Design and Implementation of Real Time Face Recognition System (RTFRS)

Zahraa Qasem Jaber
Computer Engineering Department,
College of Engineering, University of Baghdad
Al-Jadriyah, Baghdad, Iraq

Mohammed Issam Younis, Ph.D
Computer Engineering Department,
College of Engineering, University of Baghdad
Al-Jadriyah, Baghdad, Iraq

ABSTRACT
Face recognition is a pattern recognition technique and one of the most important biometrics; it is used in a broad spectrum of applications. The accuracy is not a major problem that specifies the performance of automatic face recognition system alone, the time factor is also considered a major factor in real time environments. Recent architecture of the computer system can be employed to solve the time problem, this architecture represented by multi-core CPUs and many-core GPUs that provide the possibility to perform various tasks by parallel processing. However, harnessing the current advancements in computer architecture is not without difficulties. Motivated by such challenge, this paper proposes a Real Time Face Recognition System (RTFRS). In doing so, this paper provides the architectural design, detailed design, and four variant implementations of the RTFRS. Finally, this paper determines the speed up obtained for the three advanced implementations (i.e., Hybrid Parallel model, CPU Parallel model, and Hybrid Mono model) against the convention implementation (i.e., CPU Mono model). The practical results demonstrate that the Hybrid Parallel model gain highest speed up around 82X, CPU Parallel model also have a high speed up around 71X, and finally, the Hybrid Mono model gives a slight speed up about 1.04X.

General Terms
Parallel Processing, Parallel Computing, Recognition Algorithm, Multithreading and Concurrent Computing, Heterogeneous Computing

Keywords
SIMT; GPU; Parallel algorithms; Heterogeneous computing; UML

1. INTRODUCTION
Face recognition is one of the most important biometrics methods [1]. Despite the fact that there are more reliable biometric recognition techniques such as fingerprint and iris recognition, these techniques are intrusive and their success depends highly on user cooperation [2]. Therefore, face recognition seems to be the most universal, non-intrusive, and accessible system. It is easy to use, can be used efficiently for mass scanning, which is quite difficult, in case of other biometrics [3]. Also it is natural and socially accepted [4].
Moreover, technologies that require multiple individuals to use the same equipment to capture their biological characteristics probably expose the user to the transmission of germs and impurities from other users. However, face recognition is completely non-intrusive and does not carry any such health dangers [5]. A face recognition system comprises five models as shown in Figure 1.

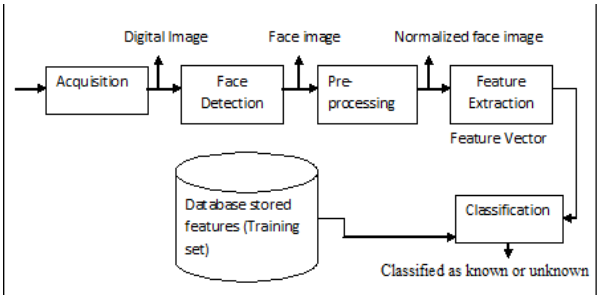

Figure1. Face Recognition System

Acquisition the image of an individual face model is the entry point of the face recognition process. There are two ways to acquire an image: A) Digitally scan an existing photograph, the source is a file; B) Acquire a live picture of a subject [6]. Face Detection Model is the first step in automated face recognition. Its reliability has a significant impact on the performance and usability of the whole face recognition system. Given a single image, the best face detector should be capable of identifying and locating all the present faces regardless of their scale, orientation, age, position, and expression. In addition, the detection should be irrespective of unrelated illumination conditions in the image content. Thus, the face detection can be considered as a task of discriminating between face and non face and segments certain face region from cluttered views [7]. Pre-processing model should be applied before feature extraction [8]. It includes images processing to improve the input image in order to get better quality and therefore making the recognition process with less effort by decreasing time complexity; this can significantly enhance and improve the performance of the overall face recognition system [9]. Feature Extraction for face representation is a central issue in face recognition. Feature extraction algorithms aim at finding effective information that is useful for distinguishing between faces of different persons [7]. In the classification, based on identification concept, face identification system has to determine the identity of the input face image based on comparisons among all templates stored in a database (DB) [10]. Face recognition is a computationally intensive process and needs heavy resources. For many applications, identification process has time constraints to be considered real-time, using sequential execution of such algorithms do not achieve this goal for a large database especially of high resolution images. Today, Current laptops and desktops equipped with recent technologies of multi-core CPUs and many core GPUs that can be employed to speed up the execution time of real time face recognition systems. Graphics Processing Unit (GPU) is powerful co-processors that can offer different advantages to Central Processing Units (CPUs), and the modern GPU provides hundreds of streaming

