Parallel Implementation of Face Detection Algorithm on GPU

Aashna R. Bhatia
Birla Vishvakarma Mahavidyalaya,
Anand, Gujarat, India.
aashna_bhatia@yahoo.com

Narendra M. Patel
Birla Vishvakarma Mahavidyalaya,
Anand, Gujarat, India.
nmpatel@bvmengineering.ac .in

Narendra C. Chauhan
A.D. Patel Institute of Technology,
Anand, Gujarat, India.
narendracchauhan@gmail.co m

***Abstract* - The development of high resolution digital cameras for recording of still images and video streams has had a momentous influence on how communication and entertainment have developed during the recent years. Processing of human faces discovers many applications in various domains like law enforcement, security surveillance etc. A standard face processing system consists of face detection, face recognition, face tracking and face rendering. Face detection is one of the most premeditated areas of computer vision in image processing not only because of its thought-provoking nature but because of its broad continuum of applications which requires face detection as the first step. The formative Viola-Jones face detection algorithm is discussed in detail and what is the future prospectus of the face detection techniques. The focus is on the benefits and the drawbacks of the Viola-Jones and why it is the most prominent face detection algorithm and how it can be further improvised in order to meet the needs of today's time. To overcome the cons serial implementation of the Viola-Jones algorithm, the algorithm will be implemented on a parallel platform. The primary objective is to increase the computational speed of the algorithm which may be achieved through parallel implementation using CUDA and OpenCV on GPU (Graphics Processing Unit) and then give comparative analysis of the better computational results between the serial and parallel implementations.**

***Keywords*- face detection; Viola-Jones; CUDA; OpenCV; parallel processing.**

I. INTRODUCTION

The primary aspect of any face processing system is Face Detection. The speed and the accuracy of the human for the detection of the faces is still unmatched by the computerized face detectors. The researchers have been actively involved in the betterment of the face detectors which have a wide range of applications such as surveillance, law enforcement, information security etc. Face Detection sets the boundaries to the presence and the location of the face by differentiating the faces from all the other objects and the background present in the scene.

The basic steps to be performed by any face detector are described in *Figure1*.

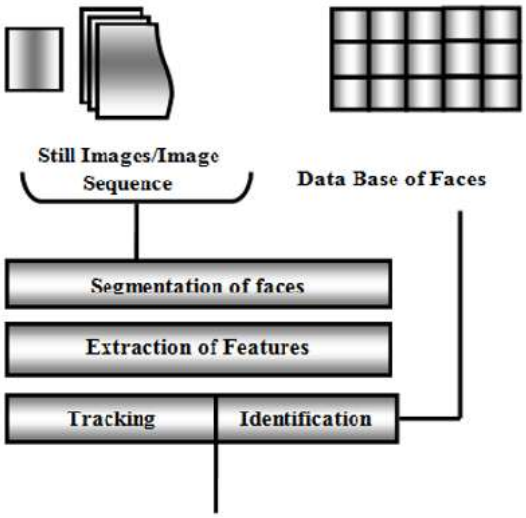


Figure 1 Face Detection Process [5]

The formative Viola Jones face detecting algorithm has been used to develop a face detector which is implemented serially as well as in parallel to get the best computational speed. The adaboost algorithm and the integral image have been implemented on parallel using OpenCV library.

II. VIOLA-JONES FACE DETECTION ALGORITHM

The Viola-Jones Face detection algorithm is one of the best face detection algorithms that have been developed through the time. The algorithm has been

divided into three phases, each of which helps the other part to improve the detection process. The three phases are;

A. Integral Image

An integral image is used for feature computation. It is an intermediate representation of an image which computes the rectangle feature rapidly by doing the summation of the pixels of an image at the very beginning to reduce the computational time. The features help in encoding the domain knowledge and above all feature operating systems work faster than the pixels.

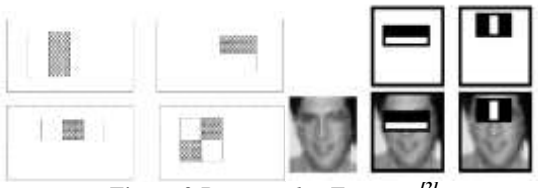


Figure 2 Rectangular Features [2]

The rectangle features are the difference between the summations of the white pixels from the black pixels [2]. There are three types of rectangle features two-, three- and four-rectangle features.

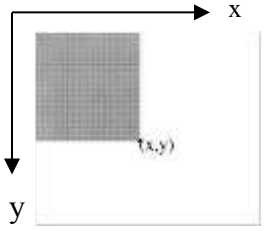


Figure 3 Integral image T (x,y) [1]

The integral image at a location(x,y) is the sum of the pixels above and to the left of (x,y) inclusive and it can be computed in a single pass and only once for each sub-window[3].

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

Where ii(x.y) is the integral image and i(x',y') is the original image. With reference to *Figure 3*, the integral image at point (x,y) is the sum of all the pixels above and to the left.

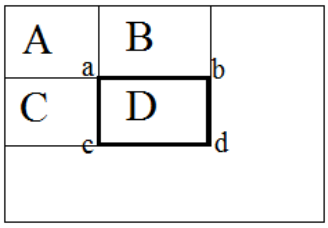


Figure 4 Example of Integral Image [2]

For example, [2] the integral sum inside rectangle D can be computed as:

***ii*(d) + *ii*(a) – *ii*(b) – *ii*(c)**

Where,

ii(a) = A
ii(b) = A+B
ii(c) = A+C
ii(d) = A+B+C+D
D = ii(d)+ii(a)-ii(b)-ii(c)

With the help of integral image, the summation of any rectangle can be computed using *four* array references as shown in *Figure 4,* while the difference between the two rectangles can be computed using *eight* array references, since two rectangular features are being used six array references can be used to compute the difference between the two rectangles.

B. Adaboost Algorithm

In a sub-window of size (24x24), there are almost 160,000 features that have to be computed which is computationally very expensive and also it is impractical to compute all these features [1,2,5]. Some of the features being irrelevant i.e. they do not contribute much in differentiating a face from a non-face, but these irrelevant features when combined can form a relevant feature which may help in modeling a face for detection.

Thus, an Adaboost is used to combine some of the irrelevant features that may help in forming a relevant feature that help in modeling and detecting a face from the scene. Adaboost being an iterative algorithm which considers each rectangle feature as a simple weak classifier. In each iteration, the algorithm tries to find these weak classifiers which may help in forming a strong classifier by combining several weak classifiers with lowest error.

A weak classifier is used as it doesn't give the best classification of the features but a combination of these features may help in building a strong

classifier. A linear combination of the weak classifiers is obtained at the end of each iteration.

C. Cascade Classifier

A cascade of classifiers is to be constructed in order to achieve increased detection performance while radically reducing the computation time. A smaller and more efficient cascade classifier is to be constructed which rejects maximum of the negative sub-windows while detecting the maximum of the true positive windows [2,5]. The simple classifiers are called up initially to reject the maximum of the false negative sub-windows and the complex ones are called at the later stage so that there is low false positive detection rate. The cascade designing depends on two factors.

- Number of layers in cascade (strong classifiers).
- Number of features in each strong classifier.

The stages of the cascade classifier are developed using the adaboost algorithm during the training phase by the training classifier. The cascade with gradually more complex classifiers will lead to better detection. The cascade classifier is used to achieve real timeliness by reducing the sub-windows that further need processing with a small number of operations [3,5].

1. Evaluate rectangle features.
2. Work out the weak classifier for each feature.
3. Combine the weak classifiers.

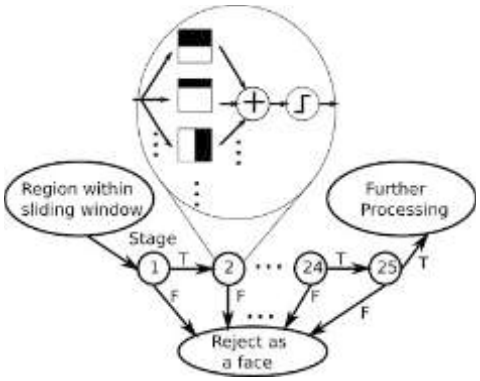


Figure 5 Cascade Classifier

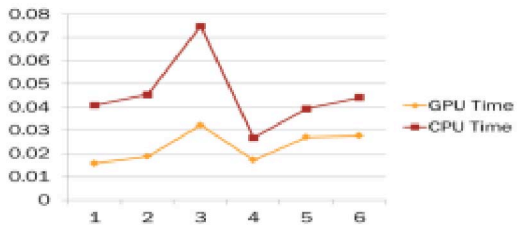
III. RESULTS AND DISCUSSION

The Viola-Jones algorithm has been implemented on the CPU as well as GPU to compare the results of the computational speed and the frames per second scanned per image.

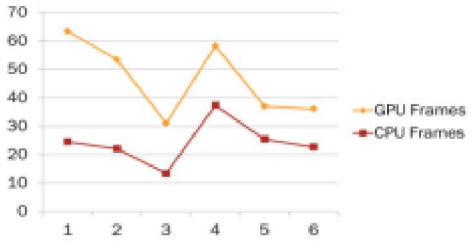
The CPU used for the implementation is Intel CORE i5 and to use the functionalities of the GPU (Nvidia GeForce 920m) CUDA v7.5 was used. OpenCv library is used as an interface between CUDA and the programmer so as to not use the CUDA language.

As per the implemented results it has been observed that GPU is better than the CPU in case of computational speed as well as for the frames per second that are calculated. The *Figure 7* shows the detected faces using the developed face detector.

The *Figure 6* shows the graphical results of the face detector. The results are shown for the few images that were used for the testing of the detector. With the help of the graph we can see that the processing time taken by the GPU is lower than that taken by the CPU. And the frames calculated by the GPU are also more than that of the CPU.



(a)



(b)

Figure 6 Graphical representations of the results. (a) Represents the graph of Processing time, where x-axis is the No. of images and y-axis is the Execution time in Seconds. (b) Represents the graph of Frames calculated during the Processing time where x-axis is the No. of images and y-axis is the No. of Frames scanned in Seconds.

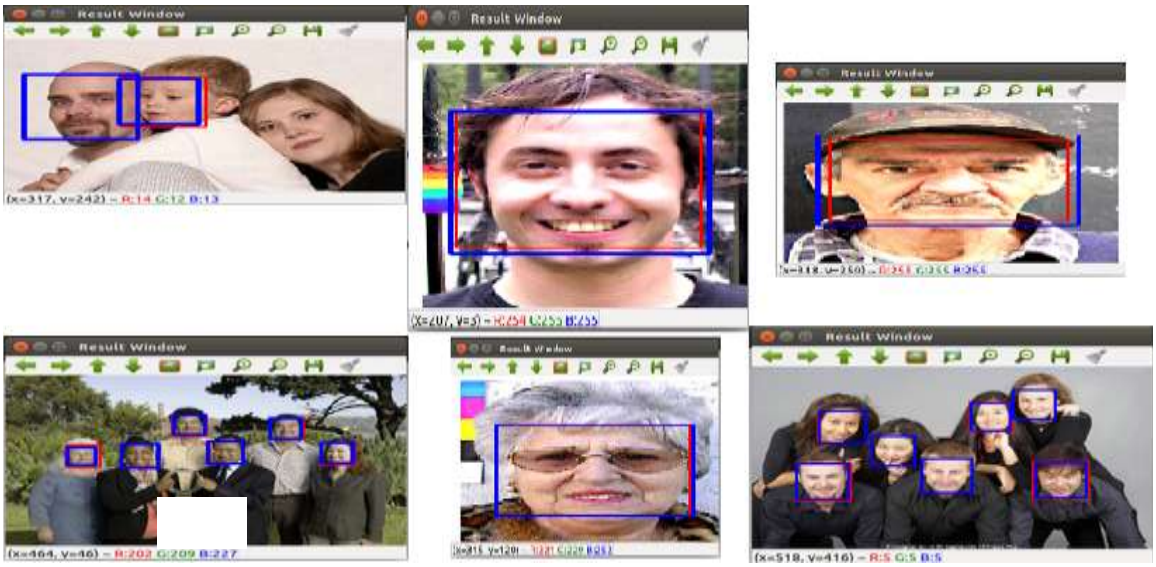


Figure 7 Detected Faces

IV. CONCLUSION

The proposed work concentrates on the methodology of the implementation of the Viola-Jones face detection algorithm using OpenCv; an open source library and its GPU module helps in using the properties of the GPU and the comparison of the outputs of the serial and parallel processing times of the respective algorithm.

The first function call is slow because of initialized overheads i.e. CUDA runtime API is called implicitly, the main processing time that is consumed is between the swapping of the processes to and from main and GPU memory. The main processing time that is consumed is between the swapping of the processes to and from main and GPU memory.

It works best in a constrained environment; the parallel implementation takes less processing time. Detection rate is more than the standard algorithm & it is practically better to use GPU for processing large data.

REFERENCES

[1]. Shivashankar J. Bhutekar, Arati K. Manjaramkar. "*Parallel Face Detection and Recognition on GPU",* IJCSIT (International Journal of Computer Science and Information Technologies), 2014.

[2]. Paul Viola, Michael J. Jones. "*Robust Real Time Face Detection".* IJCV (International Journal of Computer Vision), 2004.

[3]. Bharatkumar Sharma, Rahul Thota, Naga Vydyanathan, & Amit Kale. "*Towards a Real-Time Face Processing System using CUDA enabled Gpu's".* HiPC(High Performance Computing International Journal), 2009.

[4]. Yannick Allusse, Patrick Horain, Ankit Agarwal, and Cindula Saipriyadarshan. "*GpuCV: A GPU-Accelerated Framework for Image Processing and Computer Vision".* Springer, 2008.

[5]. Varsha Gupta, Dipesh Sharma. "*A Study of Various Face Detection Algorithms".* International Journal of Advanced Research in Computer and Communication Engineering, 2014.

[6]. Kristopher Reese, Yufeng Zheng and Adel Elmaghraby," *A Comparison of Face Detection Algorithms in Visible and Thermal Spectrums".* 2012.

[7]. Ole Helvig Jensen, "*Implementing the Viola-Jones Face Detection Algorithm,"* Technical University of Denmark, 2008.

[8]. Zhang, Cha Zhang and Zhengyou**.** "*A Survey of Recent Advances in Face Detection."* Microsoft Research, Microsoft Corporation, 2010.