turnitin

Performance_Analysis_of_GPU-CPU_forThe_Face.pdf
Oct 21, 2021
3977 words / 19661 characters

dolly indra

Performance_Analysis_of_GPU-CPU_forThe_Face.pdf

Sources Overview

12%

OVERALL SIMILARITY

1	dugi-doc.udg.edu INTERNET	2%
2	Stan Z. Li. "Face Detection", Handbook of Face Recognition, 2005 CROSSREF	1%
3	Berlian Al Kindhi, Noviyanti Susanto, Wuri Handayani, Septiana Vera Kurniasari, Afriliya Putri Pratama. "Prediction of the Tuberculosis Pati... CROSSREF	1%
4	University of Essex on 2021-08-31 SUBMITTED WORKS	1%
5	Marquette University on 2018-11-13 SUBMITTED WORKS	1%
6	Xiebing Wang, Kai Huang, Alois Knoll. "Performance Optimisation of Parallelized ADAS Applications in FPGA-GPU Heterogeneous Syst... CROSSREF	<1%
7	Mehreen Naeem, Muhammad Jawad Khan, Talha Yousf. "Eyeball Identification and Tracking using Digital Image Processing", 2021 Inte... CROSSREF	<1%
8	Universitas Siswa Bangsa Internasional on 2018-01-31 SUBMITTED WORKS	<1%
9	eeccis.ub.ac.id INTERNET	<1%
10	wwww.easychair.org INTERNET	<1%
11	journal.unika.ac.id INTERNET	<1%
12	Netaji Subhas Institute of Technology on 2015-10-29 SUBMITTED WORKS	<1%

Excluded search repositories:

- None

Excluded from Similarity Report:

- Small Matches (less than 25 words).

Excluded sources:

- Dolly Indra, Ramdan Satra, Muhammad Jamil Asshiddiq, Lukman Syafie, Erick Irawadi Alwi, Muhammad Arfah Asis. "Performance Analysis of GPU-CPU for The Face Detection", 2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT), 2021, Crossref, 82%

2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT)
April 09-11. 2021, ISTTS Surabaya, Indonesia

Performance Analysis of GPU-CPU for The Face Detection

1st Dolly Indra
Faculty of Computer Science
Universitas Muslim Indonesia
Makassar, Indonesia
dolly.indra@umi.ac.id

2nd Ramdan Satra
Faculty of Computer Science
Universitas Muslim Indonesia
Makassar, Indonesia
ramdan@umi.ac.id

3rd Muhammad Jamil Asshiddiq
Faculty of Computer Science
Universitas Muslim Indonesia
Makassar, Indonesia
muhjamil.asshiddiq@umi.ac.id

4th Lukman Syafie
Faculty of Computer Science
Universitas Muslim Indonesia
Makassar, Indonesia
lukman.syafie@umi.ac.id

5th Erick Irawadi Alwi
Faculty of Computer Science
Universitas Muslim Indonesia
Makassar, Indonesia
erickirawadi.alwi@umi.ac.id

6th Muhammad Arfah Asis
Faculty of Computer Science
Universitas Muslim Indonesia
Makassar, Indonesia
muh.arfah.asis@umi.ac.id

2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT) | 978-1-6654-0514-0/20/$31.00 ©2021 IEEE | DOI: 10.1109/EIConCIT50028.2021.9431907

***Abstract*—The face detection process is one of the research objects in the field of image processing which aims to recognize faces from an image. Performance and face detection accuracy can be measured based on the platform used. In this research, we propose a face detection approach using the Viola-Jones algorithm to perform face detection that is applied to CPU and GPU programs utilizing the architecture of the Compute Unified Device Architecture (CUDA). The image used in this study is an image in RGB format with 25 different image resolution sizes. Test results for each platform are the detection processing time on the CPU is 1433048.72 ms with an accuracy of 90% while the GPU is 233374.583 ms with 72% of accuracy.**

***Keywords*—Face Detection, Viola-Jones, Haar Cascade, Parallel Computing, CUDA**

I. Introduction

In recent years, the development of computer technology, especially in the field of multimedia for face recognition, has been widely used as an object of research and development. Face recognition technology is one of the most popular research in the field of face pattern recognition and face image processing. Face detection is the first step in the face recognition process that can help identify a face by determining the location of the face and pre-processing the face image.

