUM+j],bgr);
            equalizeHist(bgr[0], dataset[i*SAMPLENUM+j]);
       }
    }
    width = dataset[0].cols;
    height = dataset[0].rows;
    for(int i=0;i<IMAGENUM;i++)
        for(int j=0;j<SAMPLENUM;j++) //将行向量指针全部存到DataSet中，加速程序
        {
            resize(dataset[i*SAMPLENUM+j], dataset[i*SAMPLENUM+j],
Size(width,height));
            DataSet[i*SAMPLENUM+j] = (uchar**)malloc(dataset[i*SAMPLENUM
+j].rows*sizeof(double*));
            for(int k=0;k<dataset[i*SAMPLENUM+j].rows;k++)
                DataSet[i*SAMPLENUM+j][k] = dataset[i*SAMPLENUM
+j].row(k).ptr<uchar>();
        }
    originalcoord = Mat(IMAGENUM*SAMPLENUM,width*height,CV_64F);
    double** OriginalCoord =
(double**)malloc(width*height*sizeof(double*));
    for(int k=0;k<IMAGENUM*SAMPLENUM;k++)
    {
        OriginalCoord[k] = originalcoord.row(k).ptr<double>();
        for(int i=0;i<height;i++)
            for(int j=0;j<width;j++)
                OriginalCoord[k][i*width+j] = DataSet[k][i][j];
    }
    //获取旧坐标
    transpose(originalcoord, originalcoord);
    //将旧坐标转制，行向量变列向量，之后可以直接相乘
    ToUnit(originalcoord);
    //将旧坐标按照列向量归一化
}
```

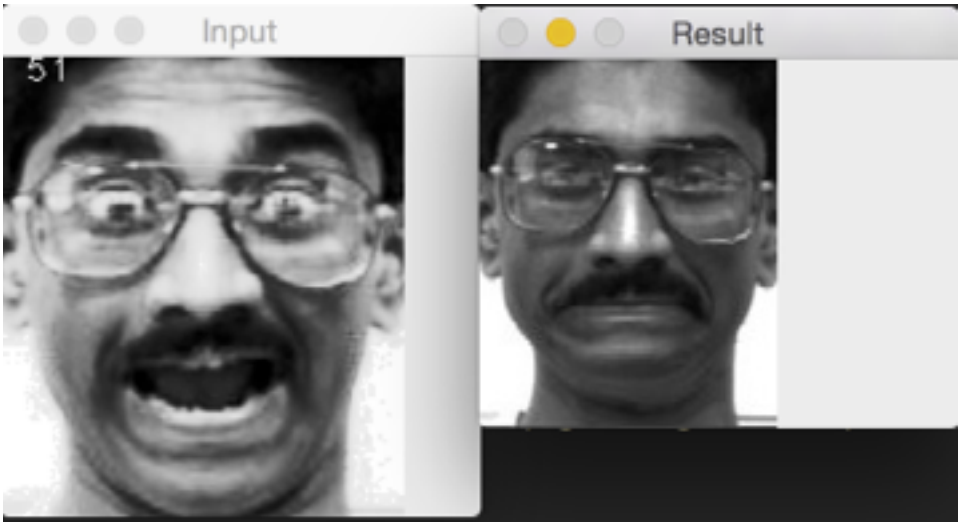
```
void GetEigenBase()
{
    double sum = 0;
/*
我们的目标是求协方差矩阵M的特征值和特征向量，但是M的维度过大，按照概率论公式计算
协方差矩阵元素m[i][j] = Σ(xi(k)-μxi)(xj(k)-μxj)，数据xi和数据xj都取第k个样本
现在构建矩阵A，A[i][j] = xi[j] - μi，则 m = A^T*A，求m的特征值即A奇异值的平方，
对应的特征向量相同。
在EigenFace算法中，每一张脸都作为一个行向量存在B中，平均脸也作为一个行向量μ，则
B中每一个行向量都减去μ向量得到A，下面的distanceset即矩阵A
*/
    distanceset = Mat(IMAGENUM*SAMPLENUM, height*width, CV_64F);
    for(int k=0;k<IMAGENUM*SAMPLENUM;k++)
    {
        DistanceSet[k] = distanceset.row(k).ptr<double>();
//将这个行向量指针取出来，避免重复使用函数取指针，可以加快程序速度，一张脸即一个行向量