cores and handles thousands of threads, which makes it specifically suitable for compute intensive applications. Multi-core processors integrate multiple execution cores on a single processor chip. The operating system treats each execution core as an independent logical processor with separate execution resources like functional units or execution pipelines; therefore, the operating system can assign different application programs to the different cores to get a parallel execution. By using techniques of parallel programming, it is possible to execute a computation intensive application program in parallel on a set of cores, results in reducing the execution time compared to the execution time on a single-core. Recent technologies make it possible to use GPU with CPU creating what's known as a "Hybrid (Heterogeneous) System's". The goal of using GPU with CPU in this paper is to release some CPU overhead by sharing GPU in the process of face detection. The difference in execution time to run any process on CPU or GPU will be clear when CPU is engaged by heavy duties.

Performance is the most factor effect to apply whether or not a face recognition algorithm could be used. This paper proposes a Real Time Face Recognition System (RTFRS). RTFRS will be tested by taking into consideration the performance as far as the speed up is concerned. In doing so, this paper introduces: the architecture design, planning the implementation by selecting suitable algorithms and tools, the detailed design and implementation of the RTFRS, and finally, makes a fair comparison of variant implementations of the RTFRS.

2. FACE IMAGE DETECTION MODEL

Face detection is the elementary step in the face recognition system and acts as a stone to all facial analysis algorithms. Many algorithms exist to implement face detection; each has its own weaknesses and strengths. The majority of these algorithms suffer from the same difficulty; they are computationally expensive. The image is a combination of color or light intensity values. Analyzing these pixels for face detection is time consuming and hard to implement because of the enormous diversity of shape and pigmentation in the human face. Viola and Jones proposed an algorithm, called Haar-cascade Detector or called Viola-Jones, to quickly detect any object, including human faces, using AdaBoost classifier cascades that are based on Haar-like features and not pixels [11]. Viola-Jones algorithm is widely used in various studies involving face processing because of its real-time capability, high accuracy, and availability as open-source software under the Open Computer Vision Library (OpenCV) [8]. Viola-Jones detectors can be trained to recognize any kind of a solid object, including human faces and facial features such as eyes, and mouths. OpenCV has implemented Viola-Jones and provides a pre-trained Haar-cascade for face detection [12].

3. FISHERFACE / LDA

Fisherface is also known as Linear Discriminant Analysis (LDA). It is more suited for finding projections that best discriminate different classes. It does this by looking for the optimal projection vectors which maximize the ratio of the between-class scatter and the within-class scatter separation, same classes should cluster strongly together, whereas different classes are as far away as possible from each other [13], [14]. One problem for LDA is that the within-class scatter matrix is almost always singular, which is a small sample size problem. The reason is that the number of all the pixels in each sample images is usually larger than the number of training sample images [15]. The singularity problem can raise detection error rate if there is a large difference in pose or lighting condition in the same face images. The Fisherface approach can take full account of within-class information; minimizing difference within each class and then the problem with variations in the same images such as different lighting conditions can be overcome [13]. The Fisherface algorithm is as follows [14]:

Let X be a random vector with samples drawn from c classes:

$X = \{X_1, X_2, \ldots\ldots\ldots, X_c\}$

$X_i = \{X_1, X_2, \ldots\ldots\ldots, X_n\}$

The scatter matrices S_B and S_w are calculated as:

$$S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

$$S_w = \sum_{i=1}^{c} \sum_{X_j \in X_i} (x_j - \mu_i)(x_j - \mu_i)^T$$

, where μ is the total mean:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

And μ_i is the mean of class $i \in \{1, \ldots., c\}$:

$$\mu_i = \frac{1}{|X_i|} \sum_{X_j \in X_i} X_j$$

Fisher's classic algorithm now looks for a projection w, which maximizes the class reparability criterion:

$$W_{opt} = \arg max_W \frac{|W^T S_B W|}{|W^T S_W W|}$$

A solution for this optimization problem is given by solving the General Eigenvalue Problem:

$$S_B v_i = \lambda_i S_w v_i$$

$$S_W^{-1} S_B v_i = \lambda_i v_i$$

There's one problem left to solve: The rank of is at most (N-c), with N samples and c classes. In pattern recognition problems, the number of samples N is almost always smaller than the dimension of the input data (the number of pixels). As so, the scatter matrix becomes singular. In order to avoid this problem, a PCA (Principal Component Analysis) can be applied on the data and projecting the samples into the (N-c) dimensional space first. Next, LDA is performed on the reduced data. As a result, the scatter matrix is non-singular anymore. The optimization problem can then be defined as:

$$W_{pca} = \arg max_W |W^T S_T W|$$

$$W_{fld} = \arg max_W \frac{|W^T W_{pca}^T S_B W_{pca} W|}{|W^T W_{pca}^T S_w W_{pca} W|}$$