Central Processing Unit (CPU) is a type of hardware used to perform in the face detection process. CPUs generally have a sequential system where the output conditions depend on the previous input or in other words a task is done in a gradual way. The use of this sequential system is less efficient which is caused by which is caused by the process is carried out in stages and certainly greatly affects the timing of the face detection process [1]. The CPU is the core hardware of the computer where its function is the execution of a command or instruction from software. CPU consists of two main logic components namely Datapath which executes arithmetic operations and control which will tell Datapath, memory, and what input and output (I / O) devices to do based on instructions from a program [2].

In recent years, the Graphics Processing Unit (GPU) has been widely used as computing resources like CPU. Nowadays, GPU is not only used for rendering 2D and 3D graphics but also for other activities especially in parallel computing. Compute Unified Device Architecture (CUDA) is a technology developed by NVIDIA to increase computing capabilities by accelerating computation intensively [3].

The architecture used to utilize the GPU in parallel computing is the Compute Unified Device Architecture (CUDA). The CUDA architecture allows software developers to create programs that run on NVIDIA's GPUs with a syntax like the familiar C syntax. The CUDA architecture consists of three basic parts which help system developers making efficient use of the computing capabilities of the graphics card. The CUDA divides devices into grids, blocks, and threads in a hierarchical structure. With several threads in one block and the number of blocks in one grid and several grids in one GPU so that parallel processing is achieved using a very large hierarchical architecture [4].

The face detection process is a complex process that will consume time, especially when the resolution of the face image used for very large. In recent years, graphics cards have been widely used because the improvement computing performance compared to CPUs. This problem can be solved by using GPU. Complex processes such as face detection can be done in parallel so that processes can be completed more quickly and efficiently than the CPU. In research by Jain & Patel with the research title "A GPU based implementation of Robust Face Detection System" showed that the face detection process used the GPU faster 5.41 ms to 19.75 ms than the CPU. However, this research only uses 1 parameter, namely the time of the detection process [5].

Parallel Computing is the simultaneous use of multiple computing resources to solve a computing problem, so that the benefits of Parallel Computing are maximized then a task will be divided into several sub-tasks so that in a process it can be divided into other processor cores to perform computations simultaneously [6]. The Viola-Jones algorithm is an algorithm created by Paul Viola & Michael Jones in 2001. The detection process by classifying an image based on the feature value through a classifier that is formed from the training data [7]. This algorithm can be implemented on a wide range of small low power devices, including hand-held devices and embedded processor [8].

Research related to the acceleration processing algorithm Viola-Jones has done a lot of them by Vikram Mutneja et al with the title is Modified Viola–Jones algorithm with GPU accelerated training and parallelized skin colour filtering-

is caused by which is caused by the process is carried out in stages and certainly greatly affects the timing of the face detection process [1]. The CPU is the core hardware of the computer where its function is the execution of a command or instruction from software. CPU consists of two main logic components namely Datapath which executes arithmetic operations and control which will tell Datapath, memory, and what input and output (I / O) devices to do based on instructions from a program [2].

In recent years, the Graphics Processing Unit (GPU) has been widely used as computing resources like CPU. Nowadays, GPU is not only used for rendering 2D and 3D graphics but also for other activities especially in parallel computing. Compute Unified Device Architecture (CUDA) is

benefits of Parallel Computing are maximized then a task will be divided into several sub-tasks so that in a process it can be divided into other processor cores to perform computations simultaneously [6]. The Viola-Jones algorithm is an algorithm created by Paul Viola & Michael Jones in 2001. The detection process by classifying an image based on the feature value through a classifier that is formed from the training data [7]. This algorithm can be implemented on a wide range of small low power devices, including hand-held devices and embedded processor [8].

Research related to the acceleration processing algorithm Viola-Jones has done a lot of them by Vikram Mutneja et al with the title is Modified Viola–Jones algorithm with GPU accelerated training and parallelized skin colour filtering-

978-1-6654-0514-0/21/$31.00 ©2021 IEEE

based face detection [9]. Then by Narmada Naik and G.N Rathna with the title is Robust Real Time Face Recognition and Tracking on GPU Using Fusion of RGB and Depth Image [10]. Then by Jan Masek et al with the title is Speeding up the Viola-Jones Algorithm using Multi-Core GPU Implementation. Then Haipeng Jia et al with the title is Accelerating Viola-Jones Face Detection Algorithm on GPUs. Then an efficient GPGPU based implementation of face detection algorithm using skin color pre-treatment by Guerfi et al. Then an Efficient GPGPU based Implementation of Face Detection Algorithm using Skin Color Pre-treatment by Imene Guerf et al [11]–[14]. Based on previous research, we conducted further research using GPU Nvidia GTX 1050 Ti technology and measured the accuracy and processing speed of the Viola Jones algorithm then compared the results with the measurement using the CPU.

