© 2017 IJEDR | Volume 5, Issue 2 | ISSN: 2321-9939

GPU Based Face Recognition System for Authentication

Bhumika Agrawal, Chelsi Gupta, Meghna Mandloi, Divya Dwivedi, Jayesh Surana
Information Technology, SVITS
Gram Baroli, Sanwer road, Indore, MP, India

ABSTRACT-Face has significant role in identifying a person for authentication purpose in public places such as airport security. Face recognition has many real world applications including surveillance and authentication. Due to complex and multidimensional structure of face it requires huge computations therefore fast face recognition is required. One of the most successful template based techniques for face recognition is Principal Component Analysis (PCA) which is generally known as Eigen face approach. It suffers from the disadvantage of higher computation cost, despite its better recognition rate. With the increase in number of images in training database and also the resolution of images, the computational cost also increases. Graphics Processing Unit (GPU) is the solution for fast and efficient computation. GPUs have massively parallel multi-threaded environment. With the use of GPU's parallel environment, a problem can be solved in parallel with much less time. NVIDIA has released a parallel programming framework CUDA (Compute Unified Device Architecture), which supports popular programming languages with CUDA extension for GPU programming. A parallel version of Eigen face approach for face recognition is developed using CUDA framework.

Keywords: GPU, CUDA, NVIDIA, PCA Eigenfaces, face recognition.

I-INTRODUCTION:

Face recognition in images is quite complicated and a time consuming problem in security systems at public places, which found use in different disciplines, e.g., security, robotics, or advertising. Face recognition is one of the important methods of biometric identification. Developing a face recognition model continues to be an extremely fascinating field for many researchers mainly because of its many real world applications like criminal identification, user authentication, security systems and surveillance systems. However, due to its complex and multidimensional structure, it is difficult to develop a face recognition model.
Face recognition is the process of recognizing faces in input images. Face detection is first step in face recognition system. Initially expensive hardware devices was used for this purpose to achieve face recognition at desired rate. The first software based real time face detection algorithm was proposed by Viola and Jones. Parallel implementation over serial implementation is the best way to achieve faster execution. To recognize the face we need to perform two steps, first we need to create the training database then we need to perform face detection for recognizing the input image. To detect face in a given image we need to run a sub window over the image and check whether it is a face or not. This particular step of checking a sub window for face basically consists of executing same set of instructions on each sub window. The calculation of each sub window doesn't depend on any other sub window, which makes face detection a highly parallelizable problem. Since GPU is a Single Instruction, Multiple Threads processor it is very much suitable for performing these calculations in parallel and thus saving a lot of processing time. With the help of NVIDIA CUDA toolkit it is now possible to perform general purpose computations on GPU. In our work, we have developed GPU based face detection system based on Principal Component Analysis algorithm. To verify our work, we compared performance of our implementation with CPU based implementation. We found that our GPU based implementation performed 5.41 to 19.75 times faster than its CPU implementation.
Application in Airport security:
Face recognition can be used in airport security for:
1. Surveillance: Real Time Watch list Alerts
2. Surveillance: Forensic Video Analysis for Lost people (Children, Elders)
3. Passengers without Documentation
4. Marketing purposes

II-RELATED WORK

The proposed architecture is mainly based on Principal Component Analysis face recognition algorithm. The algorithm can be seen as a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. The number of principal components is less than or equal to the number of original variables. 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 pre-processed. 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. The algorithm can be seen as a transformation defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set.
A lot of work is being done for accelerating the face recognition process. Highly optimized OpenCV implementation of face recognition algorithm provides speed of 1.78FPS for images of size 640x480(VGA images). There exists few GPU based Implementation of Viola-Jones algorithm. The fastest among them provides a speed of about 15.2 FPS using the cluster of four Graphics processors. But we expect that the algorithm used by us will be most suitable for face recognition using GPU.

III-GPU ARCHITECTURE AND CUDA:

A CUDA capable GPU consists of a set of streaming multiprocessors (SMs) and each streaming multiprocessor has a number of processor cores. A streaming multiprocessor core is known as streaming processor (SP). The number of streaming processors each streaming multiprocessor contains depends on the GPU. Generally, in modern GPU each streaming multiprocessor contains 32 streaming processors. So if a GPU has 512 cores that mean it contains 16 streaming multiprocessors each containing 32 cores or streaming processors. The programs running on GPU are independent of architectural differences which make GPU programming scalable.

