Effect of Computation Offload on Performance and Energy Consumption of Mobile Face Recognition

Nanoka Sumi, Akiya Baba, and Vasily G. Moshnyaga
Graduate School of Engineering, Fukuoka University
8-19-1 Nanakuma, Jonan-ku, Fukuoka, 814-0180 Japan

Abstract – Computation offloading is a paramount technology to leverage network resources for mobile devices. This paper studies effect of computation offloading on efficiency of mobile face recognition. It compares offloading alternatives used in existing mobile face-recognition system and reports on their efficiency in terms of energy consumption, processing time and recognition accuracy. The offloading method which leads to the best energy-performance tradeoff is outlined.

Keywords- *face recognition, network-based system, computation offloading, performance, energy consumption, analysis*

I. INTRODUCTION

A. Motivation

With explosive use of camera-equipped wireless devices, such as laptops, tablet PC and smartphones, new applications employing face recognition become very popular. A stand-alone face recognition application can potentially benefit a user in remembering people, retrieving name of a person, whom the user has met before, and/or finding helpful information about person of interest. The application could be also useful for elderly people to recall faces and names, enhancing their memory and social interaction. It can also assist law enforcement when an unknown person is being compared with images in a database in real time.

Incorporating face recognition into mobile devices, however, raises several challenges. Although it is easy for a human to recognize faces, performing the task by computer is hard. The difficulty stems from variation in the facial appearance because of face illumination, face occlusion, face rotation, and movement of the person relatively to the camera. Therefore, most recognition methods (surveys can be found in [1-3]) use specific constraints, such as frontal view, homogeneous background, mug shot images etc. to limit the variability. Providing accurate face recognition without constraints on face variation is still a big challenge.

Another challenge is implementing computationally intensive face recognition methods on resource constrained mobile devices. Due to limited processing capabilities of the mobile platforms, even the reduced task of frontal face recognition takes long execution time. Moreover, it attributes to high energy consumption, making the task almost unsuitable for battery operated devices. While efforts have been made to reduce computations of face detection and face identification algorithms, there is an inherent tradeoff between computations and recognition accuracy. Those algorithms

which recognize faces accurately are extremely computationally expensive, whereas computationally simple and fast algorithms often produce incorrect results.

Computation offloading is an indispensable technology to leverage network services for mobile devices. It migrates computationally demanding tasks to a more resourceful computer (i.e. server) in order to improve application performance while saving the energy of mobile device. Offloading unfortunately does not come free; it depends on a number of factors, such as the network conditions, bandwidth, and the amount of data transmitted through the network. Many algorithms have been proposed to make offloading starting from the level of programmer decisions [4] to compiler-based [5] and automatic solutions [6]. The decision can involve static schemes [7]-[9] or dynamic formulations that can change with network conditions [10], [11]. A survey of the offloading techniques is given in [12].

The goal of this paper is to analyze the energy and performance benefits of computation offloading and task migration for mobile face recognition system.

B. Related Research

Several face recognition systems have already incorporated computation offloading to cope with the limitations of mobile platforms through effective utilization of the network resources. Rather than implementing the face recognition process fully on a mobile device (e.g. [13-16]), the network-based systems [17-20] transfer the majority of computation from the device to a high-powered server of the network. In [17], the Bluetooth network-based system does image preprocessing and face detection on mobile device, while face identification is carried out on server. Similarly, [18],[19] 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. These distributed client-server architectures require mobile devices to perform 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. Imaizumi, et al. [20] mitigate all face detection and identification tasks to the server, while running on the client only I/O operations related to image acquisition, image

The work was sponsored by fund No.135010 from the Central Research Institute of Fukuoka University, Japan

Figure 1: The face recognition process

transfer and display of the results. Such task offloading

Figure 2: An illustration of an input image (a), the face detection result (b) and the result of face recognition (c)

considerably reduces computational load of mobile devices but requires the whole image to be transferred to the data server, increasing the amount and energy consumption of the data transfers in the network.

