Parallel Implementation of Morphological Image Processing Algorithm for GPGPU

Muhammad Ali Ismail and Kamran Shamim
High Performance Computing Centre
Department of Computer & Information Systems Engineering
NED University of Engineering & Technology, Karachi, Pakistan
maismail@neduet.edu.pk and kamranshamim@hotmail.com

***Abstract*— Morphological Image Processing also known as Morphology is used as a tool for extracting image components that are useful in the representation and description of region such as shape, boundaries, skeleton etc. Morphological techniques are also used for pre- and post-processing of given data for morphological filtering, noise removal, thinning, pruning and edge detection. In this paper a parallel implementation of morphological algorithm for GPGPU is presented. The presented implementation identifies the unbroken pills and remove noise from the image in parallel. The result obtained shows the speedup of almost 85 for erosion operation and around 70 for dilation operations on GPGPU.**

Keywords*— *Morphology, Errosion, Dilation, GPGPU, CUDA, Parallel Programming

I. INTRODUCTION

Mathematical Morphology (MM) is normally employed for analysis and processing of geometrical structures, based on set theory, lattice theory, and random functions. It can be formulated in terms of set theory. Sets represent objects in an image; for instance, the set of all white pixels in a binary image is a complete morphological description of an image. In binary images, the sets are members of the 2D integer space Z^2, where each element of a set is a tuple (2D vector) whose coordinates are the (x, y) coordinates of a white (or black) pixel in the image. Gray-scale images can be represented as sets, whose components are in Z^3: two components are coordinates of a pixel, and the third – its discrete intensity value. The basic idea in binary morphology is to probe an image with a simple, pre-defined shape, drawing conclusions on how this shape fits or misses the shapes in the image. This simple "probe" is called structuring element, and is itself a binary image (i.e., a subset of the space or grid). Fig 1 represents some examples of structuring elements (SE), shaded square denotes a member of the SE. The origins of SEs are marked by a black dot.

There are two major operations in Morphology which are Erosion and Dilation. When the structuring element B has a centre and located on the origin of E, then the erosion of A by B can be understood as the locus of points reached by the centre of B when B moves inside A. The erosion of the binary image A by the structuring element B is defined by:

$$A \ominus B = \{z \in E \mid B_z \subseteq A\} \quad (1)$$

Where Bz is the translation of B by the vector z, that is;

$$B_Z = \{b + z \mid b \in B\}, \forall z \in E \quad (2)$$

Whereas the dilation If B has a center on the origin, as before, then the dilation of A by B can be understood as the locus of the points covered by B when the center of B moves inside A. It can be obtained by:

$$A \oplus B = \{z \in E \mid (B^s)_z \cap A \neq \Phi\} \quad (3)$$

Where Bs denotes the symmetric of B, that is,

$$B^s = \{x \in E \mid -x \in B\} \quad (4)$$

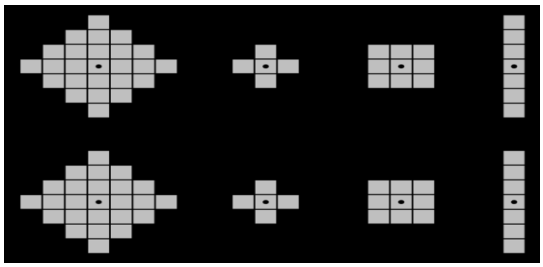

Fig. 1. Structures for Morphological Processing

Fig. 2 and Fig. 3 represent the some erosion and dilation examples of round shaped object by disk shaped structuring element of various radiuses.

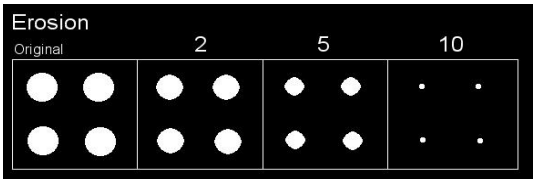

Fig. 2. Erosion operation for "o" with variable window size

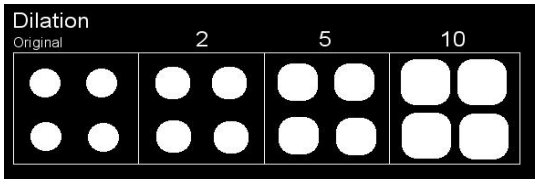

Fig. 3. Dilation operation for "o" with variable window size

Rapid developments in graphics processor unit (GPU) technology have, over the past four years in particular, inspired heightened research interest in the field of general purpose GPU computation. Areas of focus have varied from solvers for complex mathematics, physics and life sciences simulations, to high performance image processing and search optimization. In this paper, the two fundamental principles of image processing that is Dilation and Erosion have been processed through GPGPU in order to minimize the image processing time.

The distribution of paper is as follows. Section II focuses on related work. Section III elaborates the sequential and parallel implementation of Morphological Image Processing. In section IV the outcome of the serial and parallel implementations have been discussed. Section V finally concludes the paper.

II. RELATED WORK

In the past few years there were a number of attempts to implement parallel Morphological Image Processing (MIP) on different parallel architectures. some of them include GPU [1], Opto-Electronic VLSI Array Processor [2], SIMD, MIMD and PASM [3] Multi-core processors [4, 5], Heterogeneous computer network [6] and Cell/BE with OpenCV [7].

The work most related to the presented work that is GPGPU implementation of MIP is discussed in [8-12]. All these work proved GPGPU better and useful approach from other available parallel architecture as it offers a general purpose parallel multithreaded environment. In addition the work that distinguish our work with the previous is the use of multiple window sizes for morphological operations and to determine the best fit for execution on both CPU and GPGPU environment.

III. IMPLEMENTATION

This section covers both sequential and parallel implementation of the Dilation and Erosion algorithms. The basic algorithm is realized for detecting unbroken pills for the given sample set feeding as an image to the system that perform Morphological Image Processing.

Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries. The number of pixels added or removed from the objects in an image depends on the size and shape of the structuring element used to process the image. In the morphological dilation and erosion operations