II. METHOD

Viola-Jones Algorithm is an algorithm created by Paul Viola & Michael Jones in 2001. The detection process is the classification process of an image based on the feature value through a classifier that is formed from training data. This algorithm uses the concept of the Haar feature, Integral Image, AdaBoost which is then processed into a Cascade Classifier. The concept of Haar, Integral Image, AdaBoost features are used and then processed into the Cascade Classifier [8].

A. Haar Like Feature

Haar-like feature is used to detect objects in an image namely image processing in boxes, where in one box there are several pixels [15]. The squares are then processed and produce different values indicating dark and light areas. These values will be used as the basis for image processing shown in Fig. 1.

B. Integral Image

Integral Image is an image where the value of each pixel is an accumulation the top and left pixels values. For example, pixels (a, b) have accumulative values for all pixels (x, y) where x ≤ a and y ≤ b. According to Viola & Jones [7] integral image at location x, y, contains the number of pixels from the top to the left of x,y. Equation (1) is shown the formula of integral image ($ii(x,y)$) from $i(x',y')$ which is the original image.

$$ii(x,y) = \sum_{x'\le x, y'\le y} i(x',y') \quad (1)$$

$$S(x,y) = s(x,y-1) + (x,y) \quad (2)$$

$$ii(x,y) = ii(X-1,y) + S(x,y) \quad (3)$$

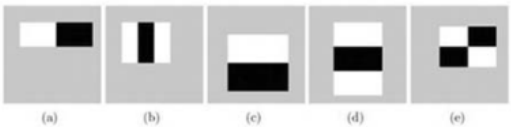

Fig. 1. Haar-Like feature pattern

1	2	3
4	5	6
7	8	9

Original image

1	3	6
5	12	21
12	27	45

Integral image

Fig. 2. Example of integral image pixels result

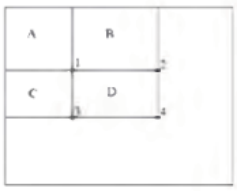

Fig. 3. Calculation process of integral image pixels.

Summation of cumulative lines $S(x,y)$ is calculated using (2) which can be used to gain the integral image as written in (3) where $S(x,-1) = ii(-1,y) = 0$. In fact, the integral image can be calculated by ignoring the original image. Fig.2 illustrates the example of the integral image result. Based on [7], the calculation of the number of pixels in rectangle D can be calculated with four reference arrays. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is A + B, the value at location 3 is A + C, and at location 4 is A + B + C + D. The addition in D can be calculated as 4 + 1- (2 + 3) shown in Fig. 3. The search process for this feature value is carried out iteratively starting from the top left corner of the image to the bottom right corner with a shift of ΔX and ΔY. The smaller the ΔX and ΔY values are, the more accurate the detection process is. The frequently used values for ΔX and ΔY are one.

C. AdaBoost

AdaBoost (Adaptive Boosting) functions sorting many features by only selecting certain features. Boosting occurs in iterations, incrementally adding the weak learner into one strong learner. One weak learner learns from training data each iteration. Then, the weak learner is added to the strong learner. After the weak learner is added, the data is then changed for each weight. The data that encountered the classification error will experience weight gain, and correctly classified data will be subjected to weight reduction. Therefore, the weak learner in the next iteration will be more focused on data that has been misclassified by the previous weak learner [16]. AdaBoost for feature selection and an intentional cascade for efficient computational resource allocation [17].

D. Cascade Classifier

The last step is to combine the Cascade Classifier in the Viola-Jones method concentrating the detection process on image areas that have the potential to have faces only. The characteristic of the Viola-Jones algorithm is the multilevel classification [18], [19]. The classification of this algorithm consists of three levels where each level issues a sub-image which is believed to be not an object or that there is no face in the image area [19]. This is done to assess whether the sub-image is an object that you want to detect or not. The workflow of multilevel classification is shown in Fig. 4.