The transformation matrix W that projects a sample into the(c-1) dimensional space is then given by:

$W = W_{fld}^T W_{pca}^T$

4. THE ARCHITECTURAL DESIGN OF RTFRS

This section describes the architectural design of RTFRS, involve: the Mono (sequential) and Parallel face recognition concepts.

The face recognition is the hardest algorithm because it has many steps before it start the real recognition. A face must be detected to increase the possibility of recognition and speed up the process by choosing one location in the image. To

detect a face, two steps must be done before the recognition. The first step is to resize the image to standard size (determine by the administrator), apply some filter to increase the quality, and convert the image into a compatible form. Next, go to detection face, such that the image required to recognize is uploaded in the memory with an *Extensible Markup Language* **(**XML**)** file to detect a face, and finally, go to recognition step. In recognition, the extracted face will be compared with training faces when they uploading to memory and extract face features by a recognition algorithm.

Any operating system (OS) has multiple ways to deal with a process for different structures. Some process has a single thread and other has multithreads architecture (threads can run in a simultaneous manner). In this paper, mono is a description for a single core CPU and parallel is a description for a multi-core CPU (i.e., the suggested names are taken with respect to CPU). In addition, the hybrid word is used whenever the computation involves the GPU, as well as, the CPU in a heterogeneous manner. Based on the previously mentioned conventions, four variants of (RTFRS) will be implemented depending on the hardware resources that employed in the computer system architecture. The first one is **CPU Mono** (single core CPU), the second is **CPU Parallel** (multi-core CPU), the third is **Hybrid Mono** (single core CPU with GPU), and finally **Hybrid Parallel (**multi-core CPU with GPU).

4.1 CPU Mono Face Recognition

In mono face recognition, both detection and recognition phases are running on a single core CPU and all face recognition phases run sequentially (i.e., step by step), because the hardware resources are limited and this leads to very slow face recognition system, as shown in Figure 2.

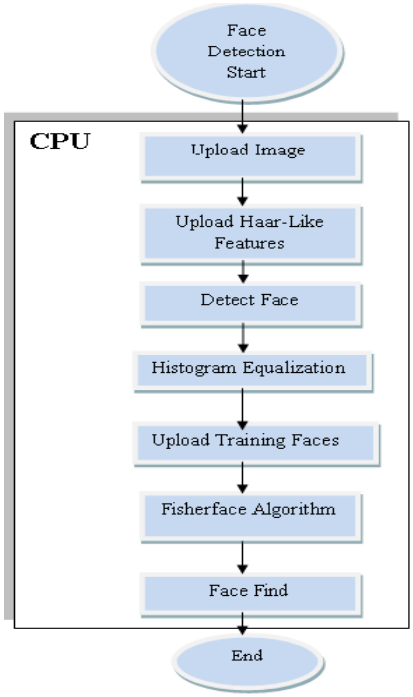

Figure 2. CPU Mono Face Recognition

4.2 CPU Parallel Face Recognition

In the parallel face recognition process, two tasks can be done simultaneously. The process of uploading training face images in the memory and the process of getting face features from the training face images. The multithreading capability can be applied on multi-core CPU to perform the recognition process, as shown in Figure 3.

4.3 Hybrid Mono Face Recognition

Using GPU increases the speed to find a face by using face recognition algorithm. As so, in hybrid solution the image is sent to GPU for detecting the face. Next, the result is sent back from GPU to the CPU in order to complete the process, as shown in Figure 4.

4.4 Hybrid Parallel Face Recognition

This variant uses maximum allowable hardware resources represented by GPU and multi-core CPU; this is done by running the face detection process on many-core GPU and recognition process on multi-core CPU. It is also employing concurrent execution of both the process of uploading training face images in the memory, and the process of getting face features from the training face images by exploiting the multithreading capability on the multi-core CPU system as shown in Figure 5.

