ISSN:0975-9646
Kailash Devran et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 2 (3) , 2011,1082-1086

Fast Face Detection Using Graphics Processor

Kailash Devrari[#1], K.Vinay Kumar[#2]

[#] *Department of Computer Science and Engineering, National Institute of Technology Karnataka Surathkal, India*

***Abstract*— Fast face detection is one of the key components of various computer vision applications. Viola-Jones algorithm provides a good and fast detection for low and medium resolution images. This paper proposes a new and fast approach to perform real time face detection. The proposed method includes the enhanced Haar-like features and uses SVM for training and classification. Experimental result shows that our proposed architecture works well for high resolution images. Our design model combines conventional programming using CPU with GP-GPU programming using NVIDIA CUDA for fast face detection.**

***Keywords*— face detection, Haar-like Features, Integral Image, SVM, GPU, CUDA.**

I. INTRODUCTION

Face detection is the process of detecting faces in input images. Face detection is important because it is the first step in various applications like face recognition, video surveillance, Human Computer Interaction etc. Since all the mentioned applications are often used in real time, so there is a need of a fast face detection system. Traditionally expensive dedicated hardware was used to achieve the desired rate of detection. The first Software-based real-time face detection algorithm was proposed by Viola and Jones [1]. It has now become the de-facto standard for real-time face detection. However it doesn't suits well for images with high resolution, hence we need to look for high performance face detection solutions for fast face detection with reasonable cost.

Parallelisation is the best way to achieve faster face detection. 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 (SIMT) 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.

The main contributions of this paper are:

1. It gives a detail explanation about calculation of Integral Image in Parallel.
2. Implementation of modified Viola Jones algorithm on graphics processors.
3. Modification of GPUSVM [2] to work properly with proposed face detection algorithm.

Rest of this paper gives details about the algorithm, its CUDA implementation and the results obtained. Section II gives a brief overview of related work done, section III describes the concept of Integral Image and haar like features, and section IV gives details of the proposed architecture. In section V we describe the algorithm used and GPU implementation of the algorithm. Section VI contains the results obtained and section VII contains the conclusion and section VIII contains the future work needed to be done.

II. RELATED WORK

The proposed architecture is mainly based on Viola and Jones face detection algorithm [1]. They introduced the concept of Integral Image, Haar-like features and Cascaded Adaboost classifier for face detection. The algorithm can be seen as a detector window scanning the image, looking for features of human face. If more than a particular number of features are present in the detector window then it is considered as the face. Integral Image and Features, two of the components of Viola and Jones algorithm, used in the proposed architecture are explained in section III.

A lot of work is being done for accelerating the face detection process. Highly optimized OpenCV implementation [3] of face detection 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 [4] using the cluster of four Graphics processors.

III. INTEGRAL IMAGE AND HAAR-LIKE FEATURES

A. Integral Image

Integral Image Int.Img.(x,y) at any point (x,y) contains the sum of all the points above and to the left of the point (x,y).

$$\text{Int.Img}(x,y) = \sum\sum(x_i,x_j) \quad (1)$$

Since calculating the sum of pixel inside a rectangle requires the Integral image values of the four corners only, the feature calculation can be done in constant time. Figure 2

explains how we can calculate the sum of pixels inside a rectangle.

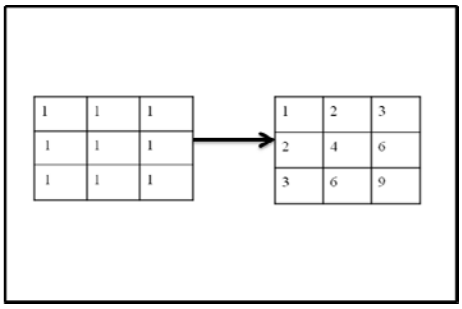


Fig. 1 A 3x3 image and its corresponding Integral Image

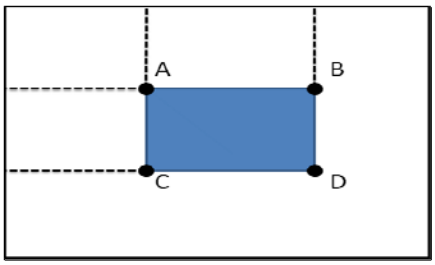


Fig. 2. Calculation of Sum of pixels in rectangle using Integral Image

$$\text{Sum (ABCD)} = \text{Int.Img(D)} - \text{Int.Img(B)} - \text{Int.Img(C)} + \text{Int.Img(A)} \quad (2)$$

The value of Int.img (A) is needed to be added because its being subtracted two times from Int.Img(D).