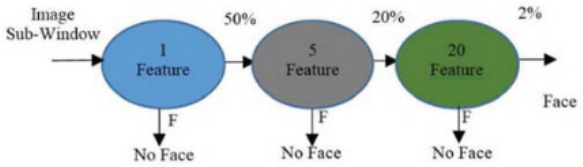

Fig. 4. Multilevel classification

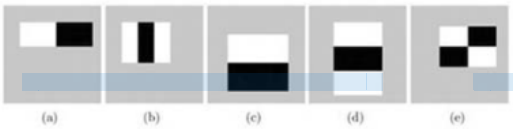

Fig. 1. Haar-Like feature pattern

1	2	3
4	5	6
7	8	9

Original image

1	3	6
5	12	21
12	27	45

Integral image

Fig. 2. Example of integral image pixels result

in the image area [19]. This is done to assess whether the sub-image is an object that you want to detect or not. The work-flow of multilevel classification is shown in Fig. 4.

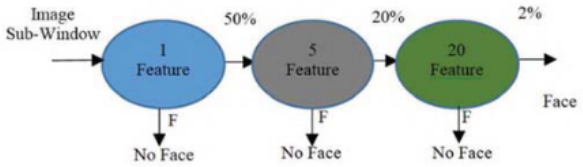

Fig. 4. Multilevel classification

In the first level classification, each sub-image will be classified using one feature. This classification approximately uses 50% sub-image to be classified in the second stage. As the classification level increases, more specific requirements are needed so that more features are used. The number of sub-images passing the classification reduces about 2%.

E. *The application of the Viola-Jones algorithm to the CPU process*

1) Step 1. The detection process begins with extracting the integral value from the image.

2) Step 2. Then, the integral value of the image will be used in the feature detection process using the classifier file.

3) Step 3. The detection process takes place until the program acquires features that have the potential to have facial features in the image. At this stage, the processing time is started to get the detection time value.

4) Step 4. When all the features that the program considers are faces, the feature area will be marked with a square around the face to indicate that there are already faces in the image. At this point, the processing time recording ends and results in how long it will take to detect faces.

The face detection process is implemented using the CPU computing platform. The steps for face detection on the CPU include: Load the face image, Allocate CPU memory, Performing CPU Processing and Free memory on the CPU

F. *The application of the Viola-Jones algorithm to the GPU process*

1) Step 1. The detection process begins with extracting the integral value from the image.

2) Step 2. Then, the integral value of the image will be used in the feature detection process using the classifier file.

3) Step 4. The detection process is carried out on the GPU with CUDA utilizing Threads and Blocks on the GPU.

4) Step 5. The detection process takes place until the program acquires features that have the potential to have facial features in the image. At this stage, the processing time is started to get the detection time value.

5) Step 6. When all the features that the program considers are faces, the feature area will be marked with a rectangle around the face to indicate that there are already faces in the image. At this point, the processing time recording ends and results in how long it will take to detect faces.

The face detection process is implemented using the GPU computing platform by utilizing the CUDA library. The GPU face detection process will take several stages for face detection, among others: a) Load the face image, b) Allocate CPU memory, c) Allocate GPU memory with the same amount of memory as the CPU using CUDA Malloc's library function, d) Retrieving data input (image/image data) from CPU memory called Memory host, d) Copying data into GPU memory using CUDA MemCpy's library function with CUDA Memcpy Host To Device parameter, so that data from host memory will be processed into device memory (GPU), e) Processing the GPU memory in the kernel is to run parallel computing according to the grids, blocks, and threads required for parallel processes, e) Copying result data in CPU memory using CUDA MemCpy's library function with CUDA Memcpy Device To Host parameter. In other words, the result data from the device memory (GPU) is returned to the host memory (CPU), and f) Freeing GPU memory or other threads using the CUDA Free library function.

III. Result and Discussion

In testing process, 5 images with 5 size variations, ranging from resolution 3000p, 2000p, 1000p, 720p, and 480p with a total of 25 images are used as shown in Fig. 5. Table I shows the results of the detection test with the Viola-Jones algorithm using the CPU computing platform.