        for(int i=0;i<height;i++)
            for(int j=0;j<width;j++)
                DistanceSet[k][i*width+j] = DataSet[k][i][j] - MeanMat[i]
[j];
/*
Dataset[k] 表示第k张脸， Dataset[k][i][j] 表示第k长脸的[i][j]处像素
MeanMet表示所有脸的平均脸，MeanMat[i][j]表示平均脸[i][j]处像素
这里求得了矩阵A
*/
    }
    Mat res,left,right;
    SVD::compute(distanceset, res, left, right);
//使用SVD分解求得A的奇异值和对应的向量
    double* p = res.col(0).ptr<double>();
    for(int i=0;i<IMAGENUM+SAMPLENUM;i++)
    {
        p[i]*=p[i];
//奇异值的平方才是协方差矩阵真正的特征值
        sum+= p[i];
    }
//对特征值求和，以按照能量比例来筛选特征向量
    double d = 0;
    for(int i=0;i<IMAGENUM+SAMPLENUM;i++)
    {
        d+=p[i];
        if(d/sum>=EnergyPropotion)
        {
            length = i+1;
            break;
        }
    }
    EigenBase = right.rowRange(0, length);//选前几个满足能量比例要求的特征向量
}
```

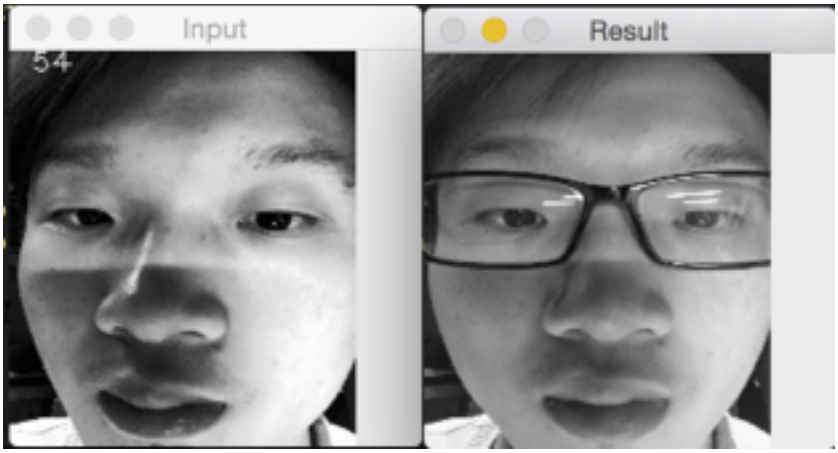
```
void CalcNewCoordinate()
{
    newcoordinate = EigenBase*originalcoord;
    transpose(newcoordinate, newcoordinate);
}
//计算新坐标其实很容易，使用特征向量作为线性转换矩阵，直接转换即可获得
```

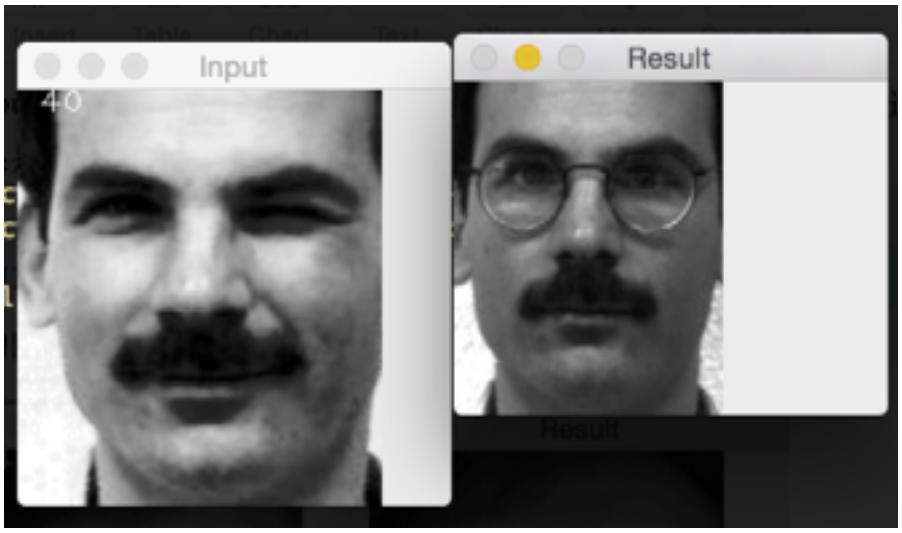
2. mytest实现要点：

mytest很容易，只需要读入Model中的数据，读入图片数据，然后将原来的图片坐标转换到新坐标，按照欧氏距离计算相似度，然后输出相似度最高的图片即可。mytest中最重要的函数式MatchFace()，下面进行介绍