B. Feature

We used Haar-like features based on Viola and Jones face detection algorithm [1]. Features are represented as the rectangles, we have used features consisting two and three rectangles. Features used in our algorithm are show in figure 3. Feature value is the difference in the sum of pixels values of the rectangle in black region to the sum of pixel values of the rectangle in the white region.

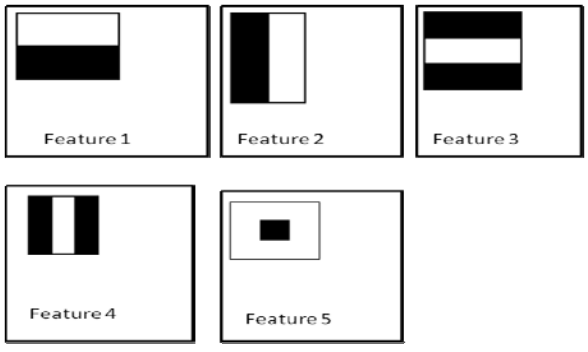


Fig. 3. Features Using In the algorithm

IV. PROPOSED ARCHITECTURE

Our proposed architecture uses both CPU and GPU for face detection. CPU main work is to grab the frame/image convert it into grey scale and copy it to the Graphics Processor.

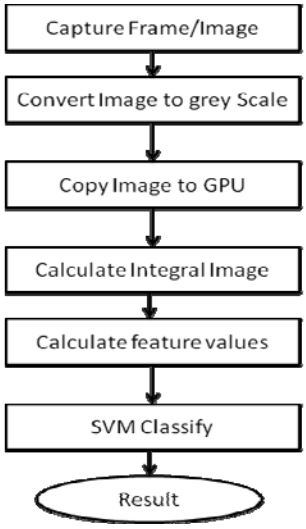


Fig. 4. Proposed Architecture for face detection

After that the high computation part i.e. the steps involving calculation of Integral Image and the feature extraction is done on GPU. Because of its very good performance in various machine learning problems SVM had become a very popular approach for face detection, but SVM classification was very slow making it unsuitable for real time face detection. Use of GPU for SVM classification removes the slowness constraints from the SVM and makes it suitable for real time face detection.

Our proposed architecture is shown in figure 4. We used GPUSVM [3] for both training and classification. Next two sections give a detail about the training phase and the classification phase of the algorithm.

A. *Training Phase*

Training was done using Labeled Faces in the Wild (LFW) database [5]. This database is chosen as it contains a large number of images in unconstrained environment. This database gives the true estimate of how an algorithm works in unconstrained environment. For training faces we randomly chose 2000 faces from LFW database, and for non-faces we created a database of 4200 non-face images. All sample images were then converted to grey scale and resized to 128x128 pixels. For feature calculation different sized rectangles were used for different features, the size of rectangle used for different feature are given in table 1. These different sized feature windows were used keeping in mind the size of eyes and all these features gives us a total of 510

features per images, which is way less than the total number of pixels for image.

Table 1. Size of rectangle Used for different features

FEATURE NO.	RECTANGLE SIZE
1	32x16
2	16x32
3	48x32
4	32x48
5	48x48

B. *Classification Phase*

Classification phase was carried out on frames grabbed from the webcam, and some of the randomly chosen images from LFW database. First we converted the image into grey scale, after that copied the grey scaled image to GPU. Then a detection sub-window of size 32x32 was run over the whole image. After processing the image with detection sub-window size of detection sub-window is increased by a factor of 1.5.Since we are using GPU for computation, we calculate features for all sub-windows in parallel and then use SVM for classifying the sub-window as face or non-face.

V. ALGORITHM AND ITS GPU IMPLEMENTATION

Proposed architecture for face detection is executed on Quadro FX580 GPU, and the language used is CUDA programming Language [6] an extension to C programming language. Algorithm for face detection is as follows:

1. Convert image into grey Scale.
2. Copy Image array to the GPU.
3. Calculate Integral Image.
4. For all Possible Sub-Window Sizes Do.
5. Create all sub-windows.
6. Assign each sub-window to different block.
7. For each sub-window Do.
8. Calculate feature for each sub-window in Parallel.
9. Classify the sub-window as face or non-face using SVM.
10. Send the result back to the CPU.
11. Draw rectangle over the detected sub-window.
12. End.