Fig. 5. Tested Images

TABLE I. Detection Result using CPU

Image	Number of Faces	Detectable Face	Undetectable face	Accuracy	Detection Time (ms)
Resolution 3000					
Image 1	23	23	0	100 %	3492633
Image 2	23	22	1	96%	3607642.67
Image 3	22	21	1	95%	3565676.67
Image 4	4	4	0	100 %	3486980
Image 5	32	32	0	100 %	3607187.67
Resolution 2000					
Image 1	23	23	0	100 %	1636842.33
Image 2	23	22	1	96%	1689009.67
Image 3	22	20	2	91%	1711453.67
Image 4	4	4	0	100 %	1622725.33
Image 5	32	32	0	100 %	1649214
Resolution 1000					
Image 1	23	23	0	100 %	549910.33
Image 2	23	22	1	96%	1513335.67
Image 3	22	20	2	91%	1497908.67
Image 4	4	4	0	100 %	1506593.33
Image 5	32	23	0	100 %	549910.33
Resolution 720					
Image 1	23	23	0	100 %	519377.33
Image 2	23	22	1	96%	523606
Image 3	22	21	1	95%	508770

face detection process will take several stages for face detection, among others: a) Load the face image, b) Allocate CPU memory, c) Allocate GPU memory with the same amount of memory as the CPU using CUDA Malloc's library function, d) Retrieving data input (image/image data) from CPU memory called Memory host, d) Copying data into GPU memory using CUDA MemCpy's library function with CUDA Memcpy Host To Device parameter, so that data from host memory will be processed into device memory (GPU), e) Processing the GPU memory in the kernel is to run parallel computing according to the grids, blocks, and threads required for parallel processes, e) Copying result data in CPU memory using CUDA MemCpy's library function with CUDA Memcpy Device To Host parameter. In other words, the result

Image 5	32	32	0	100 %	1649214
Resolution 1000					
Image 1	23	23	0	100 %	549910.33
Image 2	23	22	1	96%	1513335.67
Image 3	22	20	2	91%	1497908.67
Image 4	4	4	0	100 %	1506593.33
Image 5	32	23	0	100 %	549910.33
Resolution 720					
Image 1	23	23	0	100 %	519377.33
Image 2	23	22	1	96%	523606
Image 3	22	21	1	95%	508770

Image	Number of Faces	Detectable Face	Undetectable face	Accuracy	Detection Time (ms)
Image 4	4	3	1	75%	506212
Image 5	32	31	1	97%	502479.67
Resolution 480					
Image 1	23	17	6	74%	90494.67
Image 2	23	14	9	61%	92433.33
Image 3	22	18	4	82%	88627
Image 4	4	3	1	75%	86454.33
Image 5	32	11	21	34%	87656

Detection results using CPU shown in Fig. 6. The detailed detection results using CPU are shown in Table I. Table II shows the detection test result with Viola Jones algorithm using GPU computing platform. Detection result using GPU shown in Fig.7. The detailed detection results using GPU are shown in Table II. Evaluation of the comparison between the CPU and GPU computing platforms is carried out by looking at the comparison value of the average percentage accuracy and the average length of the detection time shown in Table III.