Does this computation offloading lead to energy reduction of mobile device? What offloading method is the best from the client energy perspective? Although there have been efforts to control offloading by energy consumption (e.g. [21,22,23], no studies on effects of offloading on the energy consumption of mobile face recognition have been reported yet.

C. Contribution

This work is the first attempt to quantify and qualify the computation offloading in the mobile face-recognition system from the performance and energy consumption perspective. We explore the offloading methods employed in real-time network-based face recognition systems; analyze their effects on performance and energy consumption and determine the method which provides the best energy-performance tradeoff.

The paper is organized as follows. Section II describes the face recognition process and methods used for its implementation. Section III explains the offloading methods which we examined. Section IV reports the results. Section V summarizes our findings and outlines work for the future.

II. THE FACE RECOGNITION PROCESS

The aim of face recognition is to identify or verify one or more persons from video image of a scene using a stored database of faces. Typically the recognition process is done sequentially (as illustrated in Fig.1) in three steps: image acquisition, face detection, and face identification. Details of these steps are given below.

A. Image acquisition

During this step, the color image frame captured by video camera of mobile device is preprocessed for noise reduction

and transformed into an intermediate format for further processing. For example, if the input frame is in YUV420SP and the face data base combines grayscale pictures, the intermediate image can be represented by a single Y plane. In this case, transformation does not invoke any computation: it is done by selecting the Y-pixels from the input image.

B. Face detection

The goal of this step is to find an area corresponding to human face in the RGB image if any. The state-of-the-art technology employed in various consumer products is Viola-Jones fast face detection algorithm [24], which transforms the RGB image into the integral image representation, and then scans it with detection window (20x20 pixels) to compute Haar-like face features. The features are then applied to a cascade of AdaBoost classifiers to find a true face from possible candidates. Because the number features and the number of classifiers is large, even this “fast” and computationally optimized algorithm may involve millions of computations and memory accesses per frame. After the location of the face is found, we search the eyes and crop it to exclude the background and decimate it down to a size of 120x120 pixels. The result of face detection is depicted by rectangle over the input image (see Fig.2, b).

C. Face recognition

Given the face image (x_i), and database of sample face images $Y=\{y_j\}$, with each image (y_j) tagged by an identifier p_j of the corresponding person, the problem of face recognition is to identify the image y_j that matches x_i and returns identity p_j of the person in the test image. Among a variety of techniques, the Eigenface method [25] with Principle Component Analysis (PCA) for feature extraction and the Fisher's Linear Discriminant Analysis (LDA)[26] for feature reduction is considered the best to achieve high discriminative power and robustness of face recognition.

Having a set of N face images, $a_1, a_2, \dots, a_N$, with each image belonging to one of C classes, $A_1, A_2, \dots, A_C$, the method calculates a mean face of each class $\Phi_j = (1/n) \times \sum_{i=1}^n a_{ij}$, the total mean, $\Phi = (1/C) \times \sum_{j=1}^C \Phi_j$, and vectors, $D_j = a_j - \Phi_j$, which represent difference between the mean and the training face images. This covariance matrix $M=\{D_1, D_2, \dots, D_C\}$ of the face images is then subjected to LDA to find a set $V_o=\{v_1, v_2, \dots, v_k\}$ 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. $S_W = 1/C (\sum_{j=1}^C M_j \times M_j^T)$. The between-class scatter is defined as the co-variance of data sets consisting of mean vectors of each class: $S_B = \sum_{j=1}^C (\Phi_j - \Phi)(\Phi_j - \Phi)^T$. With LDA the problem is reduced to finding such a projection V_o that maximizes the total scatter $S_T = S_W + S_B$ of the data while minimizing the within scatter of the classes:

$$V_o = \arg \max_V (V \times S_T \times V^T) / (V \times S_W \times V^T) = [v_1 \ v_2 \ \dots \ v_k]. \quad (1)$$

Figure 3: The offloading method 1

The eigenvector matrix V_o is then used for estimating the weights of projecting the target face x_i onto these eigenvectors. The weights are calculated as $w = V_o^T(x_i - \Phi)$. 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.2(c) shows an example. If no match has been found for the given face, the system marks it as an “unknown”.

