International Journal of
Applied Mathematics,
Electronics and Computers
ISSN: 2147-8228 http://ijamec.atscience.org
Advanced Technology and Science
Original Research Paper

Performance Evaluations for OpenMP Accelerated Training Of Separable Image Filter

Süleyman UZUN*[1], Devrim AKGÜN[2]

Accepted 3rd September 2016

Abstract: One of the widespread image processing applications is image filtering with two dimensional convolution. Determining the weights of image filters are of importance for the success of filtering operation. Heuristic algorithms such as genetic algorithms provide an efficient way of training these types of filters. Due to the high computational cost of repetitive image filtering operations, this process may take hours to implement using single core computing. OpenMP (Open Multi Processing) provides an efficient library for utilizing the computing power of multicore processors. In this study, OpenMP accelerated training of separable filters that are a subclass of convolution filters has been implemented based on genetic algorithms. Comparative speed-up results for various sizes of images using various sizes of filtering kernels were presented. Also the effect of population size of genetic algorithm and the number of working cores have been investigated.

Keywords: *OpenMP, separable filters, image processing, genetic algorithms.*

1. Introduction

Image filters are widespread operators in image processing applications such as image enhancement, image smoothing, edge detection and noise elimination [1]. Linear filtering using two dimensional convolution or correlation is one of the main filtering operations. This is realized by applying the filtering kernel to each pixel of input image where the kernel is a matrix of weights. The size of the matrix can be 3×3 or 5×5 larger such as 21×21. If the kernel has symmetric properties, it can be expressed as the multiplication of a row and column vectors. This form is called as separable filter and it reduces the number of multiplication/addition operations.

The values of the filter weights are determined according to the desired behaviour of the filter. Weights can readily be obtained using different analytical techniques [2-3]. In another approach, the kernel weights can be trained using the original and noisy image samples [4-7]. Heuristic algorithms provide an efficient way for the computation of the filter kernel weights [8-10]. One of the well-known heuristic algorithms is Genetic algorithm which provided its efficiency in various researches. Genetic algorithm is selected to train the weights of the separable filter. In the application of genetic algorithms, a fitness function is used to define the problem. In the present case, fitness functions is formed according to mean squared value of original and noisy images. For the computation of fitness function, intense multiplication and addition operations are carried out to obtain fitness value. Furthermore, computation time depends on the number of weights as well as the image size. During computations, fitness function is called at each iteration of the genetic algorithm. This significantly slow down the process and make the applications impractical. A method for the acceleration of the process is to utilize the computational power of multicore processor. For this purpose, OpenMP provides a useful tool for efficient use of the cores of a multicore processor. OpenMP helps distribute the computational load to defined number of threads.

In the present study, OpenMP is utilized to accelerate the computation of fitness function. Fitness function is computed for all individuals in the population and these operation can be realized independently on processor cores. In the experiments, an eight core computer is used and the results are obtained against the number of cores to see the effect of the number of cores. Also various filter kernel sizes, image sizes and the population sizes used in the experiments to show the efficiency of the OpenMP based acceleration.

2. Separable Image Fılters

Separable image filters are used in a slightly different way from the non-separable filters. An example of non-separable image filter is shown by Fig. 1 which has size 3×3. The filter kernel has a total of 9 weights. This means that the image filtering process train weights number of genetic algorithms is 9. When the size of the filter kernel grows, it is increasing training time of genetic algorithms. For instance, 25 weights for the filter kernel with 5×5, 49 weights for the filter kernel with 7×7, 81 weights for the filter kernel with 9×9, etc. The growth of genetic algorithms filter kernel increases the training time. The 3×3 filter kernel used in separable image filter is shown in Fig 1. This filter kernel which horizontal and vertical vectors as shown in Fig. 2 in the separable image filter is used.

1	2	1
2	4	2
1	2	1

Figure 1. 3×3 Filter size which filter kernel.

