Network-Based System for Face Recognition on Mobile Wireless Devices

Keita Imaizumi and Vasily G. Moshnyaga
Graduate School of Engineering, Faculty of Engineering
Fukuoka University
Fukuoka, 814-0180 Japan

Abstract - This paper describes new internet-based face recognition system to be used in portable devices. In contrast to existing systems, which run computationally intensive face-recognition tasks at a mobile terminal shortening its battery lifetime, the proposed system uses mobile device only for image capturing and user-interface. All complex image processing tasks are performed by a remote high-powered network server to achieve robust and real time face recognition. The system is implemented in software and tested on Android-based Sony Tablet-S wireless terminal. According to measurements, it provides face recognition in images of 240x320 pixels in size at 10fps/sec rate with very high accuracy. The paper discusses the proposed client-server architecture and the results of its experimental evaluation.

Keywords- face recognition, face identification, network-based

I. INTRODUCTION

With emerging popularity of camera-equipped wireless multimedia devices, such as Apple’s iPhone, iPad and iPod, Google’s Android, and RIM’s Blackberry, new applications employing face recognition can further enhance usage, intelligence and context-awareness of the devices. In this paper, we focus on real-time identification of a person from a digital image or video captured by the mobile device [1]. Providing a stand-alone mobile application can potentially benefit a user in remembering people, retrieving names of people, whom he/she has met before, and/or finding helpful information about person of interest. It could be also useful for elderly people to recall faces and names enhancing their memory and social interaction. The application can also assist law enforcement when an unknown person is being compared with images in a database in real time.

Automatic face recognition has been an active research area over the last two decades. Surveys on the methods and systems proposed can be found in [2], [3]. Although there are many systems capable of performing robust face recognition at desktops, incorporating them into mobile devices is not a trivial task. Additionally to common problems, such as face illumination, occlusion, rotation and movement of the person relatively to the camera, mobile face recognition is challenged by limited energy budget of batteries, limited computing power, limited storage, limited image resolution and size, limited network bandwidth, etc. Although it is easy for a human to detect and recognize faces, performing it on hardware requires complex algorithms and many energy dissipating computations. The high computational complexity of the task makes it unsuitable for energy and resource constrained mobile devices. While work has been done on algorithms which reduce the amount of computation for face detection and identification (e.g. a unified LDA/PCA algorithm [4], Haar-like Adaboost classifiers [5], geometric features [6], Local-Binary Pattern [7] and random incremental classifier [8], platform-driven mapping [9], etc.), there is an inherent tradeoff between computation and recognition accuracy. Unfortunately, those algorithms which recognize faces accurately are extremely computationally complex, whereas computationally simple algorithms often produce incorrect results.

Several systems for mobile face recognition have been already reported in literature. Some of them (e.g. [6, 7, 8, 11, 12, 1]) utilize fast algorithms to run the face recognition process entirely on a mobile device at the cost of accuracy and operation time. The others, such as [13-15] overcome the performance limitations of mobile platforms through effective utilization of the network resources. Rather than implementing the whole face recognition process on the energy constrained mobile device, these systems transfer the majority of computation from the device to a high-powered server on the network. For example, the Bluetooth-based system [13] implements on mobile device image preprocessing and face detection while the face recognition is done on dedicated computer (server). Similarly, [14] and [15] use the DROID phone to detect a face in an image, preprocess the face calculating the Fisherfaces weights, based on which the server performs face recognition. However, even these distributed architectures still enforce mobile devices to carry out many computations, affecting both the battery budget and the processing speed. This is due to the lack of mobile processors (e.g. ARM) for executing floating point operations and also to the fact that face detection performs exhaustive image scans at different locations and scales, yielding in hundreds of thousands of sub-windows to process, which is time consuming. By tuning out the system parameters, one can further speed-up the detector but at the cost of quality degradation.

In this work, we also employ a network-based approach to reduce computation on mobile device. Unlike related systems, we use mobile device only as input and output interface; all functions of face detection and face recognition are done by the network server. The key contribution of our work is new client-server architecture, which effectively utilizes the network as a powerful computational resource to achieve face recognition at a frame rate.
The proposed network-based face recognition system utilizes a high-powered remote server and a battery operated mobile wireless device (client), equipped with a video camera. The server has access to a face database which contains face images with corresponding information of the person. We assume that the database is shared between the client and the server. Also we assume that the server is activated before the user initiates face recognition application from the wireless device. The client – server connection is set before entering the data transfer phase and released after data transmission is complete. The connection is established based on TCP/IP protocol and managed by OS through a programming interface.

