International Journal of Security and Its Applications
Vol.9, No.1 (2015), pp.51-60
http://dx.doi.org/10.14257/ijsia.2015.9.1.06

Implementation: Mobile Face Identity Authentication System on Android Platforms

Pinjie Ye, Mengmeng Yu and Minghui Wu*

Department of Computing and Computer Science, Zhejiang University City College, Hangzhou, Zhejiang, P.R.China
Corresponding author email: mhwu@zucc.edu.cn

Abstract

This paper proposes a mobile face identify authentication system. In this system, Android application has to capture face, and verify the face by Web services. It is introduced how to implement an Android Client of this system in details. Using MB-LBP features, AdaBoost and CamShift algorithm, get face images by camera on mobile device. Then, rotate, crop the face images and convert them to grayscale in order to reduce the amount of face data. At last, post data and get validation results using sub-thread to realize real-time face verification.

Keywords: *Android; MB-LBP; CamShift; face detection; Handler*

1. Introduction

Due to loose management of universities, substitution for exams and courses is rampant, these cheating behaviors affect the fairness of the competition, and these cheating behaviors are spreading in the absence of strong measures to combat. These phenomena may impact on the students' learning motivation, reduce students' integrity, then school education order is difficult to maintain, and hinder the development of education severely. With advances in technology, the development of information technology, mobile Internet, cloud computing, artificial intelligence are developing fast, scanning stranger's face to get his identity by certified devices has the potential to achieve.

Limited by current smart phone design and performance, we present a face authentication system based on distributed computing. Smart phone Gets a human face through a local human face detection. Then facial image and additional information will be sent to the server. The server finds out the most similar face and its related identity information from face database using MapReduce framework. Finally get the certification result according to a preset threshold value.

For getting facial images in this system, we need to implement a fast and accuracy algorithm to detect human face firstly. And also at the same time, the client has to post the detected facial images to the server. There are two main parts of our system. We will introduce each of these two parts briefly below and then describe them in detail in subsequent sections.

The first part of this paper is a fast and accuracy algorithm for detecting human face. Because of the design of this authentication system scheme, for using distributed face verification or recognition, the client has to send images including faces to the server firstly. In view of the low speed of mobile Internet and its expensive fee, the pixels of background are less, the facial information is more in an image of fixed size. Thus, we need local face detection in client to capture an image with human face and crop the face rectangle from facial image. In the first part of this paper, we use MB-LBP [1]

ISSN: 1738-9976 IJSIA

features to represent facial images instead of Haar-like [2] features and regular LBP [3] features. And we train the classifiers using AdaBoost [4]. For improving the speed of capturing facial image, we use CamShift [5] to track human face for narrowing search area. Using MB-LBP features, AdaBoost and CamShift algorithm, the mobile device can capture facial image from preview frame of camera almost real-time. After capturing, rotate, crop the face images and convert them to grayscale in order to reduce the amount of face data and standardize the data size.

The second part of this paper is the implementation of face verification by distributed computing. In this paper, we just describe the communication of the client and the server. In order to avoid ANR (application not responding) error, the android client has to post data and get validation results using sub-thread to realize almost real-time face verification through opening interfaces of the server.

At last, we apply this solution to the campus identity authentication, like exam management system and other similar authentication systems. The solution we implement running on Note3 can reach 30fps of face detection, and its correct rate is above 90%. In a good network condition, the cost time of verifying a person is about 2 seconds in average.

2. Campus Identity Authentication System Design

2.1. System Function

This system contains two projects: a client project and a server project. The client project includes an APP for teachers and a website management system for administrators. The APP provides a specific set of functions, such as face verification, face recognition and invigilation records. And the web management system offers some data management services include loading exam arrangement from EXCEL, uploading face samples, managing roles and other aspects of data. In this system, the key face verification algorithm is implemented on the server cluster. The main functions of this system are shown as Figure1:

