ISSN:0975-9646

Parallel face Detection and Recognition on GPU

Shivashankar J. Bhutekar[1], Arati K. Manjaramkar[2]

[1]*Research Scholar* [2]*Associate Professor*

Shri Guru Gobind Singhji Institute of Engineering and Technology Nanded, India

Department of Information Technology (Nvidia Research Lab)

***Abstract*— Human face detection and recognition finds various application in domain like Surveillance, Law Enforcement, Interactive game application etc. These application requires fast image processing in real time however with the proposed work earlier does not gives that capability. In this paper we have proposed technique that process image for face detection and recognition in parallel on NVIDIA GeForce GTX 770 GPU which has compute Capability of 3.0. We have used Viola and Jones for face detection and PCA Eigenfaces algorithm for face recognition. The viola and jones shown result of processing the faces at 15 frame per second but our method shown approximately 109 frame per second in CUDA (Compute Unified Device Architecture) development framework. The algorithm are implemented to process data in parallel and made some practical optimization changes to work fast in C based programming model from NVIDIA . We have tested our proposed work on static images, video frame as well as on live camera frames captured device. We also tested that our system works robustly in extremely illumination condition. This work is done by PG students in Institute NVIDIAs Research Lab.**

***Keywords*— NVIDIA, PCA Eigenfaces, GeForce GTX 770, GPU, CUDA, face detection, face recognition.**

I. INTRODUCTION

The machine computation of human faces is most and widely active research topic in domain of Image processing, pattern recognition and in computer vision. The fact that the image processing extracted feature provide clue in many security, surveillance, banking, commercial sector system. However the face detection and recognition are increasingly used in video conferencing, video games, interactive game play station, virtual reality system etc. The real time system require algorithm that make interactivity as well as critical nature of time which makes necessity to design the parallel algorithm that process the huge amount of data generated by capturing devices like smart cameras.

A typical face processing system is composed of face detection, face recognition, face tracking and rendering. Different faces can be tracked and recognized despite of different location. In such a case it might be useful to track a person from one camera view to another camera by handing off face to another process. Now these type of information to process is critical in secure environment. To use these three modules i.e. Face Detection, Face Recognition and Tracking, system should process on current frame rates generated by device. This scenario becomes advert when we try to process multiple streams generated by multiple video devices. However thanks to Face Detection and Recognition algorithm that they can be computationally feasible for parallel processing.

In recent trends we have seen that Graphics processing units emerges with high parallel computing capabilities. For example the GeForce GTX 770 developed by NVIDIA graphics card can gives 3213 GFLOPS. The CUDA proposed by NVIDIA [1] is C-based programming model gives developer to explore the use parallel computation capabilities of GPU in easy manner without mapping algorithm to graphics programming structure.

In this paper we have used the computation capability of Modern GPU to achieve extreme parallelism for face detection and recognition. As above GPU has a compute capability of 3.0 [2] which provides a feature that below 3.0 doesn't support. The global function that run on device can be called from both host and device in compute capability greater than 3.0 that provide multiple thread launching from multiple previously launched threads. For face detection we have used Viola and Jones [3] algorithm with Adaboost framework and face recognition carried out with M. Turk, A. Pentland, "Eigenfaces for Recognition" [4] which provide ability to recognize the faces from above face detector. We have developed the parallel algorithm from above algorithms and implemented them in graphics processor with the help of CUDA.

We have tested this our propped technique on static images database, video database as well as live frames captured from video devices. Our experimental results shows that speed of face detection and recognition is much higher rates as compare to previous work. We also demonstrated our system is robust to changing illumination scenarios.

This paper is organized in following section. Section II provides the related done work in face detection and recognition. Section III gives overview of GPU architecture and memory model for general device. Section IV describe Viola and Jones face detection method and its implementation in CUDA with cascade detection. Section V gives procedural and parallel implementation for face recognition also describes the overall system design. Finally section VI and VII gives performance analysis and conclusion with future work.

II. RELATED WORK

The Image processing is widely research topic from last two decade in which face detection and recognition is mostly explored topic. The researcher proposed many technique face detection [3], [5], [6], [7], [8] .Out of these work the Viola and Jones [3] provided solution detect faces

at 15 fps (for 320×288 images) with adaboost learning algorithm on CPU. The work done in [9] detect the faces at 90 fps (for 320×240 images) with maintaining accuracy for low resolution images.