5. PLANNING THE IMPLEMENTATION OF THE RTFRS

This section gives the specification of the building blocks of the RTFRS includes the selection of algorithms, tools, and finally, the data flow in RTFRS. Viola-Jones Face Detector is selected to detect faces. While, Fisherface algorithm is selected as the recognizer algorithm. The four variants use the same selected algorithms for detection and recognition phases. The construction of the RTFRS consists of the following basic building blocks that facilitate the developing purpose.

- Microsoft .Net framework 4.
- C# programming language.
- Open Computer Vision (OpenCV) version 2.4.8.
- NVIDIA GPU and CUDA.
- EmguCV version windows universal CUDA 2.9.0.1922.
- Multi-core CPU.

The dataflow among the selected tools is illustrated in Figure 6.

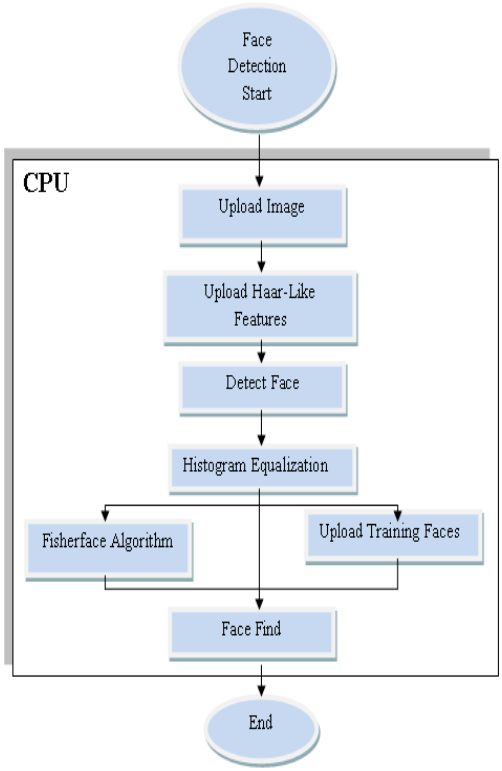

Figure 3. CPU Parallel Face Recognition

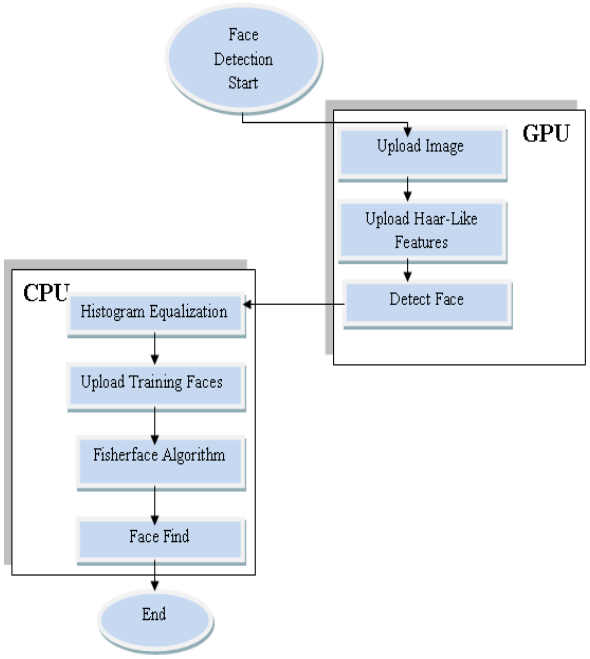

Figure 4. Hybrid Mono Face Recognition

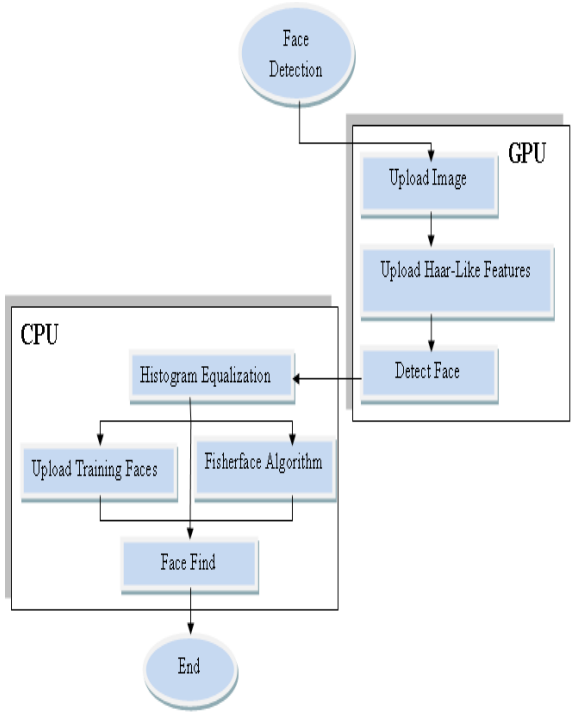

Figure 5. Hybrid Parallel Face Recognition

6. DETAILED DESIGN AND IMPLEMENTATION OF THE RTFRS

This section describes the detailed design in terms of flow chart and activity diagrams for the proposed RTFRS. Moreover, the implementation details are explained in terms of pseudo code of the RTFRS.

