Performance Estimation of Parallel Face Detection Algorithm on Multi-Core Platforms

Subhi A. Bahudaila and Adel Sallam M. Haider
Information Technology Department, Faculty of Engineering, Aden University. Aden, Yemen
sabhudail@yahoo.com, adel_ye@yahoo.com

Abstract

Parallel Viola-Joins face detection algorithm is achieved in irregular workload of parallel imbalance computation. In this paper the sequential and parallel algorithms of Viola-Joins face detection are presented and the performance of these methods are tested with CPU and GPU platforms. The scaling methods SmallerToGreater (STG) and GreaterToSmaller (GTS) are presented with sequential and parallel algorithms. Moreover, the execution time of the STG and GTS methods for both sequential and parallel algorithms are analyzed in this paper also. We found that with increasing the number of cores for face detection in the images for both STG and GTS methods, the speedup of parallel processing is increasing also, and the speedup gets good result with big size of images. Thus The face detection performance is higher in GPU method realization than in CPU. The experimental results show that the speedup of GPU architecture is the best over CPU.

Keywords: *Face detection, Parallel Viola-Joins, Cascaded classifier, integral image*, Haar *features*

1. Introduction

Face detection is a computer vision technology that determines the locations and sizes of human faces in arbitrary (digital) images. It is the basic mechanism for face recognition which is probably one of the most non-intrusive and user-friendly biometric authentication methods [1].Face detection is implemented in many of applications such as identification, human computer interaction (HCI), real-time recognition systems, digital cosmetics and many more. Different approaches are used in face detection:- finding face in images with controlled background, removing a plain monocolour background to get the face boundaries,method for face contour detection, entropy filtering, multi-scales ampling, and the Chan–Vese segmentation model[2].The main idea with these approaches is to utilize feature information to guide the final face segmentation step in an accurate manner. The Viola-Jones algorithm provides quick and robust face detection [3 and 4]. The parallel Viola-Jones algorithm is characterized as embarrassingly parallel of computationally intensive face detection of image operations [5]. The algorithm is widely implemented and deployed on diverse architectural platforms, such platforms are Field Programmable Gate Array (FPGAs), multi-core CPUs and many-core Graphical Processing Unit (GPUs).

The parallel programming becomes more portable than ever by using the Open CL standard. This is due to the APIs of Open CL that supports heterogeneous computing on both CPUs and GPUs platforms [6]. While APIs of CUDA supports the parallel computing on

GPUs platform [7]. Recently implementation of Viola-Jones face detection algorithm on GPUs is a most cost effective solution than FPGAs [8]. The streaming processors of a GPU supports a large amount of data parallelism inherent to Viola-Jones algorithm[9]. Viola-Jones face detection can be implemented and optimized with Open CL across the platforms [10]. Computer Vision software Library (Open CV) [11] contains the well-known Viola and Jones algorithm (embedded as the Cascade Classifier class) that helps to program generic object detection for applications. In this paper, we presented a strategy of parallel processing for accelerating Viola-Jones face detection algorithm, implementing on multicore of CPUs-GPUs and comparing the performance results. This paper is organized in following sections: Section 2 presents The Viola-Jones face detection. Section 3 presents the parallel algorithm face detection. The performance results are described in section 4. Finally, profiling parallelism is illustrated in Section 5.

2. Viola-Joins Face Detection

This paper describes the implementation of the Viola-Jones face detection algorithm by using sequential and parallel algorithms. The experimental results of the purposed idea are emphasized and presented also in the paper. The idea of the Viola-Jones algorithm is to scan a window of detecting faces across the image. The image processing divides the input image to pieces and then runs the fixed detector through these images. Viola-Jones runs the detector many times through the image – each time with different pieces. It has to require the same number of calculations whatever the pieces.

2.1. Haar Features Classifiers

Haar feature classifiers are used in the Viola-Jones algorithm to detect particular features of a face. They are represented as rectangles which are composed of 2, 3 and 4 rectangle features. Also they are used for particular classifiers [3]. The rectangles' weights and sizes are obtained from Open CV which uses Haar feature based cascade classifiers for object detection to track human front faces, with the support of a proper amount of training data [11]. The next step of this method is to calculate the integral image and simple types of Haar features.

2.2. Integral Image

The goal of integral image is to calculate the sum of image pixels in the original image, which is above and to the left of the concerned pixel as shown in Figure1. The sum of rectangle should be located in the bottom and to the right of the same rectangle.Another way to calculate the rectangle of the original image is to calculate the sum of the values of the integral image at the corners of a rectangle [12]. The sum can be computed as:

$$\boldsymbol{Sum = A - B - C + D} \quad \textbf{(1)}$$

Viola-Jones face detector uses sub-window, which is consisting of two or more rectangles. In this paper we used some types of features as shown in Figure. 2.The result of each feature can be calculated by subtracting the sum of the white rectangle from the sum of the black rectangle.

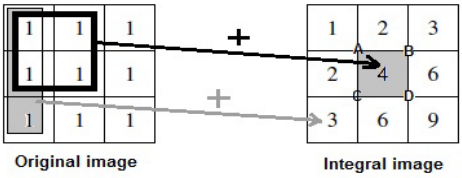


Figure 1 .The integral calculation of the image

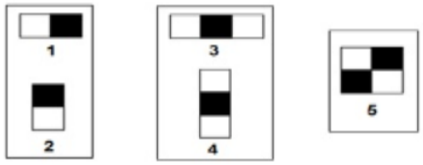


Figure 2. The different types of features

2.3. AdaBoost

AdaBoost is a machine learning boosting algorithm creates a small set of only the best features to create more efficient classifiers. The algorithm is much faster and required less data. It consists of a strong classifier through a weighted combination of weak classifiers. A weak classifier is mathematically described as:

$$h(x,f,p,\theta)=\begin{cases}1 & if\ pf(x) > p\theta \\ 0 & otherwise\end{cases} \quad (2)$$

Where x is a pixels' size of sub-window, f is the applied feature, p the polarity and the threshold that decides whether x should be classified as a positive (a face) or a negative (a non-face) [4]. Each of the features types are scaled and shifted across all possible combinations. Considering all the possible parameters of the Haar features like position, type and scale, approximately 160.000 possible features can be calculated within a detector for base size of sub-windows 24 * 24 pixels, but this is practically difficult achieved.

AdaBoost algorithm is built to help for finding the best features (weak classifiers) among all the features. The Viola-Jones uses AdaBoost algorithm to achieve better result of classification [12, 13, and 14].

2.4. The Cascaded Classifier

The image is scanned many times by this algorithm; if the image contains one or more faces, then it takes large amount of sub-windows. The cascaded classifier is consisted of stages of processing; every stage considers as a strong classifier. The function of every stage is to detect whether a given sub-window is not a face (the stage discards the image) or maybe a face (the stage accepts the image) and the process of detection is going to the next step.

More accepted stages means more chances for finding the faces. The cascaded classifier is illustrated with three stages in Figure 3.

2.5. The Scaling Method

For each size of sliding window the original image is scanned completely, from top to down and from the left to right. The weak classifiers are calculated for current window and collected into strong classifier, which shows if there is an object at current window. Thus the sliding window moves to the next position of the original image and every type of Haar features moves within sliding window to detect the weak classifier. This procedure is repeated with new size of sliding window, which is changed by given factor. This iteration is continued until finished with all sizes of windows [15].

In this paper we presented the scaling method, which is resizing the new sliding window from greater size to smaller size (GTS) for detecting big faces, and from smaller size to greater size (STG) for detecting small faces.

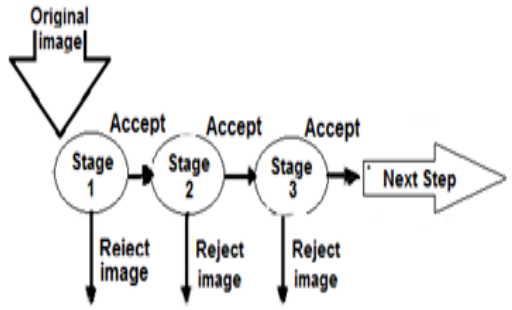


Figure 3.The Cascade classifier

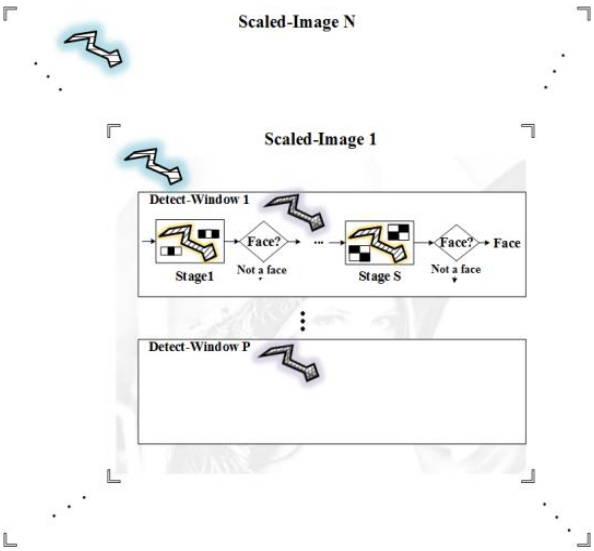


Figure 4. Level of parallelization of Viola Jones

3. Parallelization of Viola-Jones Face Detection (VJFD)

3.1. Parallelization of VJFD on Multi-Core of CPU

The Viola-Jones Face Detection Algorithm (VJFDA) can be parallelized in three levels of parallelization: feature level parallelism; cascade level parallelism; and image level parallelism as shown in Figure 4 which shows the implementation of multithreaded processing.

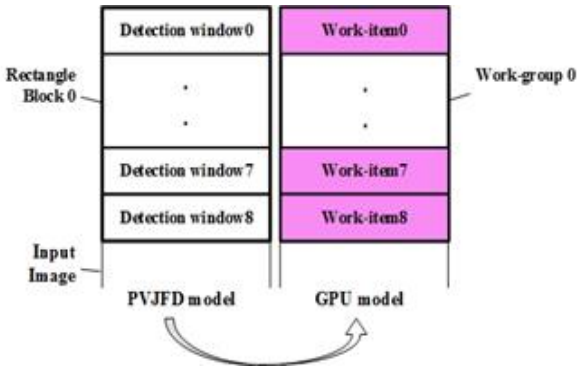


Figure 5 . Parallelizing VJFDA on GPU.

Threads-Groups are assigned in the levels of parallelism as follows:-

(A)- Multiple threads of Feature level parallelism (Threads FLP), the threads of this group work independently for feature computing in each classifier stage and synchronize to obtain the sum of all features in a same stage;

(B)- Multiple threads of Cascade level parallelism (Threads CLP), the threads of this group work independently for the computation space of face detection in each detection window without any synchronization;

(C)- Multiple threads of Image level parallelism (Threads ILP), the threads of this group work in parallel independently for detecting all daces with different size in each scale of the input image.

3.2. Parallelizing VJFDA on GPU

The independent workload of Viola-Jones face detection algorithm is inherently well-suited to Single Instruction- Multiple Data (SIMD) architecture. This is due to its computationally intensive of thousands of features over many window scales as shown in Figure 5 which shows the work-loads that are assigned to all threads (work-items) in each group (workgroup). Each thread works on exactly one detection window.

The sequential and parallel Viola-Jones algorithms are shown in the Figure 6 and Figure 7. These algorithms are tested and implemented in the environment Microsoft Visual Studio (C#).

```
1.  //for each scaling step
2.  for fector = 1 to min (fector * scalefector) do
3.      setClassifier(fector)
4.      resize_Scaled_Window(image, factor)
5.      compute_Integral_Image(integralImage, image)
6.      //for every pixel in the window column
7.      for y = 0 to image_hight (y + step) do
8.          //for every pixel in the window row
9.          for x = 0 to image_width (x + step) do
10.             //try to detect an object inside the window
11.             detect(x,y,integralImage)
12.             { for i = 0 to stage_num do
13.                 detect_at(x,y)
14.                 if sum < threldhold(stage) then
15.                     break;
16.             endfor
17.             if i = stage_num then
18.                 //an object has been detected
19.                 face.push(x,y) }
20.         endfor
21. endfor
```

Figure 6.The sequential Viola-Jones Algorithm

```
1.  //for each scaling step
2.  for fector = 1 to min (fector * scalefector) do in parallel
3.      set_Classifier(fector)
4.      par_resize_Scaled_Window(image, factor)
5.      par_compute_Integral_Image(integralImage, image)
6.      //for every pixel in the window column
7.      for y = 0 to image_hight (y + numstep) do in parallel
8.          //for every pixel in the window row
9.          for x = 0 to image_width (x + step) do
10.             //try to detect an object inside the window
11.             detect(x,y,integralImage)
12.             { for i = 0 to stage_num do
13.                 par detect_at(x,y)
14.                 if sum < threldhold(stage) then
15.                     break;
16.             endfor
17.             if i = stage_num then
18.                 //an object has been detected
19.             face.push(x,y) }
20.         endfor
21. endfor
```

Figure 7. The Parallel Viola-Jones Algorithm

4. Performance Analysis

In this paper the experiments are processed for the sequential(uni-core) and parallel (multi-core) processing algorithm of Voila Jones on multiple cores platforms. Five images with various multiple faces are used also in the experiments, as well as the size of these images is given also with various amounts of pixels.

The test is passed twice for every image, one for the image integral scaling method smaller to greater, the other for greater to smaller of the sequential and parallel algorithms. In this section, the accuracy and performance comparisons of Voila-Jones algorithm are implemented for the experimental works.

4.1. Experimental Setup

The experiment has configured in order to obtain the highest performance. The parallelism of CPU and GPU is exploited in the experiments. The CPU's cores are important for parallel processing with naive overheads, while the GPU's CUDA cores are more important in image processing; e.g. the faces detection performance is higher in the GPU implementation than that in CPU due to hundreds of cores. These specifications are described in Table 1.

Table (1). Experimental resources data sheet

Resource	Specification	Cores	RAM
CPU	Intel® Core™2 Quad CPU Q9400	4	4 GB
GPU	nVidiaGeforce 840M GPU	384	4 GB

Table (2). Experimental setup of images

ImageID	Image Size	Number of Faces
A	3008x2008	21
B	2272x1704	8
C	1280x760	12
D	1024x575	3
E	448x300	20

The experiments used Microsoft Visual Studio Ultimate 2012 IDE with Microsoft.NET Framework Version 4.5.50709, and OpenCV library, which used for face detection programming in the operating system of Windows 8 Pro 64-bit. The profiling parallelism is implemented by using the Concurrency Visualizer of Microsoft Visual Studio Ultimate 2012.Also the detection search mode "No Overlap", the exclusion of the best determined detection, is used for these experiments. The images of multiple faces are classified in five categories as they are described in Table 2.

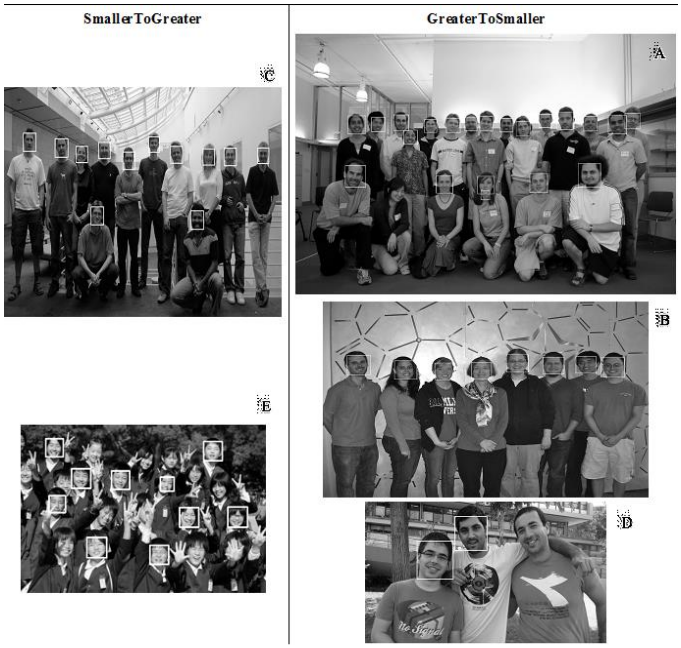


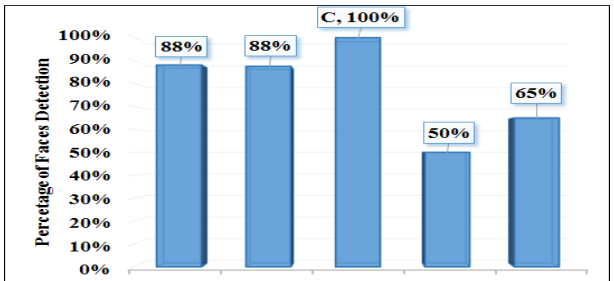
Figure 8. Accuracy images results of the scaling methods.

Table 2 shows the different sizes of images and the number of faces in each images. These sizes are arranged as follows: Large A, Middle-Large B, Middle C, Small-Middle D, and Small E for images sizes. Both the scaling methods are used for scaling detecting window of each image. The method GTS gives a better result with detecting faces in the images A, B and D. Whereas the STG method is better with the images C and E.

4.2. Results and Dissection

1) The accuracy results: The accuracy results are obtained by comparing the scaling methods of (S2G) with (G2S) of sequential and parallel VJFDA. The best accuracy results of the scaling methods are shown in Figure8.

Figure 8 shows that the scaling method GTS is the best for detecting faces in the images of class A, B and D due to the big faces in the images. On the other hand the scaling method STG is the best for detecting faces in the images of class C and E due to the small faces.



Table(3). The result of face detection

accuracy of the images.

ImageID	Sequential		Parallel	
	S2G	G2S	S2G	G2S
A	18	19	18	19
B	6	8	6	8
C	12	12	12	11
D	1	2	1	2
E	15	11	15	11

Figure. 9. The detection faces of the scaling methods in sequential VJFDA.

Table 3 gives the number of detected faces for each scaling method in the sequential and parallel VJFDA of each image. The result of comparing sequential with parallel VJFDA is not different. The representation of Table 3 for detecting faces of the scaling methods in sequential VJFDA is shown in Figure 9.

2) The performance results: This section discussed the performance comparison results of the parallel and sequential execution time of the Voila-Jones algorithm with two scaling methods. The results are taken by the average of 10 running times on every class of the five images.

