CUDA-based Real-time Face Recognition System

Ren Meng [#1], Zhang Shengbing [*2], Lei Yi [#3], Zhang Meng [*4]

[#] *School of Computer Science and Engineering, Northwestern Polytechnical University*
Xi'an China 710072

[1]renmeng433@sina.com

[*] *School of Computer Science and Engineering, Northwestern Polytechnical University*
Xi'an China 710072

[2]Zhangsb@nwpu.edu.cn

Abstract—**This paper proposes a real-time face recognition system based on the Compute Unified Device Architecture (CUDA) platform, which effectively completed the face detection and recognition tasks. In the face detection phase with Viola-Jones cascade classifier, we implemented and improved novel parallel methodologies of image integral, calculation scan window processing and the amplification and correction of classifiers. In the face recognition phase, we explored the parallelizing of the algorithm and parallelized some part of the testing phase. Through the optimization of the two important part of face recognition system, the system we proposed make a big difference. The experimental results demonstrate that, in comparison with traditional CPU program, the proposed approach running on an NVidia GTX 570 graphics card could respectively achieve 22.42 times speedup in detection phase and 1668.56 times speedup in recognition phase when only training 2000 images and testing 40 images compared with the CPU program running on an Intel core i7 processor. The recognition speed will increase until it reaches the hardware resource limit. It shows that the system we proposed achieves a good real-time performance.**

***Keywords*: face recognition, face detection, CUDA, real-time**

I. INTRODUCTION

Face recognition is the most natural mean of recognition by humans. In recent years, with the rapid development of computer technology, face recognition system is widely used in research and development. Face recognition technology has become one of the most popular researches in the field of pattern recognition and image processing. Fig 1 shows the system. Face detection is the first step of face recognition, which could help us to make sure the face location and preprocess the face image. The fast real-time object detection method – Viola-Jones cascade classifier [1] was proposed by Viola and Jones from the University of Cambridge in 2001. They use integral image, AdaBoost [2] algorithm and the classifier cascade to accelerate the detection speed on the CPU. After detecting the face region from the testing images, we move on to the next step: recognize the face. A well-known face recognition algorithm, Principal Component Analysis (PCA) with Eigen-faces [3,4], has long been proposed. However, with the continuous improvement of image resolution and real-time requirements, the CPU platform is not competent.

The Graphics Processing Units (GPU) is being widely used in computing-intensive applications. The computational power of the generation of GPUs is several times high than that of a CPU. Considering the characteristics of data parallelism inherent in face recognition, it is a good method to use parallel computing to speed up the recognition process. CUDA (Compute Unified Device Architecture), which was developed by NVidia, is a parallel computing platform to GPGPU programming and allows hundreds of concurrent threads to run at the same time [5].

Based on CUDA platform, we proposed a real-time face recognition system, which focused on accelerating the recognition process by parallel computing. An extremely fast face recognition system will have broad practical application. Although, the recognition system constructed by Ping Zhang [6] achieved a high frontal face detection rate, the research just concentrated on the detection rate and didn't consider the efficiency of the whole system. Paper [7, 8] did some research on parallel acceleration of face recognition, in comparison, they both focused on the correction rate and didn't achieve a good real-time performance, and our system significantly improved the recognition speed and the increase in speed will enable real-time face recognition.

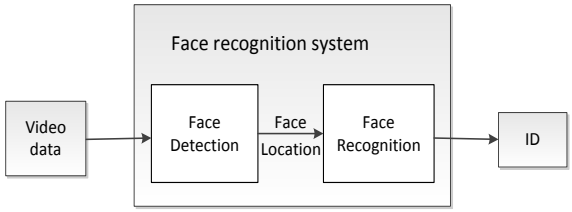

Fig 1. Face recognition system

II. FACE RECOGNITION ALGORITHM

The real-time face recognition system gets the testing images from the camera firstly, and then determines the face's information or the face ID. This process consists of two key steps: face detection and face recognition. Only those images which really contain a face should be recognized. Therefore, the system we proposed includes face detection and face recognition. We get the interested face region by face detection and identify the face ID by face recognition.