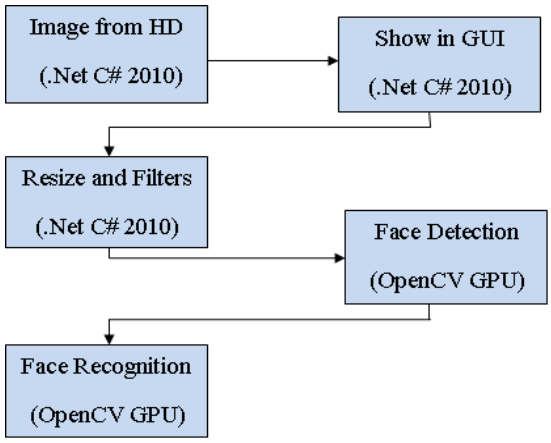

Figure 6. Dataflow in the RTFRS

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language (UML), activity diagrams are intended to model both computational and organizational processes (i.e. workflows). Activity diagrams show the overall flow of control. In RTFRS, the main activity is the training phase and the other one is the recognition phase. The activity diagram for the training phase is shown in Figure 7, and the corresponding steps' activities are tabulated in Table 1. Similarly, the activity diagram for the recognition phase is shown in Figure 8, and

the corresponding steps' activities are tabulated in Table 2. Also, the pseudo code for the training and the recognition phases are shown in Figure 9 and Figure 10 respectively.

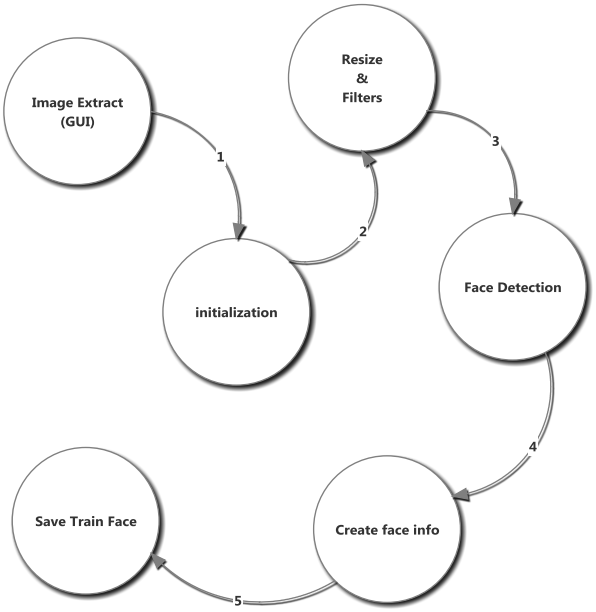

Figure 7. Activity Diagram of Training Phase

Table 1. Activity Table of Training Phase in RTFRS

Step Number	Activities
1	Sends Image from folder to start the module work.
2	Image resizing in to a fixed size; determined by the administrator; then applying some image processing filters to increase image quality.
3	New Image applies Haar-cascades algorithm to detect the face and remove other parts of image.
4	Clone the image and insert the subject name.
5	Create file in a hard disk to new image and save the image path and image name in XML DB.

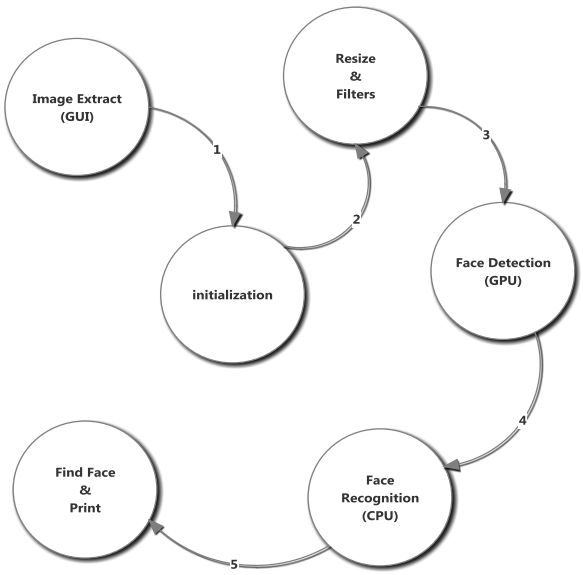

Figure 8. Activity Diagram of Recognition Phase (Hybrid Parallel Implementation)