The system splits the face recognition tasks between the client and the server, as shown in Fig.1. The server performs complex and accurate face recognition, while the client implements only I/O operations related to image acquisition and display of the results. The face recognition starts as the user activates the application from his/her mobile device. In this case, the client captures an image and sends it to the server with a request for processing. Upon receiving the request, the server converts image from YUV to RGB format and runs face detection and face recognition (see Fig.2) to identify the person of interest based on information stored in database. The results in terms of face rectangle and data identifying the person of interest are then sent back to the client, to be shown over the image displayed on the screen. In the next subsections we discuss the face recognition steps in details.

### B. Image conversion

The color image captured by video camera on mobile device is represented in YUV 420 SP format, allowing reduced bandwidth for chrominance components. During transmission, the luma (Y) and the chroma (U and V) components are compressed with the sample ratio of 4:1:1 (see Fig.3); so the picture has only a quarter as much resolution in color as it does in brightness. Because the server uses RGB888 image format (Fig.3, right) for face recognition, each image is converted at the server from YUV to RGB format as follows:

\[
B[i]=1192\times(Y-16)+2066\times(U-128) \quad (1)
\]
\[
G[i]=1192\times(Y-16)+833\times(V-128)-400\times(U-128) \quad (2)
\]
\[
R[i]=1192\times(Y-16)+1633\times(V-128) \quad (3)
\]

\[
B = (B[i] >> 10) \& 0xFF \quad (4)
\]
\[
G = (G[i] >> 2) \& 0xFFF0 \quad (5)
\]
\[
R = (R[i] << 6) \& 0xFF0000 \quad (6)
\]

### C. Face detection

The goal of this task is to find an area corresponding to human face in the given RGB image if any. It is implemented based on Viola-Jones algorithm [5], which transforms the RGB image into the integral image representation, and then scans it with detection window to compute Haar-like face features. The features are then applied to a cascade of 25 AdaBoost classifiers to find a true face from possible candidates. The algorithm is implemented in Intel’s OpenCV as cvHaarDetectObjects() using the Android’s Face Detector class [16]. This class provides information regarding all the faces found in an input bitmap image. The confidence factor (a number between 0 & 1) by which the face is identified, the distance between the eyes, position of midpoint between the eyes and the face’s pose (rotation around X, Y, Z axis) are the
extra details the class associates with each face. A face is considered detected if the confidence ratio is above 0.3. The result of face detection is presented by a face bounding box depicted over the input image as shown in Fig. 4(b). The derived face image is preprocessed to gray scale and histogram equalization to decrease the effects of illumination, subsampled to the database sample size and applied for face recognition. If no face is detected, the server terminates the recognition process, sending a corresponding acknowledge signal to the client.

D. Face recognition

Given a set of sample images labeled with the person identity (the set is stored in database) and the unlabeled face image \( x_i \), (derived by face detection), the face recognition problem is to identify the name of the person in the test image. We solve the problem based on the Eigenface method \([17,18]\) with Principle Component Analysis (PCA) for feature extraction and the Fisher’s Linear Discriminant Analysis (LDA) \([19]\) for feature reduction. Having a set of \( N \) face images, \( a_1, a_2, \ldots, a_N \), with each image belonging to one of \( C \) classes, \( A_1, A_2, \ldots, A_C \), the method calculates a mean face of each class \( \Phi_i = \frac{1}{N} \sum_j a_j \), the total mean, \( \Phi = \frac{1}{C} \sum_i \Phi_i \), and vectors, \( D_i = \Phi_i - \Phi \), which represent difference between the mean and the training face images. This covariance matrix \( M = \sum_i D_i D_i^T \) of the face images is then subjected to LDA to find a set \( V = \{v_1, v_2, \ldots, v_k \} \) of \( k \) orthogonal vectors (i.e. eigenvectors), corresponding to the \( k \) largest eigenvalues. The LDA takes advantage of the fact that the classes are linearly separable, selecting the projection in such a way that the ratio of the between the class scatter and the within class scatter is maximized. The within-class scatter, \( S_w \), is defined as the mean of the co-variances of samples within all classes, i.e. \( S_w = \frac{1}{C} \sum_i M_i M_i^T \). The between-class scatter is defined as the co-variance of data sets consisting of mean vectors of each class: \( S_b = \sum_i (\Phi_i - \Phi)(\Phi_i - \Phi)^T \). With LDA the problem is reduced to finding such a projection \( V \) that maximizes the total scatter \( S_T = S_w + S_b \) of the data while minimizing the within scatter of the classes:

\[
V = \arg \max V (V^T S_T V)/(V^T S_w V) = [v_1, v_2, \ldots, v_k]. \tag{7}
\]

