"Sea-Eye" System Design Report

Yunpeng Ge
yge6@binghamton.edu
Bin Wang
bwang43@binghamton.edu
Lu lu
llu17@binghamton.edu
Shi Tang
stang22@binghamton.edu

Keywords

Smartphone; Socket; OpenCV; Face recognition

1. Introduction

Traditional monitors are installed in the fixed positions. Even if they can rotate around, there are still some blind angles the monitors cannot cover. To solve this problem, our group proposes the "Sea-Eye" system design. "Sea-Eye" consists of two main components: an application on the smartphone and an executable program on the computer. The application on the smartphone can send the pictures or videos to the computer, and then the executable program will do the face recognition on the computer. By doing this, the pictures or videos recorded by the camera of the smartphone can be received by the computer. And the program on the computer can analysis these files to recognize which people are recorded in these files. The cameras of the smartphones become the eyes of the computer.

2. Architecture design

The Sea-Eye system architecture consists of two components: an application on the smartphone and an executable program on the computer. Sea-Eye is the bridge between the smart phone and the main computer, which can achieve smart phones' camera becoming the eyes of the main computer. The application on the smartphone will use the camera to record the surroundings first, and then send the videos or pictures to the main computer. The main computer will receive the videos or pictures in a short time and do the face recognition with the videos or pictures to find the target.

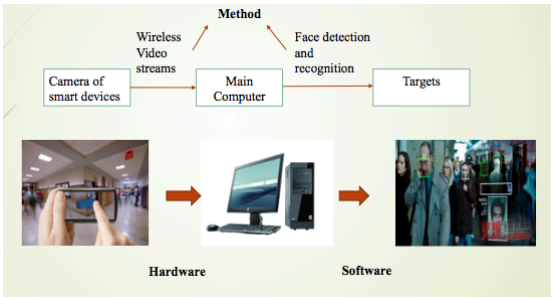

Figure 1. Sea-Eye architecture design

2.1 The smartphone application

The smartphone application is the side that sends data. To communicate with the main computer, the smartphone application needs to find out the IP address of the main computer. Since there will be many smartphones becoming the eyes of the main computer simultaneously, and the IP address of the smartphones will be easily changed, the main computer side should be the server and wait to be connected. Thus the smartphone application becomes the client side. The smartphone application needs to locate the IP address of the main computer and send the recording data. In order to send the video data, which is always a huge data, the smartphone application needs to separate the data to many segments, and send them one by one.

2.2 The program on the main computer

The executable program on the main computer is the side that receives recording data and does the face

recognition. At first, it should receive the data sent by the smartphone. As the main computer is the server, the executable program should build a socket server that wait for client requesting to connected. Once the program is connecting with the client, the program will start to receive the data stream and write this data into a file. The file is the recording video or picture. Secondly, the executable program should open the file and do the face recognition.

3. Implementation

3.1 Transfer files

3.1.1 The application on the smartphone (Client side)

Firstly, the client side needs to get the IP address of the server, then sends request to connect to the server. If the connection is built successfully, it will send the data of the file to the server. All the data of the file is stored in several arrays, and then they will be sent one by one.

3.1.2 The program on the computer (Server side)

The server should create a socket first, and then wait for the client to connect. If the server receives the request from the client, the connection will be built. Then the server receives the data from the data buffer and writes them into a file. When all the data have been written into a file, the file is received completely.

3.2 Face recognition

3.2.1 Face Recognition System Overview

Face Recognition includes two parts: face detection and face Recognition. Face detection is to check if there are faces in images or videos, if there are faces, the system will mark and track the faces. Face recognition is to tell us whose face it is, the system will compare news faces with the faces in the database, if matched, the system will display the name and cut the face image and store it to the local. If not matched, it will display "unknown".

Our Face Recognition System has several functions. The system can Register and train faces. It also can recognize faces from images and videos. Moreover, it also can call the PC camera to recognize faces in real time.

3.2.2 Tools for the Face Recognition System

What we used to do face recognition are Eclipse, OpenCV and JavaCV. OpenCV (*Open Source Computer Vision*) is a library of programming functions mainly aimed at real-time computer vision. JaveCV is the interface to OpenCV. We will use the face detection classifier ("haarcascade_frontalface_alt2") in the OpenCV and call the face recognition function in the OpenCV and JavaCV to do face recognition.