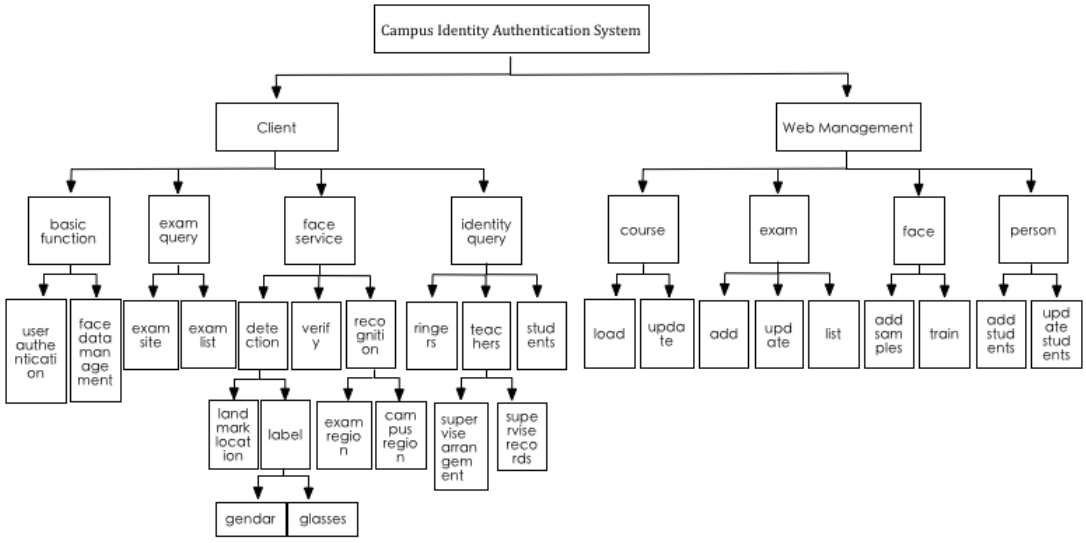

Figure 1. Campus Identity Authentication System Function Structure Diagram

2.2. System Function

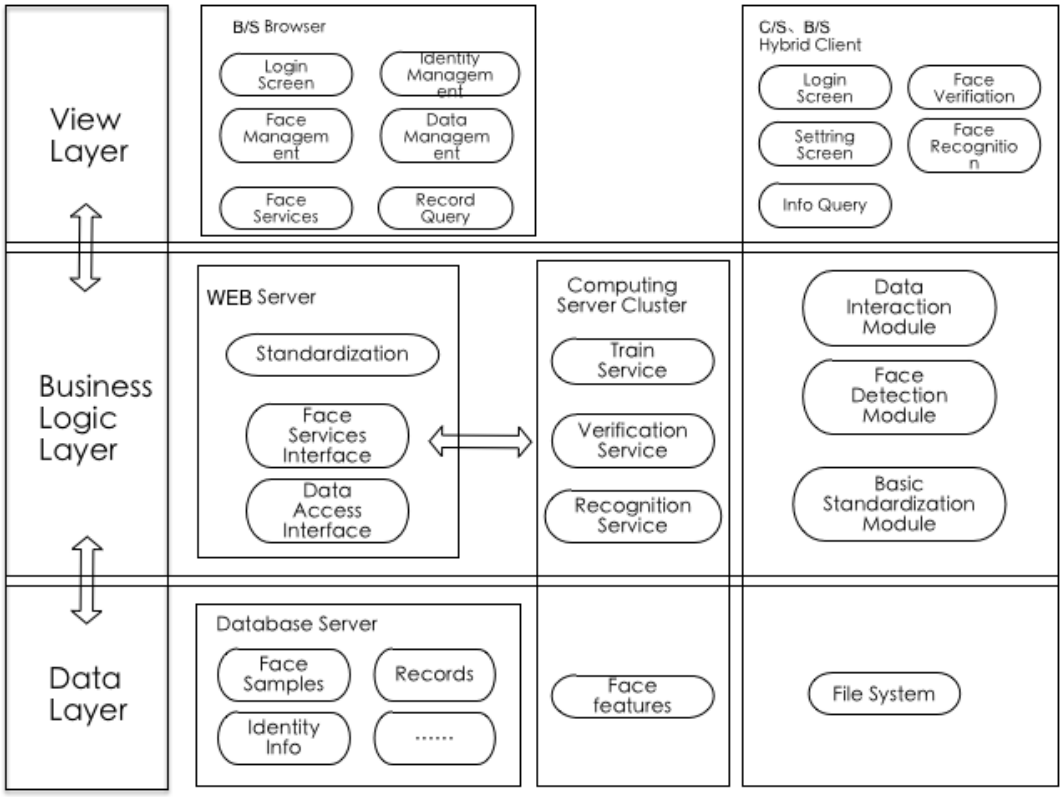

Figure 2. System Architecture Diagram

The system uses C/S and B/S hybrid architecture model, as shown in Figure 2:

The system uses C/S structure to capture face from capturer on mobile device and uses B/S structure to get identify information, invoke face verification and recognition algorithms from the server cluster. Be restricted by the low speed of network transmission and the expensive charge of network flow, we crop, resize and recompress the face images on APP to reduce the amount of facial data.