A. Face Detection Algorithm

The Viola and Jones, the University of Cambridge's students, proposed the first real-time face detection method, Cascade classifier method. Their main contribution is the

three key concepts: the integral image, AdaBoost algorithm and cascade of classifiers.

Cascade classifier method is a face detection algorithm based on Haar-like features. The white region's pixel value sum subtracts the black region's pixel value sum can get the so called characteristic value of feature rectangle, which is used as the face detection's basis. This paper used five kinds of Haar-like feature rectangle shown in Fig.2.

Fig 2. Haar-like feature rectangle

From Integral image is used to calculation the characteristic value of feature rectangles [1]. The integral image can be computed from an image using a few operations per pixel. Every calculation could be simplified as a fixed times of addition or subtraction operations. Equation(1) and Equation(2) shows how to get the point (x, y)'s integral $ii(x,y)$ and the rectangular region's pixel value sum S. i (x', y') is the point (x', y')'s pixel value of image, (x_A, y_A), (x_B, y_B), (x_C, y_C) and (x_D, y_D) are the rectangular region's top left, top right, bottom left and bottom right corner's coordinate respectively.

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x',y') \quad (1)$$

$$S = ii(x_D,y_D) - ii(x_C,y_C) - ii(x_B,y_B) + ii(x_A,y_A) \quad (2)$$

By learning from the given training set and feature rectangle set, we can generate a classifier function f=(X, Θ), where X is the feature rectangle, Θ is the threshold value. The return value of function f is determined by the relationship of X and Θ. The AdaBoost algorithm is used to select the feature rectangles which cover facial characteristics. Those feature rectangles make a weak classifier and a group of weak classifiers creates a strong classifier. After concatenating the strong classifier, a cascade classifier is finished and it can judge whether the scan window contain facial information.

B. Face Recognition Algorithm

PCA in face recognition aims to reduce the dimensionality of data and extract the features required for comparison of faces. After detection, the images which contain faces are resized (N pixel × L pixel) and preprocessed. The training images are grouped into different classes and each class contains multiple images of a single person with different facial expressions. Every image is represented as 1-D vector Γ_i (NL × 1). Supposed that the database has M training images and compute the average face vector Ψ with (3).The Γ_i subtracts the mean face Ψ could get Φ_i.Using (4) to compute the covariance matrix C.

$$\Psi = \frac{1}{M}\sum_{i=1}^{M} \Gamma_i \quad (3)$$

$$C = \frac{1}{M}\sum_{n=1}^{M} \Phi_n \Phi_n^T = AA^T \quad (4)$$

Due to the eigenvectors μ_i of AA^T (where A equals $[\Phi_1 \Phi_2 \cdots \Phi_M]$, which is a NL × M matrix) is very large and it has the same eigenvalues withA^TA, we can compute the A^TA's eigenvectors ν_i.The ν_i multiplied by A is the μ_i. Each face Φ_i in the training set can be represented in the linear combination of the best K eigenvectors, which is equal to M-1 in this paper. These eigenvectors are known as eigenfaces. The eigenvectors of all images are computed similarly and placed side by side to make up the eigenspace W. Each normalized training face $\Phi_i$is represented in this basis by a vector as (5).

$$\Omega_i = W^T\Phi_i = \begin{bmatrix} w_1^i \\ w_2^i \\ \cdots \\ w_k^i \end{bmatrix}, i = 1,2,\ldots,M \quad (5)$$

Then, normalize the test images and project these images on the eigenspace W. In order to identify the image, we use Mahalanobis distance (Equation 6) to determine which class the test face falls. The Mahalanobis distance between the projected test images and each of the projected training images are computed. If the minimum distance falls under a predefined threshold θ, the nearest distance corresponding to the class is the recognition results.