The main steps involved in the algorithm are Integral Image calculation, feature extraction and SVM classification [7]. Following sections provides a detailed GPU implementation of these three steps.

A. *Integral Image on GPU*

For parallel Integral Image calculation we have used the concept of Prefix sum. The Prefix sum is an operation on array in which each element in the resulting array is obtained from the sum of the elements in the operand array up to its index i.e. for input array $[a_0, a_1, a_2, a_3, \ldots, a_{n-1}, a_n]$ the output will be $[a_0, a_0+a_1, a_0+a_1+a_2, \ldots, (a_0+a_1+a_2\ldots a_{n-1}), (a_0+a_1+a_2\ldots a_n)]$. At any index K of the array the output will be $\sum_{t=0}^{k} a[t]$.

Integral Image calculation is divided into two steps.

1. Take each row of the Image in a temporary array and calculate its Prefix sum. Figure 5 shows state of the image after first step.

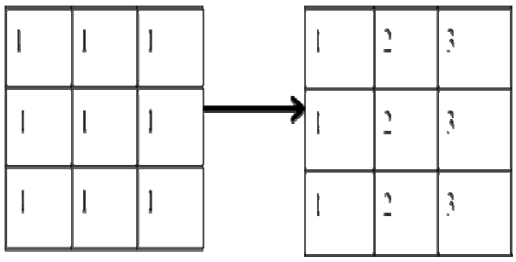


Fig. 5. Performing the row wise Prefix sum ($\sum_{t=0}^{k} a[j][t]$).

2. Take each column of the Image obtained after step 1 in a temporary array and calculate its Prefix sum. Figure 6 shows Final state of the image (i.e. Integral Image).

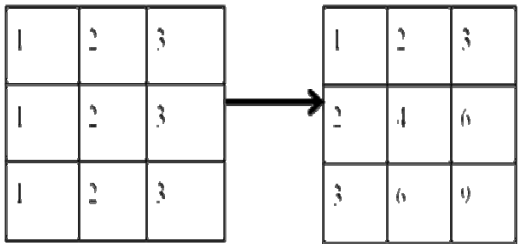


Fig. 6. Performing the column wise Prefix sum ($\sum_{t=0}^{k} a[t][j]$).

We first calculate step 1 in parallel by assigning each row to different block. Then step 2 by assigning each column to different block. After these two steps we get the Integral Image.

B. *Feature Extraction*

For feature extraction we first created all the sub-windows of a given size. Each sub-window is given an Identification Number (ID) denoted by the BlockIdx.x (i.e. X dimension of the blocks in CUDA kernel), BlockIdx.y(Y dimension of CUDA kernel) is used for calculation of features. The X (blockIdx.x) and Y (blockIdx.y) dimensions of the CUDA kernel are shown in Figure 7.

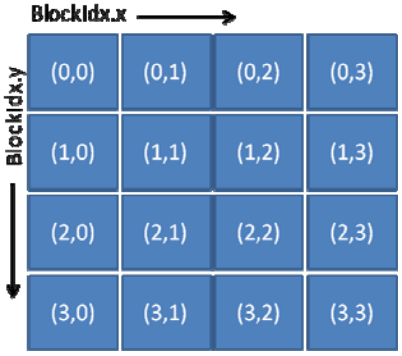


Fig. 7. X and Y dimensions of CUDA kernel

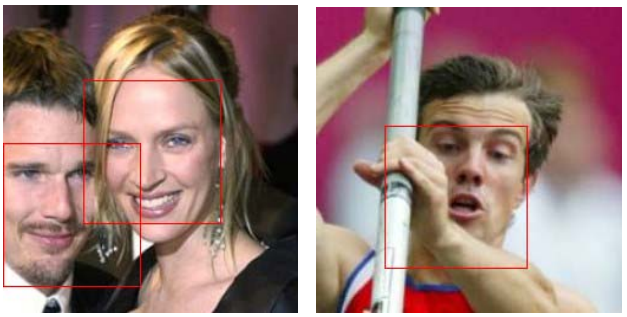
The obtained features were then given to SVM for classification. This process is repeated for all the possible sub-window sizes.

C. *Support Vector Machine*

Classification is done using GPUSVM [2]. GPUSVM provides the same functionality as that of LIBSVM [8], the most commonly used support vector machine for classification. GPUSVM takes data from file as input. We modified GPUSVM so that it can directly take the feature values from GPU memory as input and provide us the output in an array instead of the output file.