Taking into account that the calculations of face verification and recognition algorithms are large and it is not reasonable to store face database on mobile device, we put the key algorithms and data on the server cluster.

3. Implementation of Face Detection on Android

Android is a mobile phone operating system based on Linux kernel developed by Google. Owe to its open-source and freedom, Android attracts lots of developers all over the world. After Google launched Android NDK to simplifies developing native C/C++ Android applications, more and more mature C/C++ libraries for developers can be uses in their applications. In this system, we use OpenCV to implement face detection.

In Android application, for the realization of face identity authentication, we need to capture face images by real-time face detection at first. We uses MB-LBP[1] feature to represent facial image, then uses AdaBoost[4] learning algorithm to get the result of classifying, at the same time, we uses CamShift[5] to track moving face in order to accelerate

face detection and reduce the mistake rate. At last, we reduce the amount of facial data by face standardization to for being conducive to network transmission.

3.1. MB-LBP (Multi-block Local Binary Patterns) Feature

In 2001, Viola and Jones proposed a fast algorithm for computing Haar-like features and a boosting-based real-time detector. It is a breakthrough in face detection research. On PC, [4]'s performance of real-time detection is numerically robust, but it is not good on Android. The speed of detection only reaches 7 fps without optimization, far below 24fps. So, in order to accelerate face detection, and also maintain the correct rate, we use MB-LBP to replace Haar-like to represent facial images.

MB-LBP [1] is the improvement of original LBP feature. The idea of LBP [3] is that the pixels of an image are labeled as a binary string or a decimal number by thresholding the 3×3 neighborhood of each pixel with center value. LBP is less sensitive to changes in illumination and it is a non-parametric operator. It is a powerful local descriptor for microstructures of images, but LBP cannot capture larger scale structure that may be more important in face representation. Therefore, MB-LBP is proposed to overcome the limitations of LBP. Unlike the labeled object is pixel in LBP, the MB-LBP operator is defined by comparing the central rectangle's average intensity g_c with its neighborhood rectangles' average intensities $\{g_0, g_1, \ldots, g_7\}$, the result can be calculated as follows:

$$MB\text{-}LBP = \sum_{i=0}^{7} s(g_i - g_c) \cdot 2^i, \quad s(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases} \qquad (1)$$

Comparing with original LBP, a large number of MB-LBP features are at different scales and locations. It may offer more significant features to Adaboost learning.

On mobile device, the reason why we choose MB-LBP inside of Haar-like is that the amount of MB-LBP features is far less than Haar-like features. In a sub-windows size of 24× 24, there are 117941[6] Haar-like features. And the amount of MB-LBP features is only 8464, about 1/14 of Haar-like features. Hence, the speed of training process and face detection using MB-LBP is faster.

3.2. CamShift Tracking

CamShift (Continuously Adaptive Mean Shift) [6] is an improved algorithm based on Mean-Shift. Its basic idea is that all frames of video images for Mean Shift operations set the results of the previous frame as the initial search window of the next frame. When the location and size of object change, the target area can adjust automatically. The step of the algorithm as follows：

Step 1： Set the whole image as the region of interest of the probability distribution image.

Step 2：Initialize the size and location of search window.

Step 3：Calculate a color probability distribution of the region of search window.

Step 4：Calculate the new size and location of search window using Mean Shift algorithm.

Step 5：for the next frame, set the search window to the calculate result of Step4. Go to Step3.

3.3. Face Standardization

Face standardization, as preprocessing of face image, means that get the face region from the image, and normalize gray and size of the image. The mission of face standardization is to make a face set of same person under different conditions more consistent. In addition, preprocess face image in Android application can also reduce facial data size and speed up network interaction. In this system, in average, the size of a facial image (320×240) without processing is about 300KB. After face standardization, the size can be reduced to about 10KB. The steps of face standardization in Android application as follows:

1) Determine the location of two eyes center E_R , E_L in the detected face.
2) Rotate：Let E_R connect to E_L , and rotate the segment to be horizontal 。
3) Crop：Set the distance of two eyes center as standard distance d. Set the length and width of face rectangle as 2.2d(0.2d is artificial reserve length). Set the position on Perpendicular bisector of $E_R E_L$ below the line $E_R E_L$ 0.5d as the center of face rectangle. Crop the face rectangle from facial image.
4) Scale：Scale the face rectangle to make d equal 32 pixels because the input size of face verification on Server is 64×64.
5) Convert face rectangle to grayscale.