III. THE OFFLOADING METHODS

We studied two offloading methods. The first one is commonly used in many mobile face recognition systems [18]-[19]. It exploits the fact that storing and accessing the database makes the biggest bottleneck in mobile face recognition due to limited memory of mobile device. The key idea is to split the recognition process between the client and the server in order to run complex and accurate face identification on the network server while all the other tasks are done by the client, as shown in Fig.3. To support this offloading scheme, the face area, derived by face detection, has to be sent from the client to the server, and the results of face identification have to be transmitted from the server to the mobile client for display.

The second offloading method is advocated in [20]. It is based on observation that face detection is the most computationally intense task of face recognition. For example, if we assume the face detection is done by Viola-Jones algorithm [25] that scans an image of 360x280 pixels in size with a window of 20x20 pixels, repeatedly displacing the window by 1 pixel, and the number of Haar-like features evaluated per displacement on average is 5 (out of 12, no rotation), the average number of Ada-Boost classifiers is 5 (out of 16) and the number of scaling levels is 4 (out of 6), then it takes over 26M of floating point computations and over 1M of memory accesses to detect a frontal face in a single integral image frame! (Note this count does not include the cost of processing overlapping windows, computing an integral image representation, eyes localization, etc. The total amount of computations is much higher). Since both the processing time and the energy consumption of mobile devices are proportional to the amount of computation and memory accesses performed, offloading the face detection task could improve performance and extend battery life of mobile device.

Figure 4: The offloading method 2

Fig.4 illustrates the main idea of the offloading method 2. As the mobile client captures and transforms an input image, it sends the whole image frame to the server with a request for processing. After that, the client waits for the result of the offloaded computation regularly polling the interface. Upon receiving the request, the server performs all computations involved in face detection and face identification. The results of these steps are then returned to the client for display.

Both offloading methods have advantages and disadvantages. The first method enforces the mobile device to perform computationally and energy expensive face detection. However, it generates less traffic in the network than the second one, since the face area (d_i) usually is smaller than the frame area (D). As image contains H different faces, the amount of data exchanged between the client and the server by the method 1 becomes $\sum_i^H d_i$. If P_d is the power required to send data from the mobile device over the network, P_m is the power consumed by the mobile device on face detection, S_m is the speed of the mobile device, and W is the computational workload of face detection, then the energy consumed by the device to perform the task and transmit the results to the server over the network of bandwidth B is:

$$P_m \times W / S_m + P_d \times (\sum_i^H d_i) / B \quad (2)$$

If P_L is the power consumed by the device in waiting mode, and S_s is the speed of the server, the first offloading method saves more energy than the second, only when

$$P_m \times W / S_m + P_d \times (\sum_i^H d_i) / B < P_L \times W / S_s + P_d \times D / B \quad (3)$$

The parameters P_m , P_d , P_L , S_m , S_s , and D can be accurately estimated in advance for the given mobile device and the server. The other parameters: W , B , H and d_i strongly depend on the image content and the network conditions. Therefore, the only way to compare the offloading methods is to implement them in software and evaluate empirically in real conditions.