```
void MatchFace()
{
    Mat inputImg;
    Mat resultImg;
    vector<Mat> bgr;
    uchar** InputData = (uchar**)malloc(height*sizeof(double*));
    char number[20],name[20];
    inputImg = imread(inputImgName);
    resize(inputImg, inputImg, Size(width,height));
    split(inputImg, bgr);
    equalizeHist(bgr[0], inputImg);
//读入一张图片，将图片转换的对应的长宽，然后转灰度图并直方图均衡化
    for(int i=0;i<height;i++)
      InputData[i] = inputImg.row(i).ptr<uchar>();
//同样，取出所有行向量的指针加速程序
    Mat originalvec = Mat(1,width*height,CV_64F);
    double* OriginalVec = originalvec.row(0).ptr<double>();
    for(int i=0;i<height;i++)
      for(int j=0;j<width;j++)
            OriginalVec[i*width+j] = InputData[i][j];
//按照图片升恒旧坐标的向量，放到originalvec中
    transpose(originalvec, originalvec);
//将行向量转为列向量
    ToUnit(originalvec);
//归一化
    Mat newvec = EigenBase*originalvec;
//使用EigenBase线性转换，将旧坐标转为新坐标
    transpose(newvec, newvec);
//将列向量转为行向量，因为OpenCV中行向量的存储地址是连续的，而列向量则不一定
    Mat distanceset = Mat(1,PersonNumber*SampleNumber,CV_64F);
//distanceset用来存储输入的图片在新空间中与其他图片的距离
    double* p = distanceset.row(0).ptr<double>();
    double* ptr;
    double* vecp = newvec.row(0).ptr<double>();
//分别取出行向量的指针，用来加速程序
```


Input
51
Result


Input
54
Result


Input
Result


Input
Result

```
    for(int i=0;i<distanceset.cols;i++)
    {
      ptr = NewCoord.row(i).ptr<double>();
      p[i] = 0;
      for(int j=0; j<NewCoord.cols;j++)
            p[i]+= (ptr[j]-vecp[j])*(ptr[j]-vecp[j]);
      p[i] = sqrt(p[i]);
    }
//计算欧式距离
    double min=100000000;
    int chosen=-1;
//取出距离最小的图片作为最相似的图片
    for(int i=0;i<PersonNumber*SampleNumber;i++)
      if(min>p[i])
      {
            min = p[i];
            chosen = i;
      }
    sprintf(number, "/%d",chosen/SampleNumber+1);
    sprintf(name, "/%d.jpg",chosen%SampleNumber+1);
    char text[10];
    sprintf(text,"%d",chosen/SampleNumber+1);
    putText(inputImg, string(text), Point(10,10), 1, 1, Scalar(255,0,0));
//将计算得到的结果（第几号人）加到输入图片上显示
    imshow("Input", inputImg);
    resultImg = imread(trainSetPath+number+name);
    imshow("Result", resultImg);
    waitKey();
}
```

实验结果展示及分析

使用命令运行程序获取实验结果

```
./mytest trainSet/51/11.jpg Model
```

分析：

在mytrain中指去了前7张图片作为训练，因此第八张以后的图片可以作为测试图片使用，左边Input窗口显示输入图片，51为匹配结果，最相似的图片显示在右边Result窗口可以看到结果是正确地。下面展示多组结果。

```
./mytest trainSet/54/11.jpg Model
```

这个是作者自己的图片

```
./mytest trainSet/40/11.jpg Model
```

以上都是匹配正确地结果，下面这一组是匹配错误的结果，但是可以看到，这两个人确实是非常的像，而且做出了相似的表情，所以匹配错误情有可原。

```
./mytest trainSet/50/9.jpg Model
```

编程体会

1. 学会了OpenCV中的许多函数的使用方法，比如使用xml存储数据的cvLoad和cvSave，这样读写文件比自己来弄要方便很多。
2. 第一次接触PCA算法，觉得这个算法简直诠释了什么叫做数学的美丽，在算法学习过程中还对矩阵的奇异值有了更深一层次的了解，并且灵活运用了概率论和矩阵乘法的特性，大量缩短了求特征值的时间
3. 这是第一次接触机器学习，感觉挺有意思的，自己一步步将算法实现出来，在最终测试的时候很有成就感，在眼睛位置对好的情况下识别率还是比较高的
4. 这个算法有个很大的问题就是眼睛如果不对齐（就是脸不对齐）就会出现很大的问题，那么脸侧过一定角度，退后一定角度都会出现一定问题，我想是不是可以使用SIFT特征

检测出脸的scale程度和rotate程度，再构建一个透视转换矩阵H将脸拉回到正常位置，这样加大匹配的成功率？虽然这样好像有点大材小用的感觉。