The problem is solved through transformations such as rotating and scaling the axes of tested image in different ways. Depending on the size of the data, the projection can be done onto a lower or higher dimension. The computed eigenvector matrix \( V \) is then used for estimating the weights of projecting the target face \( x_i \) onto these eigenvectors. The weights are calculated as \( w = V^T (x_i - \Phi) \). A class \( A_j \) with the minimum Euclidean distance of weights is selected. The data related to the class is sent to the client as a result. Fig.5 shows an example. If no match has been found for the given face, the system marks it as an “unknown”.

E. Implementation

The proposed system has been implemented based on the Sony Tablet-S (1GHz ARM Cortex™-A9 dual core CPU, 1GB RAM, 16GB internal storage, 9.4-inch display, Android™ 4.03 OS) as a client and DELL PC (Intel® Core™ i5 2.80GHz CPU, 4GB memory, MS Windows 7 OS) as a server. The application software was created by using the Microsoft Visual Studio 2010 (Eclipse 3.6) and the Android software development kit. The client-server communication was implemented through Internet Socket API (ws2_32.lib) and TCP/IP transport protocol \([20]\). To support the OS-based control of the communication, a dedicated programming interface was also created. The face detection and face recognition software were programmed in C/C++ by using Intel’s Open CV 2.4.2 library \([21]\).

III. EXPERIMENTAL EVALUATION

A number of experiments were conducted to assess performance of the proposed system. The first group of experiments aimed at evaluating efficiency of face detection and face recognition software implemented on the server computer. The second group of experiment targeted performance evaluation of the entire system. Below we discuss the experiments in detail.

A. Evaluation of the face recognition software

To evaluate the ability of the developed face recognition software to detect and identify human faces correctly, we applied it to the “Faces 1999” database of 357 static frontal face images developed at Caltech \([22]\) in total. For the sake of experiment, all the images have been manually transformed to grey-scale representation, resized and trimmed to face area only as shown in Fig.5. In such a way we prepared an experimental database of 17 different persons (9 men and 8 women), with each person represented by 10 images (170 images in total). Fig.6 exemplifies face images of the same person. All face images were 120x120 pixels in size and labeled by a unique digital tag for identification.

In the experiment, we used all 357 original pictures from Faces 1999 as an input to the developed face recognition software.
Namely, each image was captured by Logicool C600 video camera (640x480 frame size) from a display and used as an input image of a person to be identified. For each image we evaluated whether the face detection and face recognition produced correct or false results. To consider effects of face inclination, we repeated the test for each image by rotating the camera with 5° increment and determining the maximal inclination angle at which the results were correct. Fig.7 illustrates an image inclined by 25°.

Fig.8 shows the results in terms of the recognition ratio and the maximal angle at which the recognition was correct. As one can see, the recognition ratio is high, reaching 98% in average. Though face inclination affects the results, the software can correctly recognize faces inclined by as much as 24° and 12.5° on average.

The results revealed that glasses, mustaches, beard, hairstyle, gender, age, race, etc. had no bad effect on the face recognition if the corresponding features are reflected by samples stored in database. Nevertheless, there are several factors which can impede the face recognition performance. One is face brightness. Dark images might either lead to inability to detect a face (see Fig.9, top) or cause incorrect face identification (see Fig.9 bottom). Another factor which affects the results is the number of sample images in database. For example, if database provides only a single image per person, the face recognition ratio becomes 76.5%. The recognition rate is also affected when the number of samples in each class of database is uneven. If for example, one person is represented by 10 images while the others by 2 or 3, the recognition rate does not exceed 83.3%. To increase the recognition efficiency, the number of samples in a class (i.e. for a person) has to be not only large but also the same as in the other classes.