VI. RESULTS

For testing proposed architecture random images from the LFW database and the frames captured from webcam were used. Figure 8(a) shows face detected in image with two faces and figure 8(b) shows face detection in an image with partial occlusion.



(a) (b)

Fig. 8(a) Detection of two faces in an image. (b) Detection of face with partial occlusion

We obtained a high *correct detection rate* of 97.5% on LFW database. The *false positive rate* for our algorithm is 1.5%. Experimental results shows that we were able to attain a speed of 3.3FPS, which is almost twice the speed attained by OpenCV implementation.

Figure 9 provides the comparison between the time taken by CPU and time taken by GPU for face detection. Figure 10 gives the speed up obtained by using GPU for different image size during face detection.

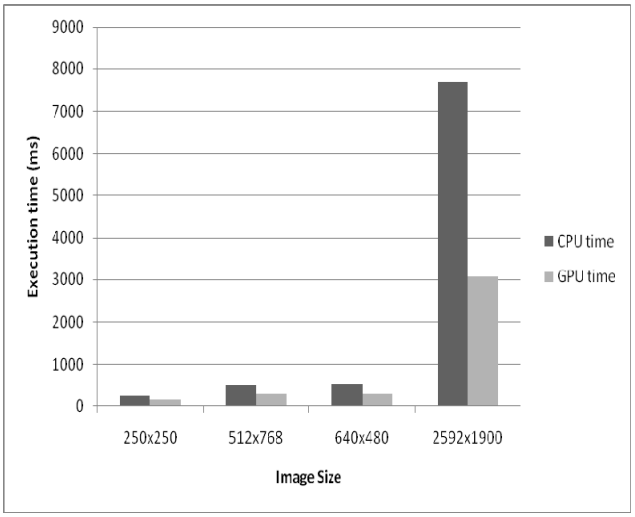


Fig. 9. Comparison of CPU execution time with GPU execution time

Fig. 10. Speed up of GPU execution time over the CPU execution time

VII. CONCLUSION

Section VI shows that our proposed architecture has high detection rate of 97.5%. Our architecture gives speed up of more than 1.6 for an image of size 250x250 which increases to more than 2.5 for image of size 2592x1900, which suggest that the speed up increases with the increase in the size of the input images. This makes our proposed GPU architecture suitable for high resolution images. Our implementation suggests that using Graphics processors, we can achieve a very high speed up for face detection. Results in Section VI suggest that with the use of high end GPUs it is possible to attain a high speed for face detection.

VIII. FUTURE WORK

Current algorithm works for frontal faces with high efficiency, it can be modified to work for images with side pose. There is need to incorporate some new features for side pose estimation in proposed algorithm. There is room for further optimization of the code developed. One of the main area of research is to fully optimize the use of GPU to attain much higher speed up. Using more than one GPU for processing is other area for future work. We can use a cluster of GPUs to process an image to attain a very high speed of detection.

REFERENCES

[1] Viola. P and Jones. M, "Robust *real-time face detection"*, International Journal of Computer Vision, vol. 2a.b., pp.137-154, 2004.

[2] GPUSVM. [Online].Available: http://www.cs.berkeley.edu/~catanzar/GPUSVM/ .

[3] J. P. Harvey, "Gpu acceleration of object classification algorithms using nvidia cuda," Master's thesis, Rochester Institute of Technology, Rochester, NY, Sept. 2009.

[4] Hefenbrock, D. Oberg, J. Nhat Thanh Kastner, R, Baden S.B. **"**Accelerating Viola-Jones Face Detection to FPGA-Level using GPUs**",** 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2010, pages 11-18.

[5] Gary B. Huang, Manu Ramesh, Tamara Berg, and Eric Learned Miller. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. University of Massachusetts, Amherst, Technical Report 07-49, October, 2007.

[6] NVIDIA CUDA Programming Guide Version 3.0, NVIDIA, Oct. 2010. [Online]. Available: http://developer.download.nvidia.com/compute/cuda/3_0/toolkit/docs/NVIDIA_CUDA_ProgrammingGuide.pdf.

[7] Waring, C.A.and Xiuwen Liu. "Face detection using spectral histograms and SVMs". IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol.35, pages: 467 – 476, June 2005.

[8] Chih-Chung Chang and Chih-Jen Lin, LIBSVM: a library for support vector machines, 2001. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm.