ELEC 474

Machine Vision

Lab 5 Solutions

Kevin Hughes

Due Date: Wednesday, November 7th, 2012 in lab

Pre-Lab

```
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/objdetect/objdetect.hpp>
#include <opencv2/highgui/highgui.hpp>

#include <iostream>
#include <string>
#include <vector>

using namespace cv;
using namespace std;

Mat detectFace(const Mat &image, CascadeClassifier &faceDetector);
void resizeFace(Mat &face);
void addToDataSet(Mat &data, vector<string> &labels, Mat &newData, vector<string> &newLabels);

int main()
{
   // Set up face detector
   CascadeClassifier faceDetector;
   if(!faceDetector.load("lbpcascade_frontalface.xml"))
    {
        cerr << "ERROR: Could not load classifier cascade" << endl;
        return -1;
    }

    Mat image;
    Mat face;
    string name;

    Mat samples;
    vector<string> labels;

    // Brad Pitt
    image = imread("BradPitt.jpg",1);
    name = "Brad Pitt";

    face = detectFace(image,faceDetector);
    resizeFace(face);
    samples.push_back(face.clone().reshape(1,1));
    labels.push_back(name);
    imshow("face", face);
    waitKey();

    // Nicolas Cage
    image = imread("NicolasCage.jpg",1);
    name = "Nicolas Cage";

    face = detectFace(image,faceDetector);
    resizeFace(face);
    samples.push_back(face.clone().reshape(1,1));
    labels.push_back(name);
    imshow("face", face);
    waitKey();

    // Arnold Schwarzenegger
    image = imread("ArnoldSchwarzenegger.jpg",1);
    name = "Arnold Schwarzenegger";

    face = detectFace(image,faceDetector);
    resizeFace(face);
    samples.push_back(face.clone().reshape(1,1));
    labels.push_back(name);
    imshow("face", face);
    waitKey();

    // Clive Owen
    image = imread("CliveOwen.jpg",1);
    name = "Clive Owen";
```

```
    face = detectFace(image,faceDetector);
    resizeFace(face);
    samples.push_back(face.clone().reshape(1,1));
    labels.push_back(name);
    imshow("face", face);
    waitKey();

    // Matt Damon
    image = imread("MattDamon.jpg",1);
    name = "Matt Damon";

    face = detectFace(image,faceDetector);
    resizeFace(face);
    samples.push_back(face.clone().reshape(1,1));
    labels.push_back(name);
    imshow("face", face);
    waitKey();

    //cout << samples.rows << " " << samples.cols << endl;
    //Mat newSamples = samples.clone();
    //vector<string> newLabels = labels;
    //addToDataSet(samples,labels,newSamples,newLabels);
    //cout << samples.rows << " " << samples.cols << endl;

    FileStorage fs;

    // Save Data to xml

    fs.open("5623851.xml", FileStorage::WRITE);

    fs << "samples" << samples;
    fs << "labels" << labels;

    fs.release();

    // Load Data from xml
    fs.open("5623851.xml", FileStorage::READ);

    fs["samples"] >> samples;
    fs["labels"] >> labels;

    fs.release();

    cout << samples.rows << " " << samples.cols << endl;
    cout << labels.size() << endl;
    return 0;
}

Mat detectFace(const Mat &image, CascadeClassifier &faceDetector)
{
  vector<Rect> faces;
  faceDetector.detectMultiScale(image, faces, 1.1, 2, 0|CV_HAAR_SCALE_IMAGE, Size(30, 30));

  if(faces.size() == 0)
  {
    cerr << "ERROR: No Faces found" << endl;
    return Mat();
  }

  if(faces.size() > 1)
  {
    cerr << "ERROR: Multiple Faces Found" << endl;
    return Mat();
  }

  //Mat detected = image.clone();
  //for(unsigned int i = 0; i < faces.size(); i++)
  //{
  //    rectangle(detected, faces[i].tl(), faces[i].br(), Scalar(255,0,0));
  //}
```

```
  //imshow("faces detected", detected);
  //waitKey();

  return image(faces[0]).clone();
}

void resizeFace(Mat &face)
{
    resize(face, face, Size(200,200));
}

void addToDataSet(Mat &samples, vector<string> &labels, Mat &newSamples, vector<string> &newLabels)
{
    if(samples.cols != newSamples.cols)
    {
        cerr << "ERROR: Cannot add 2 data sets of different size" << endl;
        return;
    }

    for(unsigned int i = 0; i < newSamples.rows; i++)
    {
        samples.push_back(newSamples.row(i).clone());
        labels.push_back(newLabels[i]);
    }
}
```

Lab

```
#include <opencv2/core/core.hpp>
#include <opencv2/contrib/contrib.hpp>
#include <opencv2/ml/ml.hpp>
#include <opencv2/highgui/highgui.hpp>

#include <iostream>
#include <string>
#include <vector>

using namespace cv;
using namespace std;

void addToDataSet(Mat &data, vector<string> &labels, Mat &newData, vector<string> &newLabels);
Mat norm_0_255(Mat src);
string recognizeFace(Mat input, Mat samples, vector<string> labels);

int main()
{
    Mat samples;
    vector<string> labels;

    FileStorage fs;

    fs.open("5623851.xml", FileStorage::READ);

    fs["samples"] >> samples;
    fs["labels"] >> labels;

    fs.release();

    // Add Data sets using code from the pre-lab
    //cout << "size before add: " << samples.rows << " " << samples.cols << endl;
    //Mat newSamples = samples.clone();
    //vector<string> newLabels = labels;
    //addToDataSet(samples,labels,newSamples,newLabels);
    //cout << "size after add: " << samples.rows << " " << samples.cols << endl;

    // Perform PCA
    PCA pca(samples, noArray(), CV_PCA_DATA_AS_ROW, samples.rows - 1);

    // Visualize Mean
    Mat meanFace = pca.mean;
    meanFace = norm_0_255(meanFace.reshape(3,200));
    imshow("meanFace", meanFace);
    waitKey();

    // Visualize Eigenfaces
    for(unsigned int i = 0; i < pca.eigenvectors.rows; i++)
    {
        Mat eigenface;
        eigenface = pca.eigenvectors.row(i).clone();
        eigenface = norm_0_255(eigenface.reshape(3,200));
        //applyColorMap(eigenface, eigenface, COLORMAP_JET);

        imshow(format("eigenface_%d", i), eigenface);
        waitKey();
    }

    // Project all samples into the Eigenspace
    pca.project(samples,samples);
    //cout << samples.rows << " " << samples.cols << endl;

    // ID faces
    for(int i = 0; i < 5; i++)
    {
        string name = recognizeFace(samples.row(i).clone(), samples, labels);
        cout << "name = " << name << endl;
    }
```

```
    return 0;
}

void addToDataSet(Mat &samples, vector<string> &labels, Mat &newSamples, vector<string> &newLabels)
{
    if(samples.cols != newSamples.cols)
    {
        cerr << "ERROR: Cannot add 2 data sets of different size" << endl;
        return;
    }

    for(unsigned int i = 0; i < newSamples.rows; i++)
    {
        samples.push_back(newSamples.row(i).clone());
        labels.push_back(newLabels[i]);
    }
}

Mat norm_0_255(Mat src)
{
    // Create and return normalized image:
    Mat dst;
    switch(src.channels())
    {
    case 1:
        cv::normalize(src, dst, 0, 255, NORM_MINMAX, CV_8UC1);
        break;
    case 3:
        cv::normalize(src, dst, 0, 255, NORM_MINMAX, CV_8UC3);
        break;
    default:
        src.copyTo(dst);
        break;
    }
    return dst;
}

string recognizeFace(Mat input, Mat samples, vector<string> labels)
{
    Mat labelIdx;
    for(unsigned int i = 0; i < labels.size(); i++)
    {
        labelIdx.push_back((int)i);
    }

    CvKNearest* knn = new CvKNearest(samples, labelIdx);

    int id = (int)(knn->find_nearest(input,1));

    return labels[id];
}
```