For Face recognition the[4], [10], [12],[13], [14] are the proposed techniques among these technique the PCA proposed by M. Turk, A. Pentland [4] works on the training set of M images to prepare face space. Training has much more computation but initially it has done identifying the unknown face can be done in minimum time. The work done by Y. Woo, C. Yi, Y. Yi [14] for face recognition on GPU shows calculation for 400 training set it takes covariance matrix multiplication 120 ms, to calculate eigenfaces 60 ms and projecting a face take 95 ms which provides a feasible solution.

With CUDA programming model we can process these two task i.e. face detection and face recognition in parallel. The graphic processor has a compute capability of 3.0 can launch multiple kernels from device. This provides a much higher detection and recognition rates. As compare to OpenGL CUDA provide faster data Transfer between CPU to GPU whenever the memory is required by threads. CUDA can also map host memory [15] that can be accessed by kernel threads. In our work we use host mapped memory as a shared Buffer that can be accessed by both GPU and CPU. The advantage of creating a shared Buffer we doesn't require to transfer the whole data to GPU Memory When the thread request the data the shared buffer data transferred to block memory. This type of memory access is faster than cudaMemcpy function. Our face recognition system provides much higher rates than previously proposed solution also this is the attempt to detect faces and recognize them parallel on GPU.

III. GPU Architecture and CUDA Programming Model

The Graphics processor has a tremendous computation capability for parallel computation previously used for Image rendering. CUDA is a C – based programming model provide easy development and deployment in general purpose computation. In this section we will see salient feature of CUDA programming and GPU architecture.

The GPU has multiple processor and each processor is a set of SIMD (Single Instruction Multiple Data) processor known as streaming multiprocessor. For example the GeForce GTX 770 has 192 multiprocessor that means each has 16 SIMD cores. The device has separate global memory where global data gets store, constant memory stores readable data and texture memory used to perform read only memory.

Figure 1 shows a CUDA programming model memory model has a parallel computation block called as grid. Grid is launched from host code. The CPU called as host and GPU termed as device. A grid has 3 dimension launched in x, y ,z coordinates form multiple grids can be launched in one device as shown kernel 1,kernel 2. Grid contains thread block also launched in 3 dimension usually referred to access thread Index and Block Index. Figure 3 shows the example of kernel launch of (3×2) block and each block contain (5×3) threads These threads are unit of execution and each thread cab be uniquely identified by block index and thread index

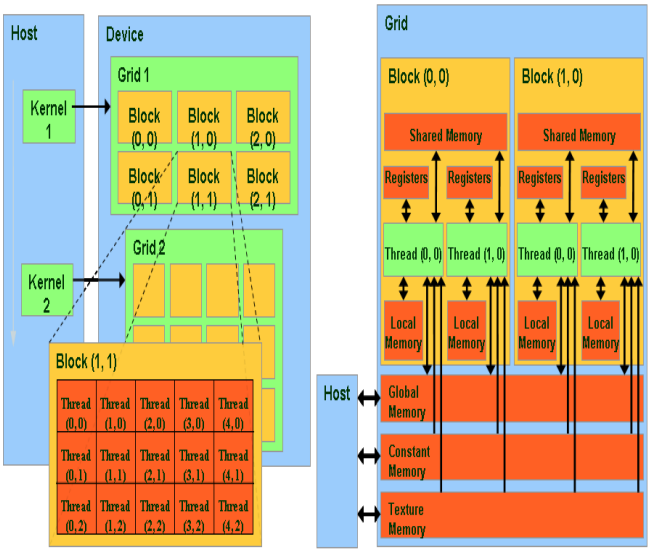

Fig. 1 CUDA Programming and Memory model (courtesy: NVIDIA)

IV. CUDA Based Parallel Face Detection

A. CUDA Image Pyramid

Image pyramid down sampling construction is done in on GPU with taking scale factor of 1.2. We used nearest neighbour down sampling method for creating the pyramid for every block organized to detect a face present in it. Each block calculate its own image down sampled until the image size becomes 24×24. In viola and jones [3] the detection window is taken as 24×24.The Fig. 2 shows image pyramid for respective blocks.