3.4. Results of Real-time Face Detection

In this experiment, we detect 5 20-second videos, the preview videos captured by rear camera on Samsung Note3 with 2.3GHz qualcomm snapdragon 800 processor and Android 4.3 operating system. The resolution of each video is 1920×800.

The first method uses Haar-like feature and Adaboost Learning algorithm. The second method adds CamShift tracking to the first method. The last two methods using MB-LBP feature replace Haar-like feature. The result with each face detection method as below:

Table 1. Results of Real-time Face Detection

METHOD	CORRECT RATE (%)	COST TIME OF DETECTION OF ONE FRAME (ms)
Haar-like+AdaBoost	91%	135.2
Haar-like+AdaBoost+CamShift	95%	78.5
MB-LBP+AdaBoost	92%	51.1
MB-LBP+AdaBoost+CamShift	97%	28.7

Overall the results show that he speed of face detection using MB-LBP is faster than using Haar-like. And using CamShift can reduce the cost time of detection and also improves the correct rate. Among these results, the last method can reach 30fps and its correct rate is 97%, meeting the requirements of real-time face capture.

4. Implementation of Face Verification on Android

4.1. Implementation of Real-time Communication

When the Android application captures a facial image, it has to post the image to the server and request face verification or face recognition services. Procedures of face verification and face recognition services are shown in Figure 3:

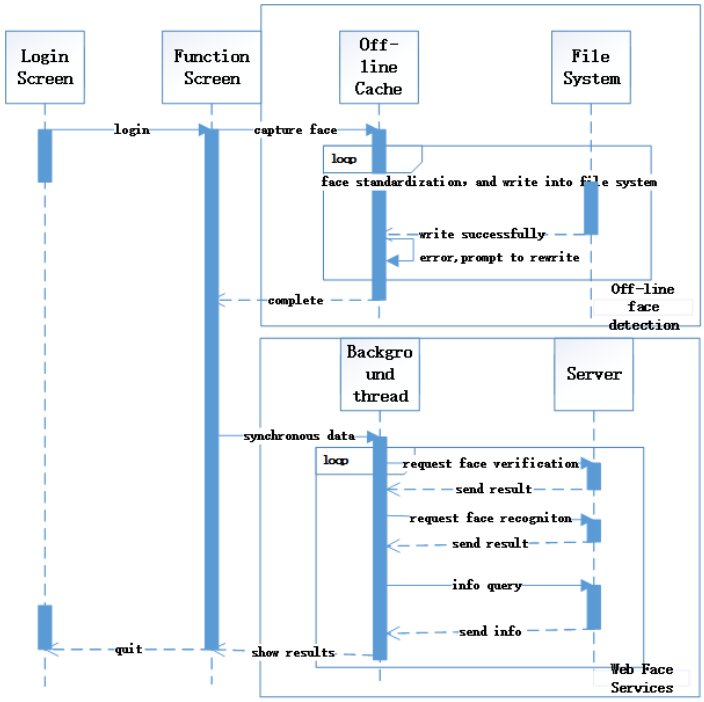

Figure 3. Procedures of Face Verification and Face Recognition Services

To interact with the server, the Android application needs to send requests and show the acquired results on UI elements. While, in Android applications, if an operator may block the UI thread, ANR (application not responding) dialog will occur. So in order to avoid this error, the Android rule for updating UI is that do not block the UI thread and also do not access the Android UI toolkit from outside the UI thread [7].

To fix above problem, we use Handler method. When the Android application send a request, the steps of interaction as below:

1. Create a sub-thread binding with looper. The looper loops over a MessageQueue which contains messages to be dispatched. The sub-thread can execute commands based on messages obtained from MessageQueue.
2. Create an instance of the Handler class, overwrite the method of HandlerMessage, and Bind with the looper.
3. Call the sendMessage method of Handler, obtain message from Handler object. The actual task of managing the queue is done by Handler which is responsible for handling (adding, removing, dispatching) messages in the MessageQueue.

4.2. Providing API Service

The services of server include data management, face verification, face recognition and so on. The main 5 APIs are shown as below:

1) Data management interface. Administrators and users can access some data by this interface。 In particular, the data of face features is stored in the database on computing server cluster and the set of facial images is stored in the file system on data server.
2) Face detection interface。Different with the face detection algorithm on Android, this algorithm on server can locate key points of face.