3.2.3 Approach and Procedures

The approach we used is Eigenface Approach. The information theory approach of encoding and decoding face images extracts the relevant information in a face image, encode it as efficiently as possible and compare it with database of similarly encoded faces. The encoding is done using features, which may be different or independent than the distinctly perceived features like eyes, ears, nose, lips, and hair.

Mathematically, principal component analysis approach will treat every image of the training set as a vector in a very high dimensional space. The eigenvectors of the covariance matrix of these vectors would incorporate the variation amongst the face images. Now each image in the training set would have its contribution to the eigenvectors (variations). This can be displayed as an "*eigenface*" representing its contribution in the variation between the images. In each eigenface some sort of facial variation can be seen which deviates from the original image.

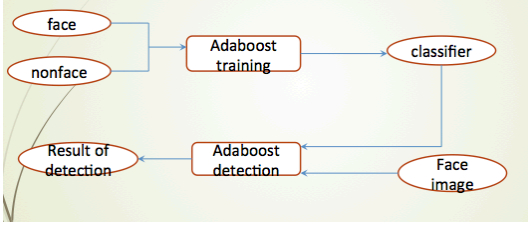

Figure 2. The procedure of face detection

The high dimensional space with all the eigenfaces is called the image space (feature space). Also, each image is actually a linear combination of the eigenfaces. The amount of overall variation, that one eigenface counts for, is actually known by the eigenvalue associated with the corresponding eigenvector. If the eigenface with small eigenvalues are neglected, then an image can be a linear combination of reduced no of these eigenfaces. For example, if there are M images in the training set, we would get M eigenfaces. Out of these, only M' eigenfaces are selected such that they are associated with the largest eigenvalues. These would span the M'-dimensional subspace 'face space' out of all the possible images (image space).

When the face image to be recognized (known or unknown), is projected on this face space, we get the weights associated with the eigenfaces that linearly approximate the face or can be used to reconstruct the face. Now these weights are compared with the weights of the known face images so that it can be recognized as a known face in used in the training set. In simpler words, the Euclidean distance between the image projection and known projections is calculated; the face image is then classified as one of the faces with minimum Euclidean distance.

The approach to face recognition involves the following initialization operations：

1.Acquire an initial set of face images (the training set).

2.Calculate the eigenfaces from the training set, keeping only the M images that correspond to the highest eigenvalues. These M images define the face space. As new faces are experienced, the eigenfaces can be updated or recalculated.

3.Calculate the corresponding distribution in M-dimensional weight space for each known individual, by projecting his or her face images onto the "face space".

These operations can also be performed from time to time whenever there is free excess computational capacity.

Having initialized the system, the following steps are then used to recognize new face images:

1.Calculate a set of weights based on the input image and the M eigenfaces by projecting the input image onto each of the eigenfaces.

2.Determine if the image is a face at all (whether known or unknown) by checking to see if the image is sufficiently close to "face space".

3.If it is a face, classify the weight pattern as either a known person or as unknown.

4.Update the eigenfaces and weight patterns.

5.If the same unknown face is seen several times, calculate its characteristic weight pattern and incorporate into the known faces.

4. Evaluation

We evaluated "Sea-Eye" system in the real world. The first advantage of Sea-eye systems is flexibility. As almost every person has a smartphone, each camera of these smartphones can be the extension and supplement of the surveillance system. When we need to track a target, the place where the fixed camera can monitor is limited. However, if some pedestrians are willing to record videos or take pictures, their smartphones will be the "moving surveillance system". Therefore, the surveillance range will be highly improved.

The second advantage of Sea-eye systems is that it can cover every angle. Since people use the cameras, the cameras can monitor any places if the users want. However, the static camera cannot do this. Although

the static cameras can monitor many place, but they still have the blind angle.

The third advantage is that our Sea-eye system can recognize the face of the target people. It will help the users recognize the target people easily.

Our system still has some disadvantages. The first disadvantage is that the system, consists of the smartphone application and the main computer program, can exchange files only in the same local area network. The convey speed is very high, because we use socket programming. But because of the socket programming, file cannot be transferred to the server in different local area network. What's more, the accuracy of the face recognition needs to be improved.