IV. IMPLEMENTATION

To examine the efficiency of task offloading we implemented three versions (V0, V1, V2) of mobile face recognition system on a 2012 Asus Nexus 7 tablet PC that runs Android™ 4.3 OS and is equipped with NVIDIA Tegra3 T30L, 1.3GHz CPU, 1GB internal storage, and 7-inch display. The version V0 does not use any computation offloading, executing the face-recognition process entirely on mobile device [13-16]. The

Figure 5: Examples of sample images obtained from the Face1999 database for the same person

Figure 6: An example of correct (left picture) and incorrect (right picture) face recognition on images from the Face1999 database

face database in this version was stored on 64GB flash card of mobile device. The versions V1 and V2 implemented the first and the second offloading methods, respectively. Versions V1 and V2 utilized a high-powered server implemented on Dell Servo PC (Intel Core i7 2.80GHz CPU, 4GB RAM, and MS Windows 7 OS). In these versions, the data base was stored on the server's HDD; the server was active before the user initiated the face recognition application on the mobile device, and the communication media was 802.11nWiFi with resources of gigabit local area network. The client-server connection was implemented as an Internet Socket API (ws2_32.lib) based on TCP/IP transport protocol and OS control, assuming that a TCP connection is set before the data transfer phase and released after the data transmission phase. To support the OS-based communication control, a dedicated programming interface was also developed. A remote procedure call for task activation was implemented through message passing. The `task_start` message was used to send the identity of the scheduled task to the opposite side (either the server or the client). Upon receiving this message, the receiver started execution of the identified task, while the sender blocks its current task. The `task_end` message was used to terminate a previously scheduled remote task. Upon its reception, the previously blocked task resumed the execution. The `data_send` and `data_request` messages were used to carry data to its recipient and to request data from its recipient, respectively.

The application software was created by using the Microsoft Visual Studio 2010 (Eclipse 4.3) and the Android software development kit. The face detection and face recognition software were programmed in C/C++ by using Intel's Open CV2.4.3 library. The face detection algorithm was implemented in Intel's OpenCV as `cvHaarDetectObjects()` using the Android's Face Detector class. This class provides information regarding all the faces found in an input bitmap image. The confidence factor (a number between 0 & 1) with which it identified the face, 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 provides associated with each face. A face is considered detected if the confidence ratio is above 0.3. The result is represented by the face rectangle and face tag (20B in total).

TABLE I. FACE RECOGNITION TIME PER IMAGE (MILLISECONDS)

Image size	360×288			180×144		
Software version	V0	V1	V2	V0	V1	V2
Image acquisition	26	26	26	26	26	26
Face detection	624	624 +16	326 +142	536	402 +16	213 +28
Face identification	768	153	153	468	153	153
Results transfer	-	6	6	-	6	6
Total	1418	825	623	1030	603	466

V. EXPERIMENTAL EVALUATION AND RESULTS

A. Face recognition rate

To evaluate the ability of the developed 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 [27]. For the sake of experiment, all the images have been manually transformed to grey-scale representation, resized and trimmed to face area only. 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.5 exemplifies different image samples stored in database for the same person. All face images were 120x120 pixels in size and labeled by a unique digital tag for identification.

The results showed that recognition accuracy does not depend much on the offloading methods. It was 93% for version V0, and V1 and 96% for V2 even when images were inclined up to $\mp 25^\circ$. Fig.6 (left) shows an example. Glasses, mustaches, beard, hairstyle, gender, age, race, etc. had no bad effect on the face recognition when the corresponding features are reflected by samples stored in database. However, dark, occluded or rotated face images might either lead to inability to detect a face or cause incorrect face identification (Fig.6, right).

To study the software efficiency in recognizing faces of real people in typical environment, we added 60 sample face images 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 ranging from 40cm to 3m. In all the tests, the face illumination was relatively good and the background was typical for computer lab. As results revealed, the developed software could detect, track and identify multiple faces with 93% accuracy at up to 1.4 meter distance for V0 and V1 and up to 1.5m for V2, unless the face rotation in vertical or horizontal directions exceeds 15° . Fig.7 shows the Nexus 7 screenshot with results of such real-time multi-face recognition. Although the recognition accuracy fails with the distance, the versions V0 and V1 provide 70% accuracy at 2.1m distance, while scheme V2 holds this level up to 2.4m. (Note a face area 2.4m distance is only 24×24 pixels).