Table 2. Activity Table of Recognition Phase in RTFRS

Step Number	Activities
1	Sends Image from folder to start the module work.
2	Image resizing in to a fixed size; determined by the administrator; then applying some image processing filters to increase image quality.
3	New Image goes to GPU to apply Haar-cascades algorithm to detect the face and remove other parts of image, then the result is returned to CPU.
4	Load train images from DB to RAM and extract features from them in parallel. Next, the CPU extracts the face features from new face and compares with other face features of the training images.
5	If find a face from DB closes to the detected face the system recognized the person by displaying the name in the GUI; otherwise; the system displays a message for an unknown person.

```
1. Initialization
      Select how to get image
         open file dialog to select image
         Create Bitmap → P
         Put new image in P
         Send P , image-viewer width , height and
         quality=72
          to Resize new image and set its quality 72
         Calculate time
         Add log →DB
2. CPU part (Face detection)
         Load Haar cascades XML and objects → H
         H→detect(P)
         Create new CPU bitmap P_CPU
         Send H.result→ P_CPU
         Calculate time
         Add log →DB
3. CPU part (Face saving)
         If (P_CPU not = empty) then
            P_gray→ P_CPU. grayscale
            Take face name from GUI P_name
            Create image file P_file
         P_file→P_gray
            P_file.name = P_name + random
            Numebr
            Save P_file in Hard Disk
         Calculate time
Add log →DB
```

Figure 9. The Pseudo Code for Training Phase

```
1.  Initialization
      Select how to get image
      open file dialog to select image
      Create Bitmap → P
      Put new image in P
      Send P , image-viewer width , height and
      quality=72  to Resize new image and set its
      quality 72
      Calculate time
      Add log →DB
2.  GPU part (Face detection)
    Open new bitmap in GPU memory P-GPU
      Send P → P-GPU
     Load Haar cascades XML and objects → H
      H→detect(P-GPU)
      Create new CPU bitmap P_CPU
      Send H.result→ P_CPU
      Calculate time
      Add log →DB
3.  CPU part (Face recognition)
    Create object from faceReco class P_Reco
      If (P_CPU not = empty) then
         Open Parallel Case
         { P_Reco load All Training Faces
         P_Reco Extract face features for All
         Training Faces}
         P_Reco find P_CPU
          Extract P_CPU features
          Compare all features faces with P_CPU
         features
            If find = true
            Print P_CPU.name on screen
            Else Print "Unknown" on screen
      Calculate time
Add log →DB
```

Figure 10. The Pseudo Code for the Recognition Phase

RTFRS has a main form named "master form" which is the basic GUI, and it contains the control running to the basic modules that perform the face recognition process. The master form has one sub form for training, and four other sub forms for recognition. By drop down menu, the sub form can be selected to be opened. The CPU Mono Training Form enables the user to select the source of the image (i.e., the image comes from a file). After the system detects a face, the user can write the person's name in the text filed, click save button to write result log in the database, save the face image and the name in image's DB, this form is running in a single core CPU. Unlike the training phase, which is implemented in CPU Mono model, the recognition phase has four separate models. These four variant models are CPU Mono, CPU Parallel, Hybrid Mono, and Hybrid Parallel. In a CPU Mono Recognition, the master form kept the same, an image from a folder in hard disk can be selected, this image will display in the image box, then the system will detect the face and search if this face is predefined in database to get face name in the text filed and then click save button to write the result log in database.

Time will be measured in each step in training and recognition models and also the overall time (for total steps) will be measured and saved in the DB in order to measure the performance factor in later procedures.

7. THE RESULTS AND EVALUATION

Speed up is a measure that captures the comparative gain of solving the same computational problem in parallel. The speed up of parallel computation operating on p processors is derived as:

$$S_p = \frac{Ts}{Tp}$$

Where T_s is the execution time taken to perform the computation on a uni-processor system and T_p is the execution time on a multiprocessor system with scale p when solving the same computation task [16]. Because all the measured time has been saved in the DB, the DB can be accessed in order to view all the timing values gained from all the four implementation variants, there are forms can be displayed to show all logs of timing measures. Experimentation has been made to evaluate the RTFRS in order to judge which one of the implementation alternatives reduces the processing time significantly. The experiment consists of applying 400 images for 40 persons' faces (10 images per person), defining, training, and recognizing these pictures on the four implementation variants (i.e., CPU Mono, CPU Parallel, Hybrid Mono and Hybrid Parallel). The experiment is taken place on the same environment; in order to get a fair comparison for variant implementations. It should be mentioned that the speed up is measured using speed up equation mentioned above. In addition, the results of the four algorithm variants are applied for the same number of persons' images.