[1] *Information Technologies Department, Bilecik Şeyh Edebali University, Gülümbe Campus, 11230, Bilecik/Turkey*
[2] *Computer Engineering Department, Faculty of Computer and Information Sciences, Sakarya University, Esentepe Campus, 54187, Sakarya/Turkey*
** Corresponding Author: Email: suleyman.uzun@bilecik.edu.tr*
Note: This paper has been presented at the 3rd International Conference on Advanced Technology & Sciences (ICAT'16) held in Konya (Turkey), September 01-03, 2016.

The product of this vector also gives the filter kernel shown in Fig. 1.

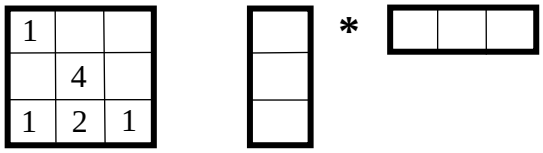

Figure 2. Separable image filter is used filter kernel.

Separable image filtering process can be divided into two stages. The first stage is to filter noisy image using one of the vectors. The second stage is to filter the resulting image from the first stage using the other vector. Therefore, image filtering process is completed in two stages. Separable image filter has the advantage of reduced number of weights over non-separable filter. For example, the number of filter weights to be trained in separable image filter for 5×5 is 10, while it is 25 for non-separable image filter.

3. Genetic Algorithms

Genetic algorithm is a search and optimization method which is based on natural selection [7, 11-12]. Genetic algorithms randomly generate multiple solutions. Bad solutions are eliminated in the next generation. Therefore, best solutions appear as the best solutions transferred to next generations.

Initial population
Fitness
While (termination criteria is not met)
Selection
Crossover
Mutation
Fitness
End while

Figure 3. Genetic algorithm steps.

Genetic algorithm involves applying selection, crossover, mutation and Fitness calculations on candidate population which are initially formed randomly. A pseudo code illustrating the operation of genetic algorithms is shown in Fig. 3. In the present case, computationally most intensive part is calculating the value of the fitness function due to the image filtering operations.

4. OpenMP (Open Multi Processing)

OpenMP is an application programming interface (API) which provide opportunity parallel computing on multicore processors. The calculations are done on multi-core processor architectures OpenMP thanks to coequally distribute all core [13].

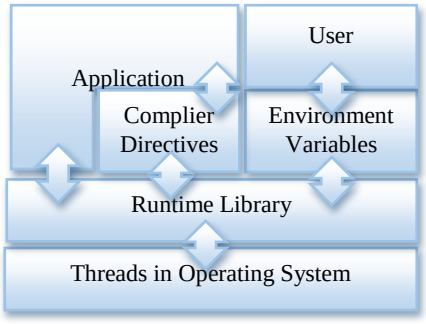

Figure 4. OpenMP architecture [14].

OpenMP Architecture is shown figure 4. OpenMP, compiler directives, runtime library and environment variables are comprised from. Programmers write the code to run concurrently by putting special comments in that codes.For instance "#pragma omp parallel". This study was parallelization of the fitness value calculating for "for" block.

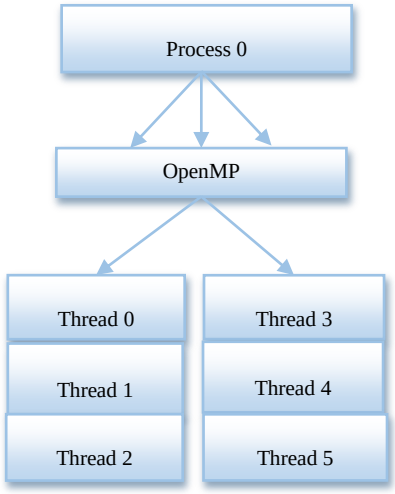

Figure 5. OpenMP hierarchy of 2D array[14].

OpenMP's operation diagram is shown in Fig 5. OpenMP is identified one of the threads as main thread. Tasks are distributed in equal amounts other threads by the main thread. Due to the fact that this study was developed with C programming language, to use the OpenMP function "#include <omp.h>" as is included in the project [15].

Fig. 6 shows the area of the genetic algorithms parallelization process on the flow chart. This block is calculated fitness function value. This block contains computationally intensive mathematical operations. Therefore parallelization process is performed here.