$$\varepsilon_i = \sqrt{\left(\frac{w_1^i - w_1}{\mu_1}\right)^2 + \left(\frac{w_2^i - w_2}{\mu_2}\right)^2 + \cdots + \left(\frac{w_k^i - w_k}{\mu_k}\right)^2} \quad (6)$$

III. FACE RECOGNITION SYSTEM BASED ON CUDA

First of all, the testing images captured by a camera and preprocessed to be a fixed size, which only contains the face region. Then, with the recognition algorithm we can easily identify the face. However due to the large amount of data, the process will be very time-consuming. We implemented and improved novel parallel methodologies of detection part and recognition part based on CUDA.

A. Parallel Detection

After loading classifier and images, we use GPU to calculate the image integral [9, 10]. Based on CUDA platform, the row integral and column calculation are realized respectively by a kernel according to priority (first row, then column), which can effectively avoid the data dependency of image integral calculation. The image width determined the number of threads. However, the CUDA threads are scheduled by warp. Each warp only has 32 threads and the same warp of row integral calculation need to access different column image data. Since the column image data is not continuous, there will be extra communication overhead. When it comes to the writing back operation of row integral calculation's results, the same problem appears. The system's performance is greatly affected.

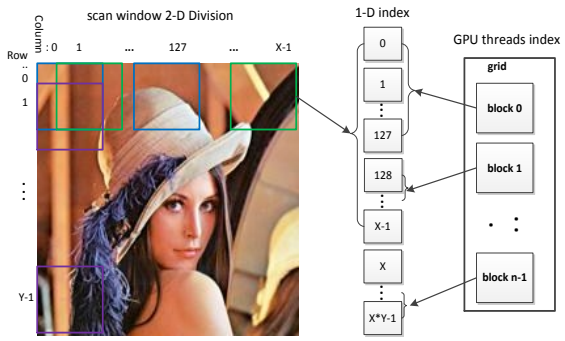

Fig 3. The parallel organization of the scan window

We optimized the memory access by using shared memory which is defined in CUDA [11]. Assuming that M is image's width, N is the number of threads within a block, the size of shared memory is N×N. The threads in a block load image data into shared memory by row access and do integral calculation. After all computations finished, write the results which is stored in shared memory back to the global memory in GPU. The memory access time is reduced and the algorithm is more suitable for the CUDA platform.

Then we use scan window to detect the image block. Since each window is independent, this part can be accelerated by kernel. Fig.3 shows the parallel organization of the scan window. However, the thread is scheduled by warp and the execution units on the GPU are SIMD architecture. When a thread1 just pass through the first strong classifier, while the another thread which belongs to the same warp with the thread1 pass through all the strong classifiers, this situation will leads to a unbalance load problem.

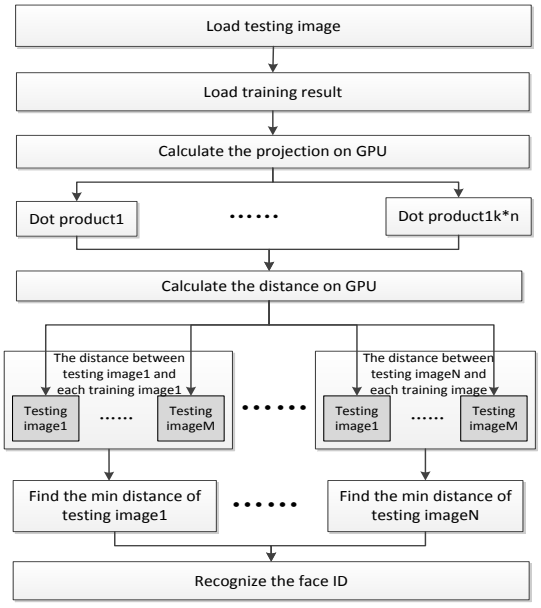

Fig 4. The parallel recognition flow chart

To solve the unbalance load problem and to make sure that a large number of threads execute concurrently on GPU, we reduced the thread parallel granularity to strong classifier and remapped the thread with the window when one stage of classifier is finished. Although this method increases the additional overhead of data communication between the host CPU and the device GPU, we use zero-copy technology [12], which means that the block of page-locked host memory can also be mapped into the address space of the device side, to eliminate this overhead.

Because the face size is not fixed, the scan window's size should be amplified proportionally after once detection, which is actually the correction of the feature rectangle in classifier.

Since the operation to the feature rectangle is independent, one thread can be in charge of a feature rectangle. Many threads execute concurrently can speed up the process and the results can be stored in the global memory to eliminate the classifier's transmission time between GPU and CPU.

Last, merge the face window and save it as a fixed format. So far, the face detection part is over and we already get the face position which will be used in face recognition part.

B. Parallel recognition

When the database is unchanged, there is no need to do the training phase in recognition part. Since the most time consuming is testing phase, we only discuss the acceleration of testing phase.Fig.4 shows the parallel recognition flow chart.

After loading the testing images and the training results, since there is no correlation between each testing image, we compute the projection in parallel. Shown as Fig.5, many threads execute the operation at the same time. Each element of the vector, which is calculated in parallel, is calculated in parallel. The number of threads in a block (K) means the dimension of a feature vector. In this way, we achieved a double parallel between images and in images and make the system more speedy.

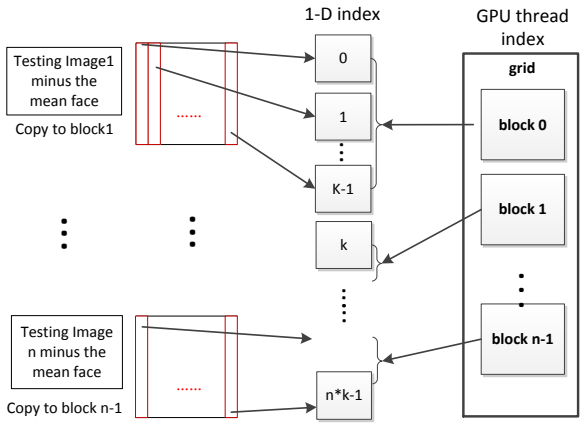

Fig 5. Project the testing image to the eigenspace

Finally, we recognize face by carrying out the minimum result between the computations explained in equation (6). However, when the number of training faces M or testing face N is very large, this operation can be very time consuming. Taking into account the real-time problem, we put this operation in parallel. Each block is responsible for a testing

image and every thread in the block is in charge of a distance between this testing image and a certain training image. All the results calculated are stored in the global memory of GPU. Then, we use reduction algorithm to get the minimum number. If the minimum distance falls under a predefined thresholdθ, the minimum number is the testing face's ID.

C. Database

We choose one of the most common databases, called the ORL Face Database [13], to achieve our implementation. Known as the AT&T Database, There are ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement). The database files are in PGM format and the size of each image is 92 × 112 pixels, with 256 grey levels per pixel. Considering the GPU's computing ability and flexibility of the program, we also create our own database in accordance with the ORL database's specification. This allowed us to further measure GPU performance and to assess the correct recognition rate.

IV. Experiments and Results

To test the performance of CPU and GPU, host CPU used in the experiments is Core i7 9300 quad-core processor and the GPU is GeForce GTX570, compute capability 2.0. The operating system is Windows 7 64-bit Ultimate edition. The ORL Database is selected as the training set which contains 400 images. In order to truly show the power of a GPU, we added another 160 people's face images with the same bmp format of ORL database to make the database as large as 2000 images.

For the detection phase, the image's size contributes to the parallel particle and the speedup. For the recognition phase, the factors that affect the speedup are the testing images' number and the training images' number.

We test the detection phase with different size images and the execution times are shown in Fig.6.The correct detection rate of CUDA is shown in table1, which is similar to CPU. Fig6 shows clearly that the time CUDA use is shorter than CPU use for the same image. That means CUDA is more suitable for real-time face detection, which is the first step of real-time face recognition.