To evaluate the software efficiency in identifying faces of real people in typical environment, we added 60 sample face image of 6 students (10 images per person, 120x120 pixels image) to the database and conducted a set of tests, in which each student appeared before the video camera at a distance ranged from 20cm to up to 2.5m. The room illumination was relatively good and the background was typical for computer lab, as shown in Fig.10. In the experiment, the students were asked to conduct five behavioral patterns: face the camera frontally with open and close eyes, turn the face up and down, turn the face to the right and to the left. Each pattern was 5 sec long and repeated twice by each user. The results (Fig.11) showed that the system was able to distinguish and track faces correctly at up to 2 m
distance, recognizing them with 91% rate at up to 1.6 meter distance unless the face rotation left/right and inclination in vertical or horizontal directions is over 15°. With larger rotation, inclination and distance to the camera, the misdetection rate increases. Although our experience shows that extending the database to rotated and inclined faces improves the system performance significantly, recognizing faces at farer distances requires a more powerful camera to be incorporated into the mobile device.

### Table I. System Performance (at 1m Distance)

<table>
<thead>
<tr>
<th>Frame size (pixels)</th>
<th>160×120</th>
<th>320×240</th>
<th>848×480</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aspect ratio</td>
<td>1.3</td>
<td>1.3</td>
<td>1.8</td>
</tr>
<tr>
<td>Processing speed (fps)</td>
<td>10</td>
<td>3</td>
<td>0.43</td>
</tr>
<tr>
<td>Recognition accuracy (%)</td>
<td>96</td>
<td>100</td>
<td>71</td>
</tr>
</tbody>
</table>

**B. Evaluation of the network-based system performance**

The system performance was tested by running the developed face recognition system wirelessly from the Sony Tablet-S device using the 802.11n WiFi protocol and communication resources of gigabit local area network. As Tablet-S device provides three different picture formats for video capturing, we evaluated the system performance using each of them while applying the system to identification of same 6 persons. Fig. 12 shows an image observed at the screen of Sony Tablet-S device.

Table I summarizes the results in terms of the processing speed and the recognition rate versus the size and the aspect ratio of images, captured by the mobile device at the 1m distance. We observe that the system processing speed decreases as the image frame size increases. At the frame size of 160x120 pixels, it achieves speed of 10 frames per second (fps) with the recognition accuracy of 96%. As the image aspect ratio equals 1.3, i.e. the ratio of images used for sample generation, the face recognition quality is high. At the image size of 320x240 pixels, the recognition accuracy even increases reaching the maximum level due to better image quality. However, the speed of processing (320x240) images is 3 fps. The cause of this speed drop can be explained by the longer time required for transmitting and processing large images. As the frame size becomes very large (848x480 pixels) these delays become increasingly high, decreasing the speed to 0.46 frames per second.

Table II puts our system in perspective to the mobile face-recognition systems reported in the literature and online. Here the processing time refers to the time in seconds required to process one image frame. We observe that the proposed system outperforms the existing solutions by both the processing speed and the recognition ratio. In comparison to the related solutions, which operate on small (160x120) images at the 1 fps rate at most, our system runs 10 times faster. Moreover, it supports face recognition in larger images which is more preferable for the users. Although it looks that more powerful resources, which our system exploits in comparison to the others, is the main cause of its speed-up. However, it turns out that as long as the training is done beforehand, the bottleneck of typical face recognition system is actually the face detection, not the recognition, since the recognition images are fairly small. Detecting a face on a mobile device and transmitting it to the server is much longer than sending and processing the image at the server. Unlike the others, our system does not run the face recognition on mobile device and therefore is fast. It turns out that new client-server architecture that relieves the mobile device from any complex processing is the key the speed-up achieved by the proposed system.

**IV. CONCLUSION**

In this paper we presented novel client-server architecture for the network-based face recognition. Experiments showed that our system outperforms the related systems in both the processing quality and speed, allowing real-time (10fps) robust (96% accuracy) face recognition for people located up to 1.6m distance from the mobile device. In the current work we restricted ourselves to a simple case of a singular person and frontal face recognition. We are currently working on mobile recognition of multiple faces, extending the work to inclined and rotated faces, recognition of faces with partial occlusion, as well as issues related to power-aware optimizations of face recognition algorithms. Also a representative database is important to the success of the face recognition system. Therefore, in order to further improve the system, a larger mobile face database is necessary. Problems related to database development, automatic generation of sample images, etc. will be also investigated in the future.
REFERENCES