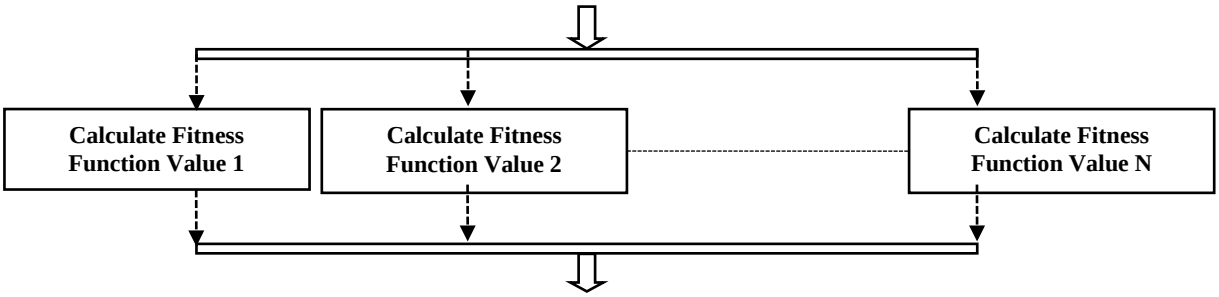

Figure 6. Computation of fitness functions using OpenMP.

5. Experimental Results and Discussion

Experimental studies on Windows Server 2012 Essentials™ 64-bit operating system, Quad-Core AMD Opteron™ 2378 2.40GHz dual processor, 18GB Ram, have been working on computer servers. The algorithm is written in C programming language. In the experiments, 256×256, 512×512 and 1024×1024 with a pixel size images are used [16]. 3×3, 5×5 and 7×7 sizes filter masks are used for images filtering process. The developed algorithm was running 10 times for each image and at the end of working, these 10 average MSE (Mean Squared Error) value and average training time are taken. The number of population for analysing the impact of population on training time while 100 and 200 were obtained results determined separately. The number of iterations has been fixed at 400 for all calculations. Mutation rate 0.005 and crossover rate 0.3 is defined as the constant. Termination criteria of genetic algorithms are defined as the number of iterations. Noisy image used were obtained by adding Gaussian noise on the original image.

Table 1. Computation times for 100 population and 3x3 filter kernel

100 Population, 3×3 Filter Kernel					
Image Size	Core Numbers (TIME (Minute))				
	1 Core	2 Core	4 Core	6 Core	8 Core
256×256	3.43	1.71	0.90	0.75	0.50
512×512	14.81	7.17	4.06	3.25	2.88
1024×1024	55.52	29.11	16.28	12.83	12.27

Table 2. Computation times for 200 population and 3x3 filter kernel

200 Population, 3x3 Filter Kernel					
Image Size	Core Numbers (TIME (Minute))				
	1 Core	2 Core	4 Core	6 Core	8 Core
256×256	6.74	3.20	1.70	1.58	0.94
512×512	25.93	14.60	8.10	6.20	5.59
1024×1024	107.02	55.85	34.34	24.66	23.07

Table 3. Computation times for 100 population and 5x5 filter kernel

100 Population, 5×5 Filter Kernel					
Image Size	Core Numbers (TIME (Minute))				
	1 Core	2 Core	4 Core	6 Core	8 Core
256×256	3.97	2.02	1.13	1.00	0.62
512×512	16.16	8.79	4.86	3.61	3.03
1024×1024	68.75	34.85	19.67	14.03	12.42

Table 4. Computation times for 200 population and 5x5 filter kernel

200 Population, 5×5 Filter Kernel					
Image Size	Core Numbers (TIME (Minute))				
	1 Core	2 Core	4 Core	6 Core	8 Core
256×256	8.47	4.24	2.30	1.74	1.16
512×512	32.46	17.08	9.33	6.96	5.80
1024×1024	138.97	69.68	38.47	27.50	23.81