Figure 7: An illustration of real-time face recognition

Figure 8: The total time of mobile face recognition vs. the offloading methods, the number of faces in image and the image size

B. Processing time

Next, we investigated the impact of computation offloading on processing time by applying all three versions for recognizing same face from 1m distance and profiling the time of task processing for 5 min. The experiment was performed on two image sizes supported by the video camera of Nexus 7: 360×288 and 180×144 (pixels). Table I summarizes the results in terms of the average time of task processing versus the system version (offloading method) and the size of images, captured by the device. The values with a plus sign show the time of data transfers over the network for corresponding task. A quick look at these results reveals that face detection and face identification consume most of computation time. These results have been expected because searching for faces and features in image frame at multiple scales and locations is a computationally expensive, and the face identification algorithms have to store the subspace models and the face database is large in size. Note that increase in database size will linearly increase the CPU and memory requirements. The memory limitation of mobile device, large computation load, and storing the face database on the memory card are the main causes of long processing time for V0.

Another observation is that computation offloading reduces time of mobile face recognition. As one can see both offloading versions have less processing time than V0. The offloading method 2 (version V2) shows better total performance in comparison to the method 1 though it involves many data transfers over the network. It should be noted, that in this result is subject to the network condition. During the tests the network bandwidth was around 50Mbps. In other conditions

Figure 9: Energy consumption of mobile face recognition vs. the offloading methods, the number of faces in image and the image size

(e.g. low bandwidth, intense traffic, signal interference, etc.), transmitting the whole image over the WiFi might delay the process significantly; so the results could be different. Nevertheless, as tests on multi-faced images showed (see Fig.8), the total face recognition time per image for V1 grows proportionally to the number of faces in the image, while for V2 it does not. Therefore we can conclude that the computation offloading method 2 improves the processing time of mobile face recognition on small multi-faced images better than the offloading method 1.

C. Energy consumption

We conducted a set of experiments to measure the total energy consumed from the Nexus 7's battery during face recognition. Each version was examined on two image sizes (360×288 and 180×144 pixels) and three image patterns having 1, 2, and 3 faces in the image, respectively. The power metering architecture consisted of several hardware units: the mobile device (Nexus 7) that ran the face recognition versions; a power meter that measured voltage and current, respectively, drained from the battery terminals during the runs; and a notebook PC that calculated the energy rates and stored the results. In order to avoid bias due to discharge, the battery of Nexus 7 was removed and the battery terminals were directly connected to a DC supply, providing 5V steadily. The power meter was connected to PC via optical cable. The data from the meter was retrieved every second based on the PCLink7 software provided by Sanwa.

Table II summarizes the results in terms of average energy consumed by the face recognition tasks as a function of image size and software implementation. Fig.9 shows dependency of the total energy of mobile face-recognition vs. the offloading methods and the number of faces in an image. Clearly, the energy consumption strongly depends on the offloading method and the image size. The offloading method 2 saves more energy than method 1 due to extensive offloading of computations for both face detection and face identification.

TABLE II. AVERAGE ENERGY CONSUMPTION PER IMAGE (J)

Image size	360×288			180×144		
Software version	V0	V1	V2	V0	V1	V2
Image acquisition	0.025	0.025	0.025	0.025	0.025	0.025
Face detection+ trans.	1.348	1.404	1.352	0.47	0.352	0.243
Face identification	0.863	0.147	0.147	0.56	0.048	0.048
Total	2.236	1.576	1.524	1.05	0.425	0.311

Figure 10: The image transmission vs. the network bandwidth

Figure 11: The total time of mobile face recognition time vs. the offloading methods, internet bandwidth and image size

D. Limitations

Our work has a number of limitations which needed to be kept in mind when using our results.