First, we recognize 10 images with different scale of the database. Fig.7 shows the execution time of CPU and GPU (training time not included). When the size of the database increasing, the time of CPU is increasing rapidly while GPU time is very gentle. It shows that it's better to use CUDA for face recognition when the database is large.

The number of threads within each block is equal to the number of training images. When we only have 10 testing images, we just use 10 block and each block has the number of training images' threads. While the experimental platform using GTX 570 have 15 Multiprocessor, each one can support 1536 threads, the threads on the each Multiprocessor have not fully utilized and there also have idle Multiprocessor. So, with the increase of database images, the speedup is terribly increasing.

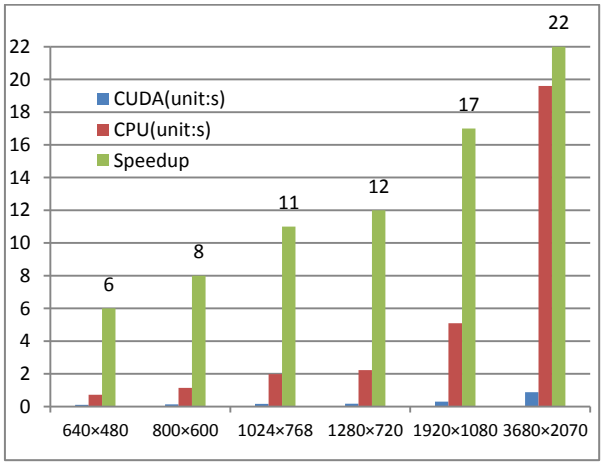

Fig 6. The detection time of different sizes images and the speedup

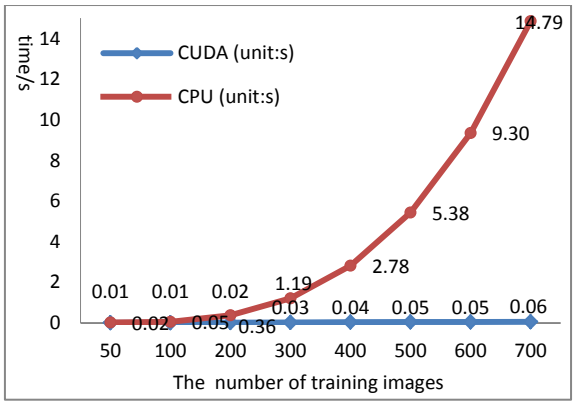

Fig 7. The recognition speed with different numbers of training images

Considering the factors which affect the performance of the real-time face recognition system, we test 40 images captured from the video with different size of the database.Fig.8 shows the speedup of CUDA. In face recognition phase, each of the testing images corresponds to a block. Since there are no enough database images, we can't measure the peak speedup. The experiment statistics show that the accuracy of recognition almost reaches 98%.

TABLE I. THE CORRECT DETECTION AND RECOGNITION RATE

	correct detection rate	*correct recognition rate*	*system correct recognition rate*
CUDA	89.775%	97.189%	87.251%
CPU	89.571%	96.875%	86.771%

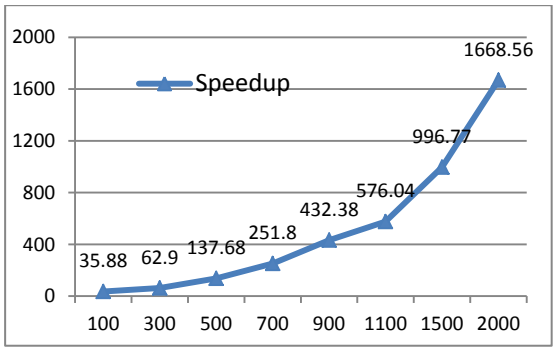

Fig 8. The speedup of CUDA

However, shown as table I, though the CUDA program and the CPU program is almost the same, the system's correct recognition rate is not very high. The relatively low correct detection rate will be responsible for it. Since we don't detect the face's location, we can just not recognize it. If we want to improve the system's correct recognition rate, the correct detection rate should be improved first.