3) Feature extraction interface。Based on facial landmark location, normalize facial image more closely. Then, post the processed image to computing server cluster. The server cluster will extract the facial feature using machine-learning algorithm and save the feature data into local database.
4) Face verification interface. Post the facial images and target ID, then acquire the degree of confidence.
5) Face recognition interface。Post the facial images and specifying the search scope to the server, then obtain a list of similar people. Because the time of recognition may be very long, so it offers asynchronous query by session_id.

5. Experiment and Analysis

The test hardware is Samsung Note3 mobile phone. In this experiment, we only test the cost time of face verification, because the implement of face recognition is asynchronous. We set the size of preview window is 320×240 and the format of preview image is YUV420. After processing, the size of image is converted to 100×100, and the format is bitmap. In a good network condition, the cost time of verifying a person is shown in Table 2:

Table 2. The Cost Time of Real-time Face Verification

VERIFICATION TIMES	CAPTURE TIME/s	COMMUNICATION TIME（SEND+RECIEVE）/s	VERIFICATION TIME/s	TOTAL COST TIME/s
1	0.03	0.8	0.5	1.33
2	0.06	1.6	1	2.16
3	0.09	2.4	1.5	2.99
4	0.12	3	2	3.82
5	0.15	3.6	2.5	4.65

In actual operation, a right face only needs to be verified one or two times. If it is verified over 5 times, it means it is not the right face. The screenshots of face verification is shown in Figure 4:

Figure 4. The Screenshots of Face Verification

6. Conclusion

It is introduced the implement of mobile face identity authentication system on android platforms. On Android, we complete three main functions including real-time face capture, interaction with server and show obtained result on UI. In the implement of face capture, we improve the speed and correct rate of face detection using Adaboost learning algorithm, MB-LBP operator and CamShift tracking algorithm. Then normalize the facial image by eye locations to reduce the size of communication data. At last, using multithread, make the Android application can capture new facial image and request face verification service at the same time. Thus, the face verification is close to real-time in a good network condition. In the practice, the solution we implement running on Note3 can reach 30fps of face detection, and its correct rate is above 90%. In a good network condition, the cost time of verifying a person is about 2 seconds in average.

Acknowledgements

This paper is partly supported by the Academic project of the Young Academic Leaders in Colleges of Zhejiang Provinece (No. pd2013453). Open Fund of Mobile Netwok Application Technology Key Laboratory of Zhejiang Province, Innovation Group of New Generation of Mobile Internet Software and Services of Zhejiang Province (No. 2010R50009).

References

[1] L. Zhang, R. Chu, S. Xiang, S. Liao and S. Z. Li, "Face detection based on multi-block lbp representation", In Advances in biometrics, Springer Berlin Heidelberg, **(2007)**, pp. 11-18.
[2] C. P. Papageorgiou, M. Oren and T. Poggio, "A general framework for object detection", Computer vision, sixth international conference on. IEEE, **(1998)**, pp. 555-562.
[3] T. Ojala, M. Pietikainen and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns", Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 7, no. 24, **(2002)**, pp. 971-987.
[4] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features", Computer Vision and Pattern Recognition, 2001. Proceedings of the 2001 IEEE Computer Society Conference on. IEEE, **(2001)**, pp. I-511-I-518.
[5] J. G. Allen, R. Y. D. Xu and J. S. Jin, "Object tracking using camshift algorithm and multiple quantized feature spaces", Proceedings of the Pan-Sydney area workshop on Visual information processing. Australian Computer Society, Inc., **(2004)**, pp. 3-7
[6] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection [C]", Image Processing. 2002. Proceedings. 2002 International Conference on. IEEE, **(2002)**, pp. I-900-I-903
[7] W. Yan and J. Ye, "Multithread technology in the development of Android application", INFORMATION and COMMUNICATION, vol. 1, **(2012)**, pp. 46-47.

Authors

Pinjie Ye (1991-) he got his bachelor's degree in Computer Science from Zhejiang University City College in 2014. His research interests include machine learning, computer vision and distributed computing.

Minghui Wu (1976-)was born in Leping, Jiangxi Province of China. He received the BS degree in Computer Science and Engineering from Nanchang University in July 1997 and MS, Ph.D. degrees in Computer Science and Technology from Zhejiang University in March 2000 and June 2011, respectively. Now, He serves as a Professor in Computer Science at Zhejiang University City College. His major interests include Software Engineering, Mobile Internet, and Embedded System.