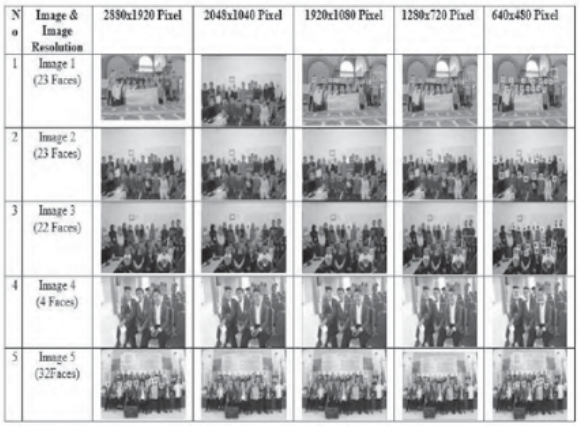

Fig. 6. Detection results using CPU

TABLE II. DETECTION RESULTS USING GPU

Image	Number of Faces	Detectable Face	Undetectable face	Accuracy	Detection Time (ms)
Resolution 3000					
Image 1	23	23	0	100%	564612
Image 2	23	22	1	96%	576047
Image 3	22	14	8	64%	572597
Image 4	4	4	0	100%	560399
Image 5	32	15	17	47%	570872
Resolution 2000					
Image 1	23	23	0	100%	274466
Image 2	23	23	0	100%	270991
Image 3	22	19	3	86%	271876
Image 4	4	3	1	75%	267999
Image 5	32	17	15	53%	272816
Resolution 1000					
Image 1	23	13	10	57%	98036.1
Image 2	23	23	0	100%	235073
Image 3	22	19	3	86%	237245
Image 4	4	3	1	75%	233177
Image 5	32	17	15	53%	238121
Resolution 720					
Image 1	23	13	10	57%	98260.3
Image 2	23	15	8	65%	94988.1
Image 3	22	11	11	50%	96666.8
Image 4	4	2	2	50%	97855.2
Image 5	32	18	14	56%	97093
Resolution 480					
Image 1	23	18	5	78%	21431.3
Image 2	23	20	3	87%	20270
Image 3	22	15	7	68%	20674.6
Image 4	4	2	2	50%	22328.6
Image 5	32	13	19	41%	20468.5

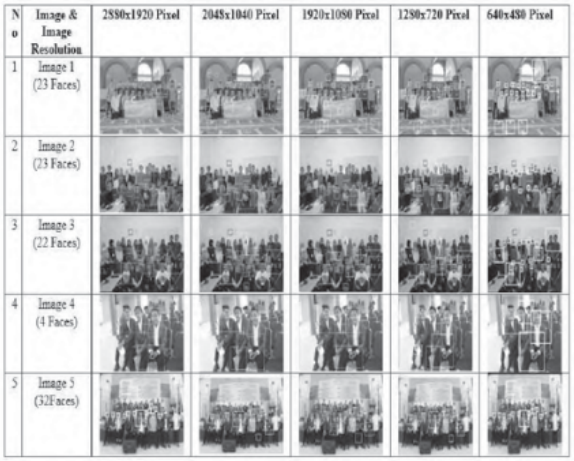

Fig. 7. Detection results using GPU

TABLE III. COMPARISON

No	Resolution	Average Accuracy		Average Detection Time (ms)	
		CPU	*GPU*	*CPU*	*GPU*
1	3000	98%	81%	3552024	568905.4
2	2000	97%	83%	1661849	271629.82
3	1000	97%	74%	1350148.533	208330.4067
4	720	93%	56%	512089	96972.693
5	480	65%	65%	89133.067	21034.593
Average		90%	72%	1433048.72	233374.583

Based on the average accuracy, detection using CPU is more superior the comparison result is presented that the GPU is superior because the algorithm used is more optimal for running on the CPU computing platform than on the GPU. This is due to the detect Multiscale value which is a key parameter to be able to detect facial objects which have a static value and can only detect objects from a certain distance, while the objects used in this paper are very varied in terms of the distance between the face with the camera. On the other hand, the performance of detection using GPU in terms of time consumption is six times faster than CPU because GPU uses more numerous CUDA Cores numerous than the cores on the CPU.

CONCLUSION

Based on the test results conducted in this research, the GPU was proven to be superior in terms of computation but

Image 1	23	23	0	100%	564612
Image 2	23	22	1	96%	576047
Image 3	22	14	8	64%	572597
Image 4	4	4	0	100%	560399
Image 5	32	15	17	47%	570872
Resolution 2000					
Image 1	23	23	0	100%	274466
Image 2	23	23	0	100%	270991
Image 3	22	19	3	86%	271876
Image 4	4	3	1	75%	267999
Image 5	32	17	15	53%	272816
Resolution 1000					
Image 1	23	13	10	57%	98036.1
Image 2	23	23	0	100%	235073
Image 3	22	19	3	86%	237245
Image 4	4	3	1	75%	233177

running on the CPU computing platform than on the GPU. This is due to the detect Multiscale value which is a key parameter to be able to detect facial objects which have a static value and can only detect objects from a certain distance, while the objects used in this paper are very varied in terms of the distance between the face with the camera. On the other hand, the performance of detection using GPU in terms of time consumption is six times faster than CPU because GPU uses more numerous CUDA Cores numerous than the cores on the CPU.

CONCLUSION

Based on the test results conducted in this research, the GPU was proven to be superior in terms of computation but

weak in terms of accuracy, while the CPU was superior in terms of accuracy compared to the GPU. This is because the algorithms used are basically developed using the CPU computing platform. So, when optimization is done on the GPU computing platform, the results obtained are not optimal. One of other reasons that causes inaccurate detection is that the library used can only detect facial objects visible from the front, so that when someone is not looking at the camera, then the object is considered not a face. Other factors are static detect Multiscale and various test objects. The detect Multiscale factor is very influential because it is a reference parameter of how close the object is to the camera that will be considered a face. In future research, it is hoped that it can be tested using other libraries that can detect facial objects not only from the front.