Because the block in GTX 570 maximum support 1024 threads, we need to reorganize the threads when the number of training images exceeds 1024. The block in the grid should be changed to 2-dimensional form. The training image is identified by both block's x index and thread's index. The testing image is determined by the block's y index.

The experiment demonstrates that the face recognition system based on CUDA is faster than the program runs in CPU and the correct rate almost the same. This means that we really achieved a real-time face recognition system.

V. Conclusions

We have proposed a real-time face recognition system based on CUDA, which effectively completed the face detection and recognition tasks. Our face recognition system has a high acceleration performance because that we not only did parallel optimization to detection part but also did it to recognition part in this system. Compared with the CPU program, our program implements a high speed. This increase in speed will enable real-time face recognition applications on occasions that need more real-time applications.

We plan to use the new equipment, such as NVidia's Kepler architecture graphics card, to solve the data migration questions in recognition phase, and this also will do better to the detection phase. We also plan to make a new classifier [14] which has more contribution to correct detection correct rate. Through optimization, the system will be greatly improved.

Acknowledgment

This research was supported by Shaanxi Natural Science Foundation (Grant No. 2013JQ8034), the Doctoral Fund of Ministry of Education (Grant No. 20116102120049) and the Basic Research Foundation of Northwestern Polytechnical University (Grant No. JC20120239).

References

[1] P. Viola, M. Jones, "Rapid object detection using a boosted cascade of simple feature" in Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Hawaii, vol.1, pp. 511-518, August 2001.

[2] V. Freund, R. E. Schapire, "A short introduction to boosting," JSAI Transl. Japan, vol. 15, no.5, pp. 771-780, Sep. 1999.

[3] Turk M,Pentland A P. Eigenfaces for Recognition[J]. Journal of Cognitive Neuroscience, 3(1):71-86, 1991.

[4] V.P. Kshirsagar, M.R.Baviskar,and M.E.Gaikwad,"Face Recognition Using Eigenfaces," In Proc. of IEEE Int. Conference on Computer Research and Development, pp. 302-306, 2011.

[5] R.Farber, CUDA Application Design and Development, Massachusetts: Morgan Kaufmann, 2011, pp. 2-16.

[6] Ping Zhang "A video-based face detection and recognition system using cascade face verification modules", IEEE Applied Imagery Pattern Recognition Workshop, pp.1-8, 2008

[7] K. Rujirakul , So-In, C. Arnonkijpanich, B. Sunat, K. Poolsanguan, S."PFP-PCA: Parallel Fixed Point PCA Face Recognition", International Conference on Digital Object Identifier, pp.409 – 414, 2013

[8] U. Ali , M. Bilal, "Video based Parallel Face recognition using Gabor filter on homogeneous distributed systems" IEEE Engineering of Intelligent Systems, International Conference on Digital Object Identifier,pp.1-5, 2006

[9] F.Crow, "Summed-area tables for texture mapping", in Proceedings of SIGGRAPH, 18(3):207–212, 1984

[10] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection", ICIP02, pp. I: 900–903, 2002

[11] J. Sanders, E. Kandrot, CUDA by Example: An Introduction to General-Purpose GPU Programming, Boston: Addison-Wesley Professional, 2010, pp. 27-41.

[12] NVIDIA Inc., "CUDA_C Programming Guide," 4.2 ed. Sisha Clara: NVIDIA, 2012.

[13] ORL, "The ORL Database of Faces" Apr. 1994. DOI= http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

[14] Xueming Qian, Yuan Yan Tang, Zhe Yan et al. ISA Boost: A weak classifier inner structure adjusting based AdaBoost algorithm-ISA Boost based application in scene categorization [J]. Neurocomputing, 2013,103(Mar.1):104-113.