Execution Time (measured in ms) to recognize variable number of images running on variant implementations is tabulated in Table.3. According to Table.3, the shaded field represents the fastest variant (i.e., the Hybrid Parallel), using the same framework on 400 face images, therefore, employing parallel processing provides better speed up.

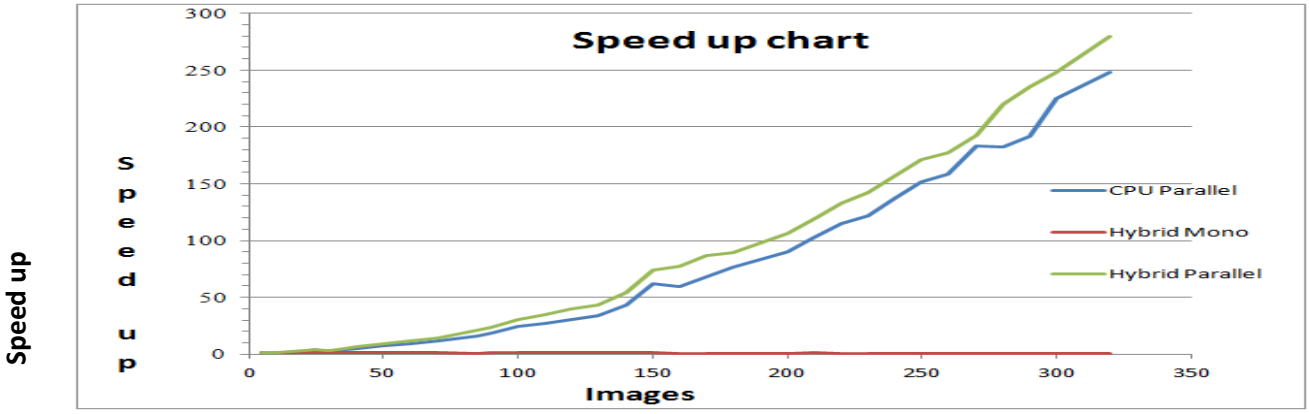

Figure 11. Speed up Gains with respect to the CPU Mono Implementation

ıll the four variant implementations run on the same conditions, and the CPU Mono Recognition is the slowest than all other three variants, therefore, the speed up is measured of the three other variants with respect to the CPU Mono implementation using the same speed up equation.
Figure 11 represents the speed up gains with respect to the CPU Mono Implementation.

The Hybrid Parallel Recognition is the fastest algorithm variant among the all, because it provides an overall (average) speed up around **(82)** times. The CPU Parallel provides an overall speed up around **(71)**. Finally, the Hybrid Mono provides a little improvement about **(1.04)**.

Table 3. The Execution Time (ms) to Recognize Variable Number of Images Running on Variant Implementations

No. of images	CPU Mono	CPU Parallel	Hybrid Mono	Hybrid Parallel (ms)
5	102	86	82	66
10	123	87	104	67
15	160	92	138	68
20	206	93	182	69
25	279	68	221	69
30	305	177	288	94
35	368	87	357	76
40	448	87	427	69
50	612	87	594	70
60	823	90	796	71
70	1027	88	937	72
85	1452	92	1440	70
90	1700	92	1562	71
100	2110	85	1977	69
110	2440	90	2307	70
120	2648	87	2575	67
130	3108	91	2981	72
140	3840	88	3782	71
150	5117	83	4908	69
160	5560	93	5586	72
170	6357	93	6337	73
180	6899	90	6930	77
190	7705	89	9768	97
200	8401	93	8456	79
210	9394	91	9233	79
220	10238	89	10151	77
230	11106	91	11140	78
240	12102	88	12095	77
250	13340	88	13819	78
260	13647	86	14083	77
270	15054	82	15376	78
280	17362	95	17589	79
290	18610	97	18800	79
300	19845	88	21373	80
310	21289	90	21814	95
320	22371	90	22901	80