Fig. 2 CUDA Image pyramid for each grid block

B. CUDA Integral Image Calculation

Viola and jones uses adaboost machine learning algorithm for accurate and fast face detection the algorithm uses to pick up the most promising feature from over complete set of haar feature to recognizes it is a face or not. Figure 3 shows the four kinds of haar feature that need to apply on image. Generally we can use 10^5 such feature of different sizes and location on to image.

The Viola and jones used the frame detection window of size 24×24. Now these feature are applied on to image to calculate the sum of all pixel values under dark region subtracted from sum of all pixel values under bright region.

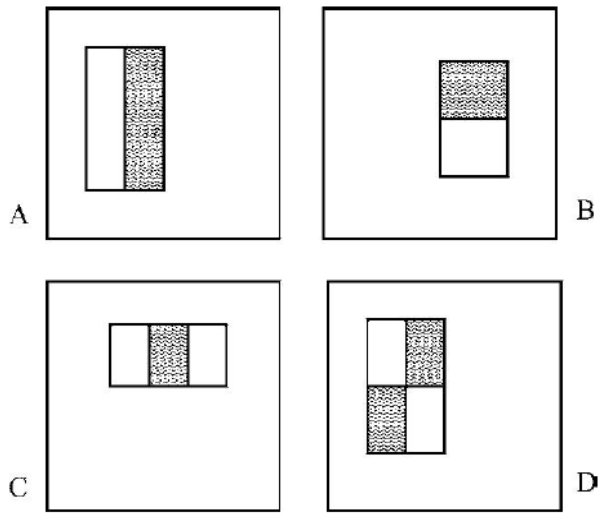

Fig. 3 haar feature rectangle A and B are 2 rectangle feature C and D are 3 and 4 rectangle feature to calculate the prefix sum (courtesy : [3])

For haar feature A (shown in fig. 3) applied on image gives a higher value at nose area over a complete image. But this method not feasible Viola and Jones used Integral image calculation [3] .The integral image calculation at location (x, y) is a sum of the pixel values above and to left of the point (x ,y).

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

Using the integral image we can calculate rectangle sum easily in four value access. Now we can calculate the image integral on GPU using vertical prefix or by horizontal prefix as shown in fig. 4 for each thread calculate sum row wise or can be done in column wise.

A. Cascade Detection

Integral image represented in previous section computes the sum of for rectangle in Fig. 3 shows that for rectangle A and B takes six value access from cornel points for figure C and D it takes eight and nine point access respectively. There are large set these feature viola and jones [3] solved this problem by defining feature as weak classifier and strong classifier. The weak classifier decided as threshold value assigned to them.

$$h(U) = \begin{cases} 1 & if\ pf(U) < p\theta \\ 0 & otherwise \end{cases}$$

The equation h (U) is a weak classifier of image detection window 24×24 indicated as U. Θ is a threshold over parity p. This equation construct a set of weak classifier which are used in adaboost learning algorithm to construct a strong classifier. Combining these classifier gives to form a strong classifier gives probability of sub window 24 ×24 has a face or not. Following equation gives describe strong classifier.

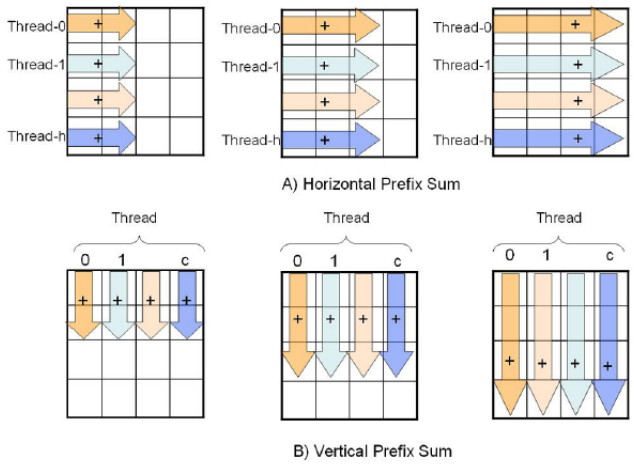

Fig. 4 Integral image calculation vertical prefix sum and horizontal prefix sum (Courtesy [9])

$$H(U) = \sum_{t=1}^{T} \alpha_t h_t(U)$$

H (U) is a linear combination of weak classifier in a stage from 1 to T where α_t is a normalized weight of h_t weak classifier. Image of size 480 × 600 has 35 detection sub window. Alternatively the size of window increased as the image pyramid downscaled as shown in Fig. 2.To compute strong classifier over a window of size 24 × 24 takes huge amount of computation for all weak classifier. Viola and Jones [3] comes with stagging solution consist set of weak classifier in each stage. The first stage weak classifier give most promising solution weather sub window contain a face or not. The size number of weak classifier of stage increases from one stage to another. Fig 5 shows at first stage if a sub window passes first stage it given to second stage shows that the window contain face else rejected from further processing. Maximum non face sub window rejected in first and second stage.

The cascade detector when passes all stages stage gives result that the sub window has a face. This face is given to another CUDA device function for face recognition described in next section.

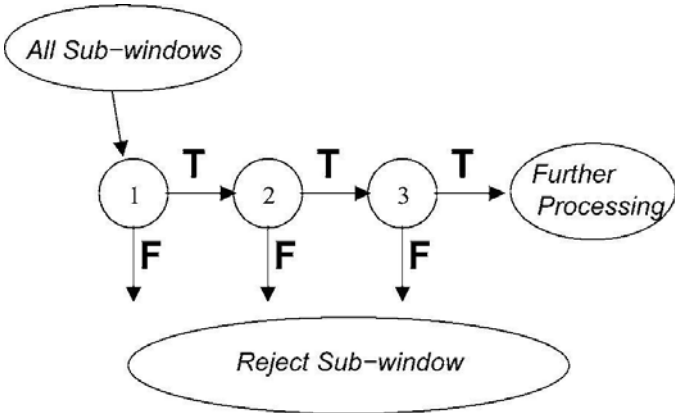

Fig. 5 A schematic progress of strong classifier as every window does this processing in stages maximum non faces are rejected in stage 1 and 2 further processing avoided (courtesy :Viola and Jones [3])

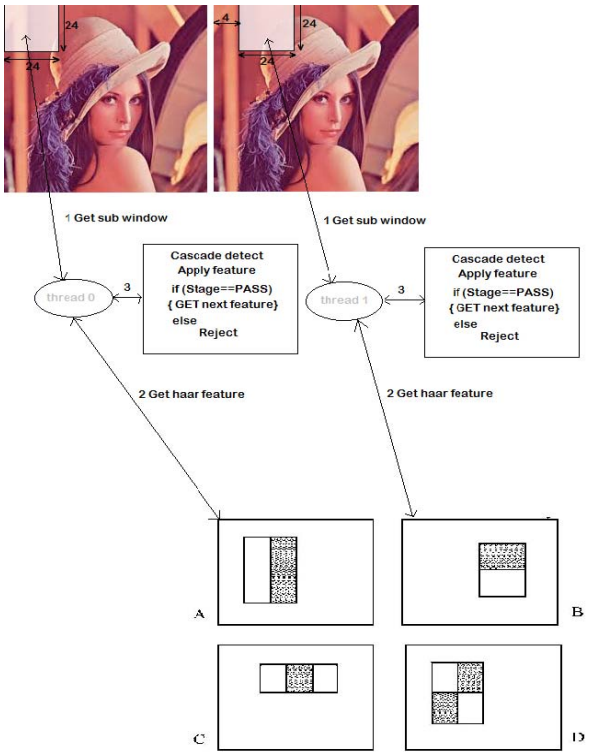

Fig. 6 grid view of a particular block along with thread operation

V. CUDA FACE RECOGNITION

Face Recognition we have used technique described by M. Turk, A. Pentland [4]. Eigenface technique works on previously training dataset from known faces calculates its Eigenfaces. For our experiment we used two dataset first yale dataset [16] of images contains 5760 images (90×112) of 10 subjects size .second database used from BioId face dataset [17] 1521 images (384×286) of 23 different person. We also included the local faces in our experiment. The training have been done on CPU but new face recognized on GPU.

The set of training is organized to compute Eigenfaces by taking all M images. Computes the average face from image set as adding all matrix into each other taking average let I be set of training images.

$$\mathrm{I} = \{\Gamma_1, \Gamma_2, \Gamma_3 \ldots\ldots \Gamma_M \}$$

The mean face can computed and subtracted from all training image set.

$$\Psi = \frac{1}{M}\sum_{n=1}^{M}\Gamma_n$$

$$\Phi_i = \Gamma_i - \Psi$$

The sample training set of 16 people and there mean face is shown in Fig.7 and Fig. 8 respectively. Converting the two dimensional images of N×N into one dimension $N^2 \times 1$ into matrix A size $N^2 \times M$ .For computing the covariance matrix need $N^2 \times N^2$ computation which is not feasible as

$$C = AA^T$$

C has N^2 eigen vectors which are difficult to choose from and ay also run out of memory.so C can be computed as

$$C = A^T A$$

Subsequently the no of computation reduced from $N^2 \times N^2$ to M×M also covariance matrix C has M no of eigen vectors denoted as u_i. The eigen vectors are selected from M eigen vector.

$$u_l^T u_k = \delta_{lk} = \begin{cases} 1 & if \quad l = k \\ 0 & otherwise \end{cases}$$

$$\lambda_k = \frac{1}{M}\sum_{n=1}^{M}(u_k^T \Phi_n)^2$$

Where u_k is a eigen vector and λ_k is eigen value for covariance matrix C.

Whenever a face is detected by any of the thread from face detection it is given to face recognizer. Let new face be Γ transformed into its eigenface component as

$$w_k = u_k^T(\Gamma - \Psi)$$

Where k is number of eigen faces in face space [4].The weights of eigenfaces $\Omega^T = [\omega_1, \omega_2, \omega_3 \ldots \omega_{M'}]$ provides the contribution of each training images into input image eigenface. Now the by using euclidian minimum distance we can find the best describing face from training set.

$$\varepsilon_k = \| \Omega - \Omega_k \|$$

Where k is a number of eigenface that give ε_k minimum over some threshold value θ. If a new face passes minimum Euclidian distance then it the face from training set else it is an unknown face. The Fig. 9 shows the complete over view of our proposed system.

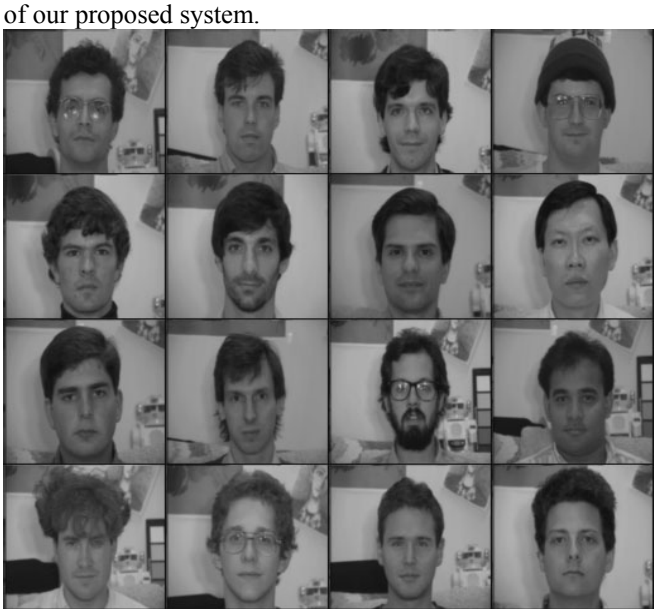

Fig. 7 an example of training set data with 16 image face (courtesy M. Turk and A. Pentland [18])

Fig. 8 Mean face of faces from Fig 7 (courtesy M. Turk and A. Pentland [18])

As the system overview describe each block computes its own image from up to down sampling. Each thread computes its sub window to find whether window contains face or not by using viola and jones face detection algorithm [3]. If any thread find face that face resizes to training image size. The new face will be processed instantly for recognition (as described in section V). Since the face get processed instantly save time to other faces recognize. Our experimental result show that difference.

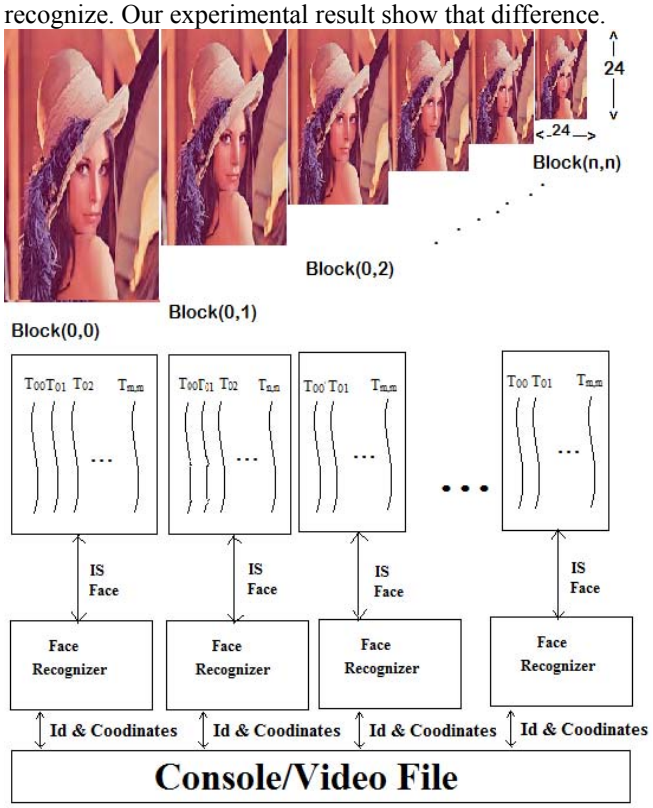

Fig. 9 Complete overview of face processing system

VI. Performance Analysis

The programming language used for CPU is C++ (gcc version 4.6.3) and for GPU programming we used CUDA 5.5 C-based programming language provided by NVIDIA. For loading and saving images used openCV 2.7.1 for support. The processor intel ® Xenon(R) CPU E3-1220 V2 has 3.10GHz clock speed with 4 core and has 7.8 GiB RAM. The GPU used GeForce GTX 770 GPU has 1536 cores, memory bandwidth of 220.6 GB/sec and has a 2 Gib of global memory. The Fig. 10 shows some snapshot taken form video device to show a person identity. Fig 10, 11 shows result on static images.

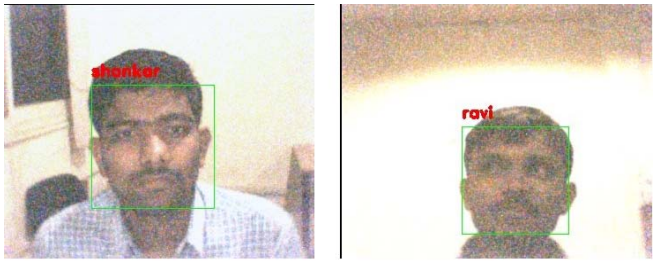

Fig. 10 Face detection and Recognition via live video device captured frames

Fig. 11 face processing on static image.

Fig 12 gives a comparative analysis between the GPU and CPU.As the frame size increases CPU execution time increases rapidly for the last two sizes the CPU even takes 2 second to process the frame but for GPU it takes less than second

Fig. 13 shows the response of face detector and recognizer for on GPU as well as for CPU. As the face recognizer work in parallel with face detector on GPU the required time to finish as compare to CPU execution is extremely less.

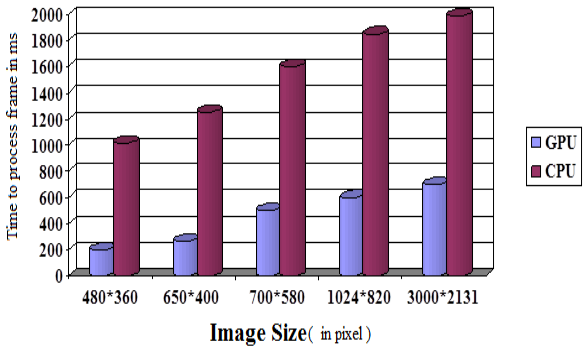

Fig. 12 statistical analysis of face processing system on GPU vs CPU with different frame sizes.

Remember that these reading are taken for varying width and height of same video device along with the one person in front of camera device. On CPU face processing system takes much time than GPU as it has to wait for all faces detected in frame but in case of parallel system any face detected at any time gets recognized instantly. That's make a difference in between CPU implementation and GPU implementation in real time.

Time (in ms)	Image Size					
	480 * 360		650 * 400		700 * 580	
	GPU	CPU	GPU	CPU	GPU	CPU
Face Detection	102	780	160	900	350	1406
Face Recognition	90	207	127	279	312	300

Fig. 13 Time taken by individual component on GPU vs CPU.

Fig.14 and 15 are frames of size 1219×810 and 2048×1536 respectively. The first frame has taken 893.78 ms and second frame has taken 1466.38 ms for overall processing (excluding image loading and writing to hard disk)

Fig. 14 Frame size of 1219×810 taken 893.78 ms for processing on GPU

Fig. 15 Frame size of 2048×1536 taken 1466.38 ms for processing on GPU

VII. CONCLUSION AND FUTURE WORK

As the developed system processes face both detection and recognition in real time on GPU fastest than CPU. The algorithms are run in parallel one detects a face and forward it to face processing system is a key point of this paper. In conventional face processing system face recognizer has to wait until all the faces in frame gets detected therefore a time consumption is large. As from the experimental results it is clear that the GPU plays a vital role in parallel computing.

This system is for one camera device if there are multiple cameras and wants to track a person from one device to another how the hand off take place between multiple devices? How to synchronise time frame between two devices? are the future work for this paper.

REFERENCES

[1] VIDIA C- programming guide http://docs.nvidia.com/cuda/cuda-c-programming-guide.

[2] Compute Capabilities of GPU https://developer.nvidia.com/cuda-gpus.

[3] P. Viola and M. J. Jones, "Robust real-time face detection," Int. J. Computer. Vision, vol. 57, no. 2, pp. 137–154, 2004.

[4] M. Turk, A. Pentland ,"Eigenfaces for Recognition", Journal of Cognitive Neuroscience, pp.71-86, 1991

[5] K.-K. Sung and T. Poggio, "Example-based learning for view-based human face detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 1, pp. 39–51, 1998.

[6] H. A. Rowley, S. Member, S. Baluja, and T. Kanade, "Neural network based face detection," IEEE Transactions On Pattern Analysis and Machine intelligence, vol. 20, pp. 23–38, 1998.

[7] *H. Schneiderman and T. Kanade, "A statistical method for 3d object detection applied to faces and cars," IEEE Conference On Computer Vision and Pattern Recognition, vol. 1, pp. 746–751, 2000.*

[8] M. hsuan Yang, D. Roth, and N. Ahuja, "A snow-based face detector," in Advances in Neural Information Processing Systems 12. MIT Press, 2000, pp. 855–861.

[9] Bharatkumar Sharma, Rahul Thota, Naga Vydyanathan, and Amit Kale "Towards a Robust, Real-time Face Processing System using CUDA-enabled GPUs" High Performance Computing (HiPC), 2009 International Conference 2009, Page(s): 368 – 377.

[10] A. K. Jain, R. Bolle, and S. Pankanti, "Biometrics:Personal Identification in Networked Security," A. K. Jain, R. Bolle, and S. Pankanti, Eds.: Kluwer Academic Publishers, 1999.

[11] K. Kim, "Intelligent Immigration Control System by Using Passport Recognition and Face Verification," in International Symposium on Neural Networks.

[12] J. N. K. Liu, M. Wang, and B. Feng, "iBotGuard: an Internet-based intelligent robot security system using invariant face recognition against intruder," IEEE Transactions on Systems Man And Cybernetics Part C-Applications and Reviews, Vol.35, pp.97-105, 2005.

[13] H. Moon, "Biometrics Person Authentication Using Projection-Based Face Recognition System in Verification Scenario," in International Conference on Bioinformatics and its Applications. Hong Kong, China, 2004, pp.207-213.

[14] Youngsang Woo, Cheongyong Yi, Youngmin Yi "FAST PCA-BASED FACE RECOGNITION ON GPUS"

[15] CUDA Runtime API http://docs.nvidia.com/cuda/cuda-runtime-api.

[16] Yale face database http://vision.ucsd.edu/datasets/yalefacedataset

[17] BioId face dataset ftp://ftp.uni-erlangen.de/pub/facedb/BioID-FaceDatabase-V1.2.zip

[18] M. Turk and A. Pentland, "Face Recognition Using Eigenfaces," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1991, pp.586-591.