The biggest one is that the computation offloading is tested on high-speed 802.11n WiFi internet with 56Mbps bandwidth, which is not a typical wireless environment yet. Although it is expected to be widely used in the near future, most wireless networks which are in use nowadays are much slower. If we assume that data packets are sent continuously, a packet is 1KB in size, the propagation delay is 0.4μs (50m distance), the round-trip time (RTT) is 40ms, and an initial handshake of 2×RTT takes place before a transmission, the time of transmitting an entire image over the internet becomes too long (see Fig.10). For instance, transmission of 360x280 bytes over 1Mbps network takes 891ms, while 720x576 bytes require more than 3.5s to be transmitted. As Fig.11 shows, the processing time of version V2 strongly depends on the network's bandwidth and becomes longer than that of V1 as the bandwidth gets smaller than 50Mbps (for 750x576 images) and 10Mbps (for 360x280 images), respectively. To determine the energy-delay tradeoffs at low bandwidth networks, additional studies will be performed.

Furthermore, the results obtained in this work are limited to tablet PC, namely to Nexus 7. Clearly, they might change for other mobile devices and hardware platforms. We are now extending this work, by studying the face recognition offloading methods on low rate WiFi, 3G and 4G networks, as well as on the different hardware platforms, such as smartphones.

Another limitation is that the data base used in our experiments was small, consisting of only 230 static images of 23 persons. Undoubtedly, practical applications require larger data bases. We think however that such limitation does not jeopardize the results. Since both the processing and energy requirements of face detection and face identification grow proportionally with the number of face samples in the database, the trend shown on the small data base will hold for a larger database as well. That is offloading both tasks to the server leads to better system efficiency than offloading only one of them.

Finally, our work explored neither power nor timing optimizations. This also might be considered as a limitation. The figures shown in the work have been obtained by techniques implemented in OS and hardware by default settings. Changing the settings or exploiting additional techniques to improve performance and lower energy consumption (e.g. when idling in the waiting phase) could change the results. This issue also will

be studied in future.

IV. CONCLUSION

In this paper we analyzed the energy and performance gains of two computation offloading methods proposed for mobile face recognition system. Experiments on the Wifi-network based face recognition system implemented on Nexus 7 tablet PC showed that both offloading methods are beneficial for improving system performance and energy consumption. The second method however leads to better gains especially for recognition of multiple faces on small images. For the sake of experiment, we imposed a number of limitations in the current study. Future work will extend the study to overcome the limitations.

REFERENCES

- [1] Zhao, W. , Chellappa, R., Phillips, P. J. and Rosenfeld, A., Face Recognition: A Literature Survey, ACM Computing Survey, December Issue, 399-458, 2003
- [2] X.Tan, S.Chen, Z.-H.Zhou, F.Zhang, "Face recognition from a single image per person: A survey," Pattern Recognition, Vol.39, pp.1725-1745, 2006.
- [3] R.Jafri and H.R. Arabnia, "A Survey of Face Recognition Techniques", J. Inf. Processing Systems, Vol.5, No.2, June 2009, pp.41-67
- [4] Z. Li, C. Wang, and R. Xu, "Computation offloading to save energy on handheld devices: a partition scheme," in Proc. of the 2001 Int. Conf. on Compilers, architecture, and synthesis for embedded systems, ser. CASES '01, NY, USA: ACM, 2001, pp. 238-246.
- [5] U. Kremer, J. Hicks, and J. M. Rehg, "Compiler-directed remote task execution for power management," in Workshop on Compilers and Operating Systems for Low Power, 2000.
- [6] R. K. Balan, M.Satyanarayanan, S.Y.Park, and T.Okoshi, "Tacticsbased remote execution for mobile computing," in Proc. of the 1st Int. Conf. on Mobile Systems, Applications and Services, MobiSys'03. NY, USA: ACM, 2003, pp. 273-286.
- [7] Z.Li, C.Wang, R.Xu, "Computation offloading to save energy on handheld devices: a partition scheme", CASES, pp 238-246, 2001.
- [8] G. Chen,et al., "Studying Energy Trade Offs in Offloading Computation/Compilation in Java-Enabled Mobile Devices", IEEE Trans. Parallel Distributed Systems, vol. 15, no. 9, pp. 795-809, 2004.
- [9] Z. Li, C. Wang, R. Xu, "Task Allocation for Distributed Multimedia Proc. on Wirelessly Networked Handheld Devices", IPDPS, 2002.
- [10] G. H.-Canepa, D. Lee, "An Adaptable Application Offloading Scheme Based on Application Behavior", AINAW, pp.387-392, 2008.
- [11] Xian et al, Adaptive Computation Offloading for Energy Conservation on Battery-Powered Systems, in ICDPS, 2007.