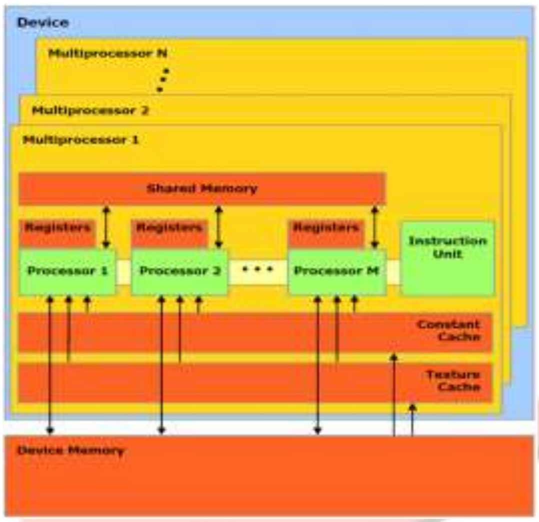

Fig 1: GPU Architecture

CUDA (Compute Unified Device Architecture) is a proprietary GPGPU(A general-purpose GPU (GPGPU) is a graphics processing unit (GPU) that performs non-specialized calculations that would typically be conducted by the CPU) platform from Nvidia. It is used for utilizing Nvidia made graphics cards for general computing. The platform involves sequential programming on the CPU and kernel functions (CUDA threads) executed on the GPU in the SPMD manner. A CUDA program consists of two kinds of codes, one is the standard C code which runs serially on the CPU, and the other is the extended C code which executes parallely on the GPU. In the CUDA programming model, CPU is also refers to a host, while GPU refers to a device. NVIDIA provides us a nvcc compiler that divides a CUDA program into two part, first part consist of host code and other part consist of device code. Host code is compiled by standard c compiler and runs on CPU in sequential manner and device code is compiled by nvcc compiler and runs on GPU parallely. This is how we are utilizing GPU and CPU both for executing our program parallely and perform faster execution using CUDA.

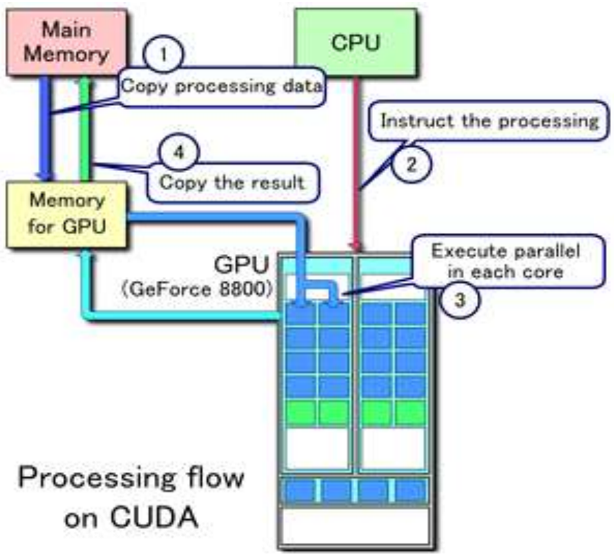

Fig 2: CUDA Process Flow Diagram

IV-FACE RECOGNITION ALGORITHM:

We have used Principal Component Analysis algorithm in our work. Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. PCA is sensitive to the relative scaling of the original variables.

PCA is mostly used as a tool in exploratory data analysis and for making predictive models. PCA can be done by eigenvalue decomposition of a data covariance (or correlation) matrix or singular value decomposition of a data matrix, usually after mean centering (and normalizing or using Z-scores) the data matrix for each attribute. The results of a PCA are usually discussed in terms of component scores, sometimes called factor scores (the transformed variable values corresponding to a particular data point), and loadings (the weight by which each standardized original variable should be multiplied to get the component score).

PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

Computing PCA using the covariance method
The goal is to transform a given data set X of dimension p to an alternative data set Y of smaller dimension L. Equivalently, we are seeking to find the matrix Y, where Y is the Kosambi-Karhunen–Loève transform (KLT) of matrix X:
Y=KLT(X)
Step 1: Organize the data set
Step 2: Calculate the empirical mean
Step 3: Calculate the deviations from the mean
Step 4: Find the covariance matrix
Step 5: Find the eigenvectors and eigenvalues of the covariance matrix
Step 6: Rearrange the eigenvectors and eigenvalues
Step 7: Compute the cumulative energy content for each eigenvector
Step 8: Select a subset of the eigenvectors as basis vectors
Step 9: Convert the source data to z-scores (optional)
Step 10: Project the z-scores of the data onto the new basis

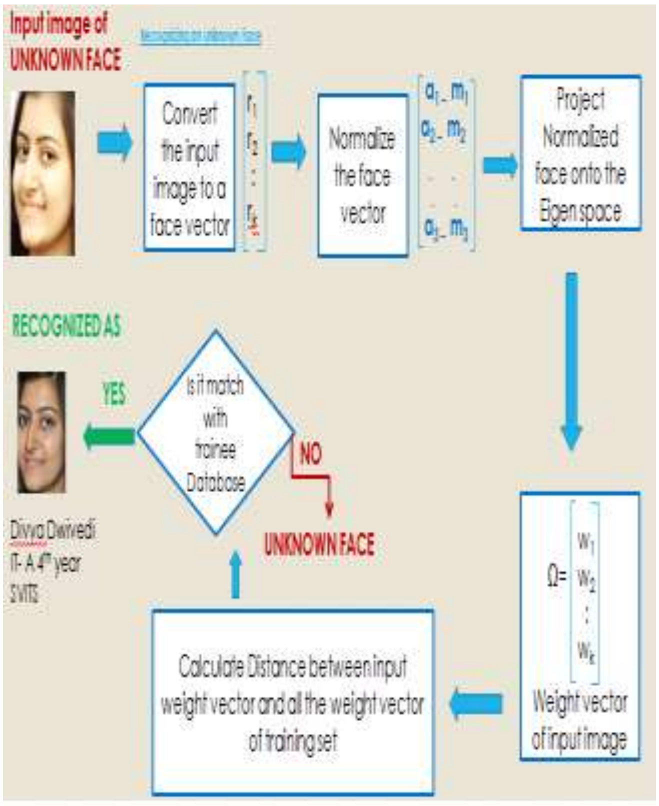
Input image of UNKNOWN FACE
Convert the input image to a face vector
$[r_1, r_2, \vdots, r_k]$
Normalize the face vector
$[a_1 - m_1, a_2 - m_2, \dots, a_3 - m_3]$
Project Normalized face onto the Eigen space
RECOGNIZED AS
YES
Is it match with trainee Database
NO
UNKNOWN FACE
Divya Dwivedi
IT- A 4th year
SVITS
$\Omega = [w_1, w_2, \vdots, w_k]$
Weight vector of input image
Calculate Distance between input weight vector and all the weight vector of training set

Fig 3:Face Recognition Using PCA for authentication purpose

V-CONCLUSION AND FUTURE WORK:

We have proposed a real-time face recognition system based on GPU, which effectively completes the face detection and recognition tasks which help in authentication of a person. Our face recognition system has a high acceleration performance because 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 as in many authentication systems. 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, which will have more contribution to correct the detection rate. Through optimization, the system will be greatly improved

V-REFERENCES:

[1]Manik R. Kawale (June 2014): Parallel Implementation of Eigenfaces for Face Recognition on CUDA by

[2] Vaibhav Jain, Dinesh Patel (2016): A GPU based implementation of Robust Face Detection System

[3]Divyarajsinh N. Parmar, Brijesh B. Mehta (Jan-Feb 2013): Face Recognition Methods & Applications

[4]Kailash Devrari, K.Vinay Kumar (2011): Fast Face Detection Using Graphics Processor
[5] M. Hradis, A. Herout, P. Zemcik. Local rank patterns: novel features for rapid object detection. In: ICCVG '2008: International
Conference on Computer Vision and Graphics, Warsaw, Poland; 2008, p. 239-248.

[6] B. Sharma, R. Thota, N. Vydyanathan, A. Kale. Towards a robust real-time face processing system using CUDA-enabled GPUs. In:
HIPC'2009: IEEE 18th International Conference on High Performance Computing, Kochi, India; 2009, p.368-377.

[7] Shivashankar J. Bhutekar, Arati K. Manjaramkar. Parallel face Detection and Recognition on GPU. In: International Journal of
Computer Science and Information Technologies, vol. 5; 2014, p. 2013-2018.