Table 5. Computation times for 100 population and 7x7 filter kernel

100 Population, 7×7 Filter Kernel					
Image Size	Core Numbers (TIME (Minute))				
	1 Core	2 Core	4 Core	6 Core	8 Core
256×256	4.03	2.29	1.27	1.14	0.69
512×512	19.27	9.65	5.68	3.86	3.00
1024×1024	77.81	38.79	21.85	15.63	12,29

Table 6. Computation times for 200 population and 7x7 filter kernel

200 Population, 5×5 Filter Kernel					
Image Size	Core Numbers (TIME (Minute))				
	1 Core	2 Core	4 Core	6 Core	8 Core
256×256	9.54	4.75	2.64	2.02	1.30
512×512	36.45	20.87	10.68	7.52	5.93
1024×1024	157.92	80.12	42.75	29.94	24.52

Table 1 and Table 2 shows the computational times for 3×3 window using 100 and 200 populations respectively. Table 3 and Table 4 shows the computational times for 5×5 window using

100 and 200 populations respectively. Table 5 and Table 6 shows the computational times for 7×7 window using 100 and 200 populations respectively. All results show that as the number of cores increased, the computational times reduces significantly. Figure 7a to 7f show the graphical comparison of the results. Best acceleration rates are obtained for 256×256 image.