8. CONCLUSIONS

This paper proposed a real time face recognition system (RTFRS). RTFRS has been implemented in four implementation variants (i.e., CPU Mono, CPU Parallel, Hybrid Mono and Hybrid Parallel. Fisherface algorithm is employed to implement recognition phase and Haar-cascade algorithm is employed for the detection phase. In addition, these implementations are based on industrial standard tools involve Open Computer Vision (OpenCV) version 2.4.8, Microsoft .Net framework 4, C# programming language, EmguCV version windows universal CUDA 2.9.0.1922, and heterogeneous processing units. The experiment consists of applying 400 images for 40 persons' faces (10 images per person), defining, training, and recognizing these images on these four variants, the experiment is taken place on the same environment (laptop computer Intel core i7 processor 2.2 GHz, Nvidia GPU GeForce GT 630M, 7GB RAM). The speed up factor is measured with respect to the CPU Mono implementation (the slowest than all other three variants). The practical results demonstrated that, the Hybrid Parallel Recognition is the fastest algorithm variant among the all, because it gives an overall speed up around **(82)** times. The CPU Parallel gives an overall speed up around **(71)**. Finally, the Hybrid Mono gives a little improvement about **(1.04)**. Thus, employing parallel processing on modern computer architecture can accelerate face recognition system.

9. ACKNOWLEDGMENTS

The authors desire to express their gratitude and thanks to the computer center at the University of Baghdad for their support to this work, and offer thanks and appreciation for everyone who contributes in doing this work.

10. REFERENCES

[1] See, J.; Eswaran, C. and Fauzi, M. F. A. "Video-Based Face Recognition Using Spatio-Temporal Representations", in Reviews, Refinements and New Ideas in Face Recognition, Corcoran P. ,Ed., InTech, Croatia, pp. 273-293, 2011.

[2] Rady H. "Face Recognition using Principle Component Analysis with Different Distance Classifiers", International Journal of Computer Science and Network Security, Vol. 11 No. 10, pp. 134-143, October 2011.

[3] Patel R.; Rathod N. and Shah A. "Comparative Analysis of Face Recognition Approaches: A Survey", International Journal of Computer Applications, Vol. 57, No. 17, pp.50-61, November 2012.

[4] Xie, S. J.; Yang J.; Park, D. S. ; Yoon, S. and Shin, J. "State of the art in biometrics" in Iris Biometric Cryptosystems, Yang, J. and Nanni, L., Eds., InTech, , Croatia, pp. 179-202, July 2011.

[5] Jafri R. and Arabnia, H. "A Survey of Face Recognition Techniques", Journal of Information Processing Systems, Vol. 5, No. 2, pp. 41-68, June 2009.

[6] Bhatia R. "Biometrics and Face Recognition Techniques", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, No. 5, pp. 93-99, May 2013.

[7] Li S. and Jain A. "Handbook of Face Recognition", 2nd edition, Springer, 2011.

[8] Jain A.; Ross A. and Nandakumar K. "Introduction to Biometrics: A Textbook", Springer, 2011.

[9] Krishna B.; Bindu V.; Durga K. and AshokKumar G. "An Efficient Face Recognition System by Declining Rejection Rate using PCA", International Journal of Engineering Science & Advanced Technology, Vol. 2, No. 1, pp. 93 – 98, February 2012.

[10] Lih-Heng C.; Sh-Hussain S. and Chee-Ming T. "Face Biometrics Based on Principal Component Analysis and Linear Discriminant Analysis", Journal of Computer Science, Vol. 6, No. 7, pp. 693-699, 2010.

[11] Wilson P. and Fernandez J. "Facial Feature Detection using Haar Classifiers", The Journal of Computing Sciences in Colleges, Vol. 21, No. 4, pp. 127-133, April 2006.

[12] Runarsson K." A Face Recognition Plug-in for the PhotoCube Browser", M.Sc. thesis, Reykjavik University, December 2011.

[13] Bedre J. S. and Sapkal S. "Comparative Study of Face Recognition Techniques: A Review", International Journal of Computer Applications, Vol. 1, No. 1, pp. 12-15, 2012.

[14] Philipp Wagner, "Face Recognition with Python", available at: http://www.byte_sh.de, last accessed 20 April 2014.

[15] Zhao Q.; Liang B. and Duan F. "Combination of Improved PCA and LDA for Video-Based Face Recognition", Journal of Computational Information Systems, Vol. 9, No. 1, pp. 273-280, 2013.

[16] Xiong H.; Zeng G.; Zeng Y.; Wang W. and Wu C. "A Novel Scalability Metric about Iso-Area of Performance for Parallel Computing", The Journal of Supercomputing, Springer, December 2013.