REFERENCES

[1] A. Mohanty, N. Suda, M. Kim, S. Vrudhula, J. S. Seo, and Y. Cao, "High-performance face detection with CPU-FPGA acceleration," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016, pp. 117–120.

[2] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design MIPS Edition: The Hardware/Software Interface*. Morgan Kaufmann, 2020.

[3] A. W. Y. Wai, S. M. Tahir, and Y. C. Chang, "GPU acceleration of real time Viola-Jones face detection," in *2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, 2015, pp. 183–188.

[4] D. Kurniawan, T. B. Adji, and N. A. Setiawan, "Analisis Perbandingan Komputasi GPU dengan CUDA dan Komputasi CPU untuk Image dan Video Processing," in *Seminar Nasional Aplikasi Teknologi Informasi (SNATI)*, 2015, vol. 1, no. 1.

[5] V. Jain and D. Patel, "A GPU based implementation of robust face detection system," *Procedia Comput. Sci.*, vol. 87, pp. 156–163, 2016.

[6] S. N. Kane, A. Mishra, and A. Gaur, "International Conference on Recent Trends in Physics (ICRTP 2014)," in *Journal of Physics: Conference Series*, 2014, vol. 534, no. 1, p. 11001.

[7] 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*, 2001, vol. 1, pp. I–I.

[8] G. E. Lescano, P. Santana Mansilla, and R. Costaguta, "Analysis of a GPU implementation of Viola-Jones' algorithm for features selection," *J. Comput. Sci. Technol.*, vol. 17, no. 1, pp. 68–73, 2017.

[9] V. Mutneja and S. Singh, "Modified Viola–Jones algorithm with GPU accelerated training and parallelized skin color filtering-based face detection," *J. Real-Time Image Process.*, vol. 16, no. 5, pp. 1573–1593, 2019.

[10] N. Naik and G. N. Rathna, "Robust real time face recognition and tracking on GPU using fusion of RGB and depth image," *arXiv preprint arXiv:1504.01883*. 2015.

[11] I. Guerfi, L. Kriaa, and L. A. Saidane, "An Efficient GPGPU based Implementation of Face Detection Algorithm using Skin Color Pretreatment," in *InProceedings of the 15th International Conference on Software Technologies (ICSOFT 2020)*, 2020, pp. 574–585.

[12] J. Masek, R. Burget, V. Uher, and S. Güney, "Speeding up Viola-Jones algorithm using multi-Core GPU implementation," in *2013 36th International Conference on Telecommunications and Signal Processing (TSP)*, 2013, pp. 808–812.

[13] H. Jia, Y. Zhang, W. Wang, and J. Xu, "Accelerating viola-jones facce detection algorithm on gpus," in *2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*, 2012, pp. 396–403.

[14] V. Gangadhar, S. Shridhar, and R. S. M. Bamdhamravuri, "An Efficient GPGPU Implementation of Viola-Jones Classifier Based Face Detection Algorithm."

[15] W. Wang, Y. Zhang, S. Yan, Y. Zhang, and H. Jia, "Parallelization and performance optimization on face detection algorithm with OpenCL: A case study," *Tsinghua Sci. Technol.*, vol. 17, no. 3, pp. 287–295, 2012.

[16] M. D. Putro, T. B. Adji, and B. Winduratna, "Sistem Deteksi Wajah dengan Menggunakan Metode Viola-Jones." 2012.

[17] R. Patel and I. Vajani, "Face Detection on a Parallel Platform using CUDA Technology," *Natl. J. Syst. Inf. Technol.*, vol. 8, no. 1, p. 17, 2015.

[18] S. Rahmad, R. A. Asmara, D. R. H. Putra, I. Dharma, H. Darmono, and I. Muhiqqin, "Comparison of Viola-Jones Haar Cascade classifier and histogram of oriented gradients (HOG) for face detection," in *IOP conference series: materials science and engineering*, 2020, vol. 732, no. 1, p. 12038.

[19] R. Marineau, "Parallel Implementation of Facial Detection Using Graphics Processing Units." 2018.