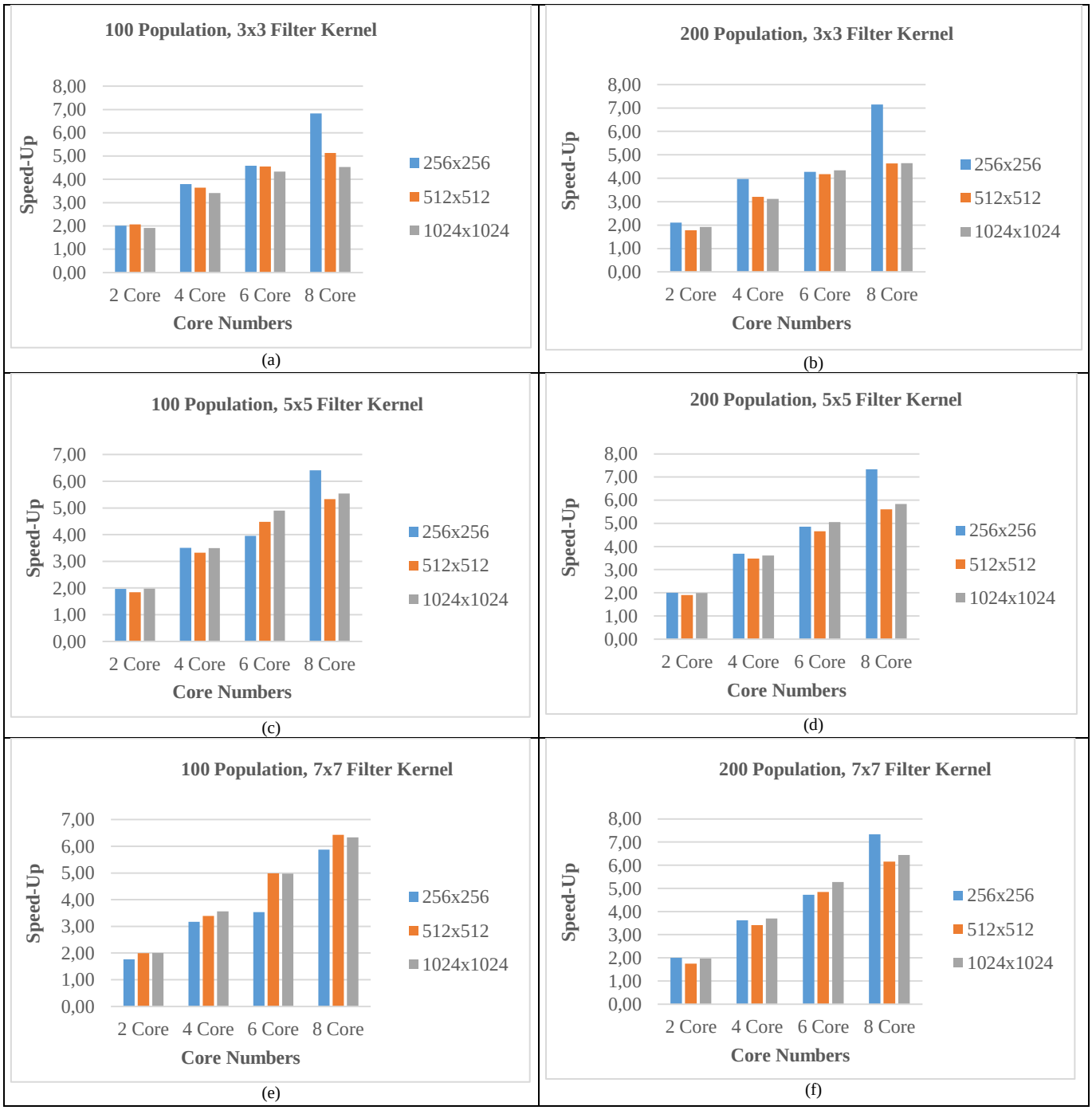

Figure 7. Comparative acceleration graphs.

6. Conclusion

In this study, OpenMP with accelerated training of separable image filter were analysed. In the experimental results, various sizes of kernels, and images and population sizes were tested. According to the results, doubling the population size has an increasing effect when the speed-up values on the average. Increasing the kernel size doesn't change the results much. In general, the results show significant accelerations over single core running durations. For future studies the results will be obtained on a machine to see the efficiency limit of the number of cores.

References

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. 2007.

[2] A. Karasaridis and E. Simoncelli, "A filter design technique for steerable pyramid image transforms," *1996 IEEE Int. Conf. Acoust. Speech, Signal Process. Conf. Proc.*, vol. 4, pp. 2387–2390, 1996.

[3] J. Yang, L. Liu, T. Jiang, and Y. Fan, "A modified Gabor filter design method for fingerprint image

enhancement," *Pattern Recognit. Lett.*, vol. 24, no. 12, pp. 1805–1817, 2003.

[4] R. Poli, "Genetic Programming for Image Analysis," in *Genetic Programming 1996: Proceedings of the First Annual Conference*, 1996, pp. 363–368.

[5] D. Akgün and P. Erdoğmuş, "GPU accelerated training of image convolution filter weights using genetic algorithms," *Appl. Soft Comput.*, vol. 30, pp. 585–594, 2015.

[6] D. J. Krusienski and W. K. Jenkins, "Particle swarm optimization for adaptive IIR filter structures," *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 1. p. 965–970 Vol.1, 2004.

[7] G. J. E. Rawlins, "Foundations of Genetic Algorithms," in *Foundations of Genetic Algorithms*, 1991, vol. 21, p. 341.

[8] M. Haseyama and D. Matsuura, "A filter coefficient quantization method with genetic algorithm, including simulated annealing," *Signal Process. Lett. IEEE*, 2006.

[9] D. M. Weber and D. P. Casasent, "Quadratic Gabor filters for object detection," *IEEE Trans. Image Process.*, vol. 10, no. 2, pp. 218–230, 2001.

[10] Y. Wang, B. Li, and Y. Chen, "Digital IIR filter design using multi-objective optimization evolutionary algorithm," *Appl. Soft Comput.*, 2011.

[11] G. Emel and Ç. TAŞKIN, "Genetik Algoritmalar ve Uygulama Alanlari," *Uludağ Üniversitesi İktisadi ve İdari Bilim.*, 2002.

[12] K. de Jong, "Learning with Genetic Algorithms: An Overview," *Mach. Learn.*, vol. 3, no. 2, pp. 121–138, 1988.

[13] A. S. Al-Hamoudi and A. Ahmed Biyabani, "Accelerating data mining with CUDA and OpenMP," in *2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA)*, 2014, pp. 528–535.

[14] "OpenMP Architecture." [Online]. Available: http://www.lrz.de/services/software/parallel/openmp/.

[15] H. P. Computing, "Parallel Programming with OpenMP," *Sci. Technol.*, pp. 1–91, 2010.

[16] G. Weber, "USC-SIPI image database: Version 4," 1993.