- [12] K.Kumar et al., "A Survey of Computation Offloading for Mobile Systems", *Mobile Networks and Applications*, pp. 1-12, 2012.
- [13] C.K. Ng, M. Savvides, P.K. Khosla, "Real-time face verification system on a cell-phone using advanced correlation filters, 4th IEEE Workshop on Automatic Identification Advanced Technologies, 2005, pp. 57-62.
- [14] C.Schneider, N.Esau, L.Kleinjohann, B.Kleinjohann, "Feature based face localization and recognition on mobile devices", *Int. Conf. Control, Automation, Robotics and Vision*, 2006, pp.1-6.
- [15] Hadid, A.; Heikkila, J.Y.; Silven, O.; Pietikainen, M.; , "Face and Eye Detection for Person Authentication in Mobile Phones," *ACM/IEEE Int. Conf. on Distributed Smart Cameras*, pp.101-108, 2007
- [16] J.Ren, X.Jiang, and J.Yuan. "A complete and fully automated face verification system on mobile devices." *Pattern Recognition*, vol.46 pp.45-56, 2013
- [17] S.Kumar, P.Singh, V.Kumar, "Architecture for mobile based face detection/ recognition", *IJCSE Int. J. on Computer Sc. and Engineering*, Vol. 2, No. 3, 889-894, 2010
- [18] A.Pabbaraju, S.Puchakayala, "Face Recognition in Mobile Devices", *EE & CS*, University of Michigan, 2010.
- [19] H.Yu, *Face Recognition for Mobile Phone Using Eigenfaces*, University of Michigan, 2010.
- [20] K.Imaizumi, V.G.Moshnyaga, "Network-Based System for Face Recognition on Mobile Wireless Devices", *IEEE Int. Conf. Consumer Electronics (ICCE-Berlin)*, 2013, pp.406-409.
- [21] Y-C.Wang, K-T.Cheng, "Energy-optimized mapping of application to smartphone platform – a case study of mobile face recognition", *IEEE Workshop on Embedded Computer Vision*, 2011
- [22] A.Y.Ding, et al, "Enabling Energy-Aware Collaborative Mobile Data Offloading for Smartphones", *10th Annual IEEE Comm. Society Conf. on Sensor, Mesh and Ad Hoc Commun. and Networks (SECON)*, 2013
- [23] A.P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. of the 2nd USENIX conf. on hot topics in cloud computing*, HotCloud'10. Berkeley, CA, USA: 2010, pp. 4-14
- [24] M. Jones, P. Viola, "Face recognition using boosted local features, *Proc. of Int. Conf. on Computer Vision*", 2003 .
- [25] M. Turk and A. Pentland, "Eigenfaces for recognition", *J.Cognitive Neuroscience* 3, 72-86
- [26] K. Etemad, R. Chellappa, *Discriminant Analysis for Recognition of Human Face Images*, *J. the Optical So America A*, vol. 14, No. 8, Aug. 1997, pp. 1724-1733
- [27] *Faces 1999 (Front)*, California Institute of Technology, <http://www.vision.caltech.edu/archive.html>.