ELEC 474

Machine Vision

Lab 5 Pre-Lab

Preparing Data for Classification

Kevin Hughes

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

In the pre-lab for lab 5 you will prepare data (images with faces) for a facial recognition algorithm that will be implemented in the lab session. Organizing data is often a very big part of a machine vision system.

Marking Scheme

Completion	2 marks
Overall Effort	1 mark
Total	3 marks

Part 1 - Gathering Face Data

You have been provided with a function that detects faces in an image and returns the sub-image of the face. The detection algorithm being used is the Local Binary Pattern Cascade Classifier that is distributed with OpenCV. OpenCV has both the code for this classifier and pre-made training data for use (you can see the latter being loaded in the main function). You can uncomment the section in the middle of the function to see the rectangles drawn around the detected faces. The classifier can find multiple faces but for this exercise it has been wrapped to only return a single face. Take a second to review this function:

```
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();
}
```

You can read more about the Cascade Classifier here:
`http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html`

For the Appearance Based Facial Recognition (Eigenface) that you will implement in lab 5 all the face images must be the same size. The first task of the pre-lab is to fill in code for the *resizeFace* function to re-size the face image to be 200×200 pixels.

Part 2 - Preparing the Data Set

For the next part of the pre-lab you need to create and prepare a data set of 5 faces with labels. For the 5 faces in your data set you can use whoever you like: friends, family or celebrities, you should put a reasonable amount of effort into ensuring your data set is unique. 1 mark for the pre-lab will be given for overall effort in creating your data set.

Your data set needs to be in the following form: A cv::Mat called samples which has a single sample in each row. This means you need to reshape your face images into row vectors. The reshape method of cv::Mat can do this for you. Afterward you can use the push_back method with the samples Mat to build your samples matrix.

The following code snip may be useful:

```
Mat face;
Mat samples;

face = detectFace(image,faceDetector);
resizeFace(face);
samples.push_back(face.clone().reshape(1,1));
```

You also need to create a vector of strings that contains the labels of the data. Labels are usually numbers and are often binary (0 or 1) but for this application we are going to use a vector of strings that contain the name of the person whose face it is. The index of the name in this vector should match the index of row where the face is in the samples Matrix.

A quick check to make sure your data set is correct is to print out the matrix size after you are done building it. The samples matrix should have 5 rows and 40000 columns. The labels vector should have a .size() of 5.

After creating the data we are going to save it to file so that we can load it from a different program later. To do this we are going to use the cv::FileStorage class which will make this process very simple. Just add the following code to your program (just the save part for your prelab, the load code will be used in the lab) and change the file-name to be your student number.

```
FileStorage fs;
Mat samples;
vector<string> labels;

// 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();
```

cv::FileStorage can store most common data types and most OpenCV data types in several file formats. Open the xml file and have a look at how it has saved the data.

Part 3 - Combining Data Sets

The final part of the pre-lab is to create a function that will combine 2 data sets into a single data set. This function will be used in the lab as you will share your data with your colleagues to create a data set of a meaningful size.

For example this function should be able to take your data set and the data set of another student and return a single data set with all the data. This would mean the samples matrix would have 10 rows, still 40000 columns and the labels vector would have a .size() of 10 as well.