The ratio of the execution time for sequential and parallel VJFDA is given by Equations 3 and 4, which are implemented in Table 4 and Table 5.

$$Tseq\% = \frac{Tseq}{Tseq + Tpar}\ 100 \quad (3)$$

$$Tpar\% = \frac{Tpar}{Tseq + Tpar}\ 100 \quad (4)$$

Where: Tseq% is percentage of the sequential execution time, Tpar% is percentage of the parallel execution time.

The parallelization results as shown in the next figures are very important in the face detection implementations for the various sizes of images. These are due to the integral image of every Haar Feature calculation in constant time, which expends a long execution time.

Table(4). The result of face detection accuracy of the images

ImageID	Ts[ms]	Tp[ms]	seqS2G	parS2G
A	72464	19263	79.00	21.00
B	43246	11732	78.66	21.34
C	10230	2927	77.75	22.25
D	6010	1782	77.13	22.87
E	1220	433	73.81	26.19

Table(5). The result of face detection accuracy of the images

ImageID	Ts[ms]	Tp[ms]	seqS2G	parS2G
A	35520	10601	77.01	22.99
B	14409	4479	76.29	23.71
C	3531	1184	74.89	25.11
D	4550	1542	74.69	25.31
E	525	210	71.43	28.57

The performance comparison percentage between the sequential and parallel execution time of different images with the scaling methods STG and GTS are shown in Figure10 and Figure11respectively. From these figures we can conclude that the parallel execution time of

images processing for detecting faces is better than sequential with the difference value about 42\%.

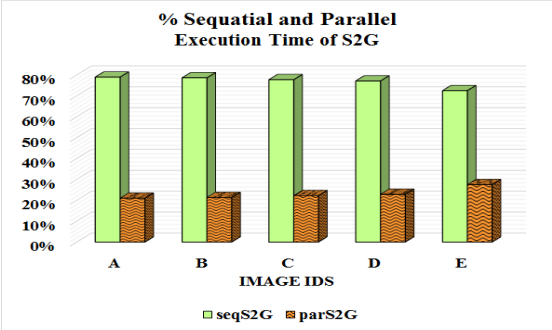


Figure. 10. The percentage of Sequential and parallel execution time of VJFD Algorithm with scaling method STG.

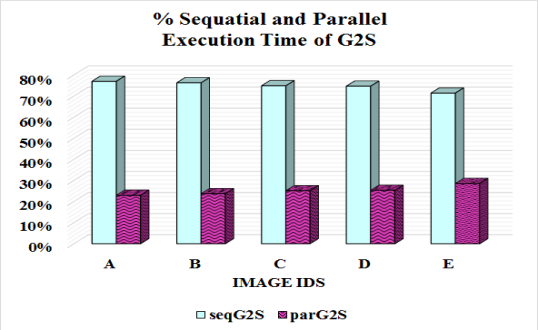


Figure. 11. The percentage of Sequential and parallel execution time of VJFD Algorithm with scaling method GTS.

The comparison between two methods GTS and STG for sequential and parallel VJFDA are shown in the Figure12 and 13 respectively. From these figures we can conclude that the GTS method has the less execution time than the STG method for both sequential and parallel VJFDA.

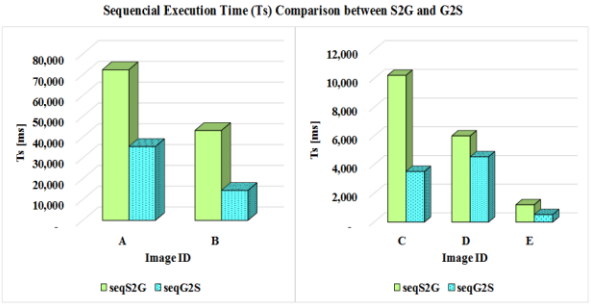


Figure. 12. Comparison between the Sequential execution time of VJFD scaling method (STG and GTS).

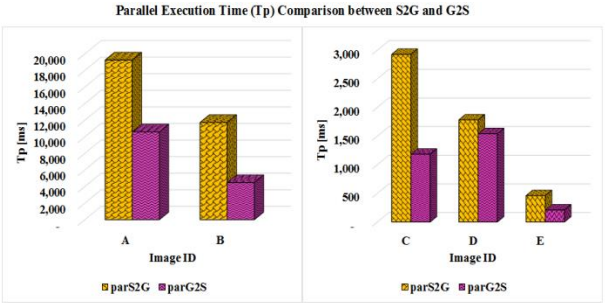
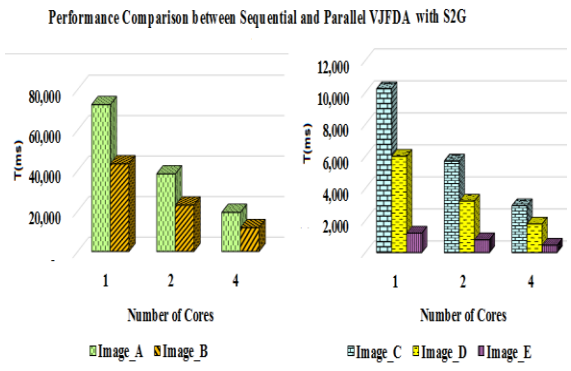


Figure. 13. Comparison between the parallel execution time of VJFD scaling modes (STG and GTS).

Also the result illustrates the relationship ratio between the sequential and parallel execution time with different sizes of the images. Thus with increasing the dimensions of images, the benefit of parallel processing is increasing also.



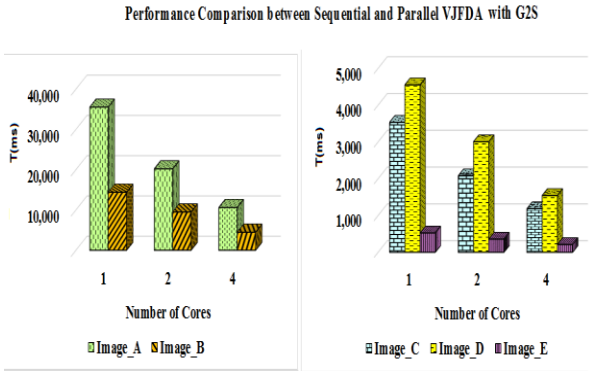


Figure. 14. Performance comparison between the sequential (one core) and parallel execution time of VJFDA. with scaling method (STG).

Figure. 15. Performance comparison between the sequential (one core) and parallel execution time of VJFDA with scaling method (GTS).

Figure14 and Figure15 illustrate the performance of the cores; it is clear that with increasing the number of cores for face detection in the images for both STG and GTS methods, the execution time is enhancing rapidly.

Figure16 and Figure17 illustrate the performance speedup with both STG and GTS methods by using multi-core parallelization; the speedup of images A and B get good result than the others. This result is obtained by dividing the sequential execution time by parallel execution time for every class. Again we can conclude that with increasing the number of cores for face detection in the images for both STG and GTS methods, the speedup of parallel processing is increasing also, and the speedup gets good result with big size of images.

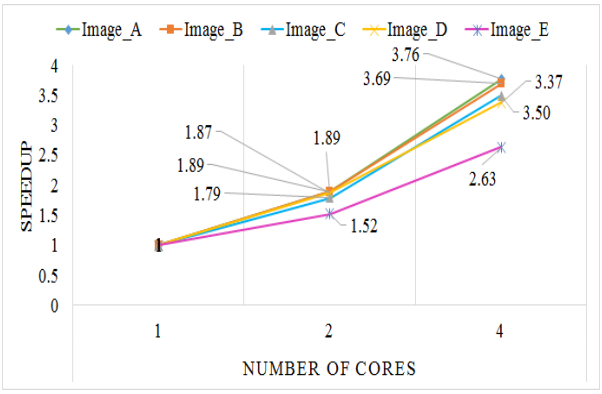


Figure. 16. Performance comparison (Speedup) of VJFDA. with scaling method (STG).

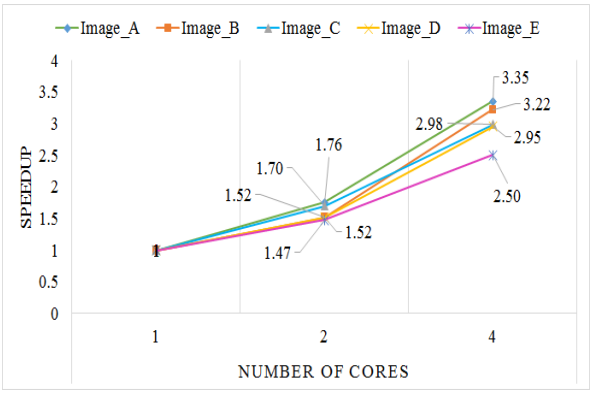


Figure. 17. Performance comparison (Speedup) of VJFDA. with scaling method (GTS).

Figure18 shows the speedup of both scaling methods on 4 CPU cores. In this figure the class A gets the value about 3.76 with scaling method STG, this value is near to the ideal speedup of 4 CPU cores. Whereas the speedup value for GTS is about 3.35. This result is depending on the difference between the sequential and parallel execution time for each class.

More optimization of the performance of VJFDA is the use of the GPU parallelism. Its performance result is compared with the result of both CPU performance of the sequential and parallel VJFDA as shown in Figure19. This result illustrates the rapid Voila-Jones with the highest performance of the GPU implementation. The figure shows the comparison of the methods sequential, parallel and GPU for STG method. It is clear that the result shows the advantage of using the GPU techniques over the other.

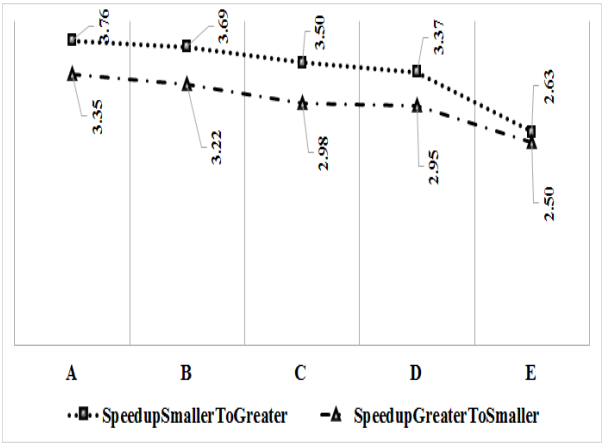


Figure. 18. Parallelization Comparison between the speedup of GTS and STG on 4 CPU cores.

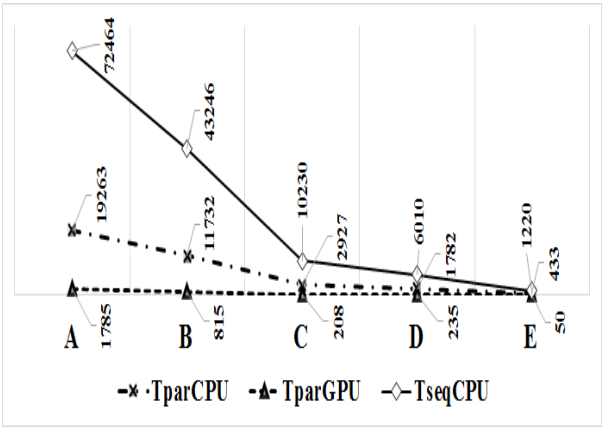


Figure. 19. Parallelization Comparison for sequential, 4 CPU cores and GPU Execution Time.

4.3. Profiling Parallelism

1) **Multi-Core Parallelization Profiling:** The profiling of Multi-Core parallelization is illustrated by using the performance resource monitor of each CPU in the multi-core platforms as shown in Figure20. The total result of the performance for the sequential processing is showing the big execution time than the others, whereas the performance of 2 CPU cores greater than 4 CPU cores. Thus the performance is enhancing with increasing the number of CPU cores.

2) **Multi-Threading Parallelization Profiling:** The information of Profiling Parallelism is obtained by using the Concurrency Visualizer of .Net Visual Studio 2012 IDE as shown in Figure21. This information is obtained by getting the performance comparison results of detecting multiple faces of the image E. The profiler shows the maximum utilization of the CPU's cores (=4) in high performance of the parallel version, while it is poor utilization (only one core works) in the sequential version.

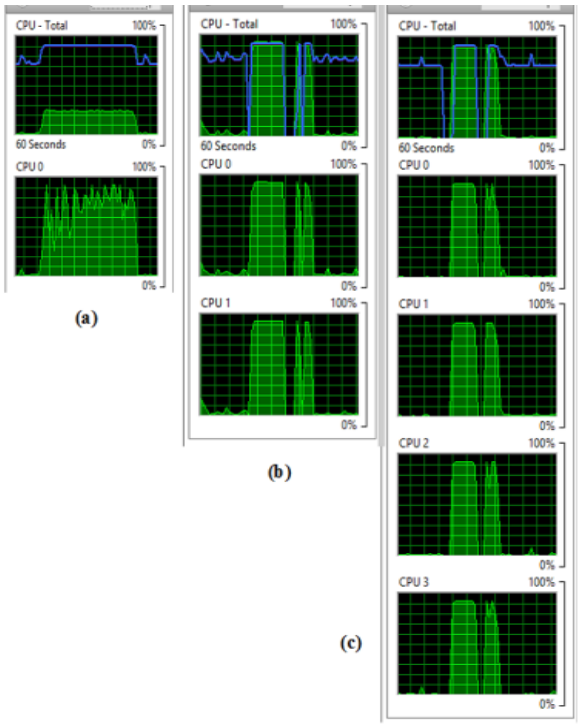


Figure. 20. Multi-Core Parallelization Profiling (a) sequential, (b) 2 CPU cores and (c) 4 CPU cores

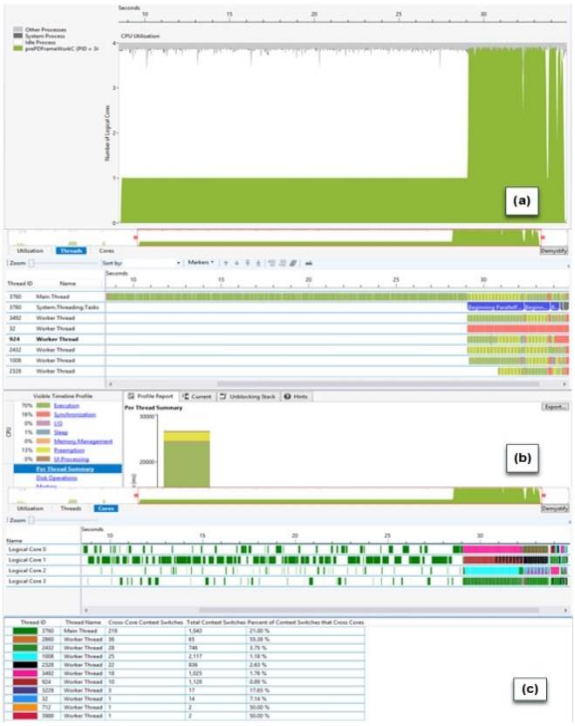


Figure. 21. Profiling parallelism of the Concurrency Visualizer

5. Conclusion

This work presented the Viola-Jones face detection algorithm on sequential, multi-core CPUs and GPUs for GTS and STG methods. The scaling method of GTS is the best for detecting big faces in the images. Whereas, the STG is the best for detecting small faces. The accuracy of detecting faces in the images for sequential and parallel algorithms is not different. The parallel execution time of images processing for detecting faces is better than sequential with the difference value about 42%. This conclusion leads to the need of using the parallel processing for the images.

The GTS method has the less execution time than the STG method for both sequential and parallel VJFDA and with increasing the dimensions of images, the benefit of parallel processing is increasing also. With increasing the number of cores for face detection in the images for both STG and GTS methods, the speedup of parallel processing is increasing also, and the speedup gets good result with big size of images. This speedup is depending on the difference between the sequential and parallel execution time for every class. The ideal speedup of 4 CPU cores is 4.In this work we have got value about 3.76 with scaling method STG.

Moreover, this paper also presented the highest performance of the GPU implementation, and found that the using of GPU techniques is better than others methods. The profiling of Multi-Core parallelization is showing that the performance is enhancing with increasing the number of CPU cores.

References

[1] S. Y. Kung, M. W. Mak, S. H. Lin, "Biometric Authentication: A Machine Learning Approach", Prentice Hall PTR (c) 2004.

[2] Xiaohua Shia and others, "Parallelization of a color-entropy preprocessed Chan–Vese model for face contour detection on multi-core CPU and GPU", Parallel Computing Journal 49, 2015,pp 28–49.

[3] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR 2001, pp. 511–518.

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

[5] Barry Wilkinson, Michael Allen, "Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers 2nd E", Pearson Prentice Hall © 2005.

[6] Khronos Open CL Working Group. The Open CL specificationV1.2. November 2011.

[7] CUDA C Programming Guide, PG-02829-v7.5 | (c) 2015 NVIDIA, available at http://www.nvidia.com.

[8] D. Hefenbrock, J. Oberg, N.T.N. Thanh, R. Kastner, S.B. Baden, "Accelerating Viola–Jones face detection to FPGA-level using GPUs," in: Proceedings of the 2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM'10, IEEE Computer Society, Washington, DC, USA,2010, pp. 11–18.

[9] Haipeng Jia, Yunquanzhang, Weiyan Wang, Haipeng Jia and Jianliang Xu, "Accelerating Viola-Jones Facce Detection Algorithm On GPUs," 14th International Conference on High Performance Computing and Communications, (c) 2012 IEEE.

[10] Weiyan Wang1 and others, "Parallelization and Performance Optimization on Face Detection", Tsinghua Science and Technology, June 2012, 17(3): 287-295.

[11] "Open CV Cascade Classification", available at http://wwww.docs.opencv.org/modules.

[12] Yi-Qing Wang, "An Analysis of the Viola-Jones Face Detection Algorithm", Image Processing On Line, 4, 2014, pp. 128-148.

[13] Alpika Gupta, RajdevTiwari, "Face Detection Using Modified Viola Jones Algorithm", International Journal of Recent Research in Mathematics Computer Science and Information Technology Vol. 1, Issue 2, October 2014 – March 2015, pp: 59-66.

[14] K. T. Talele, S. Kadam, A. Tikare, "Efficient Face Detection using Adaboost", IJCA Proc. on International Conference in Computational Intelligence, 2012.

[15] ReinhardKlette, "Concise Computer Vision, An Introduction into Theory and Algorithms", Springer-Verlag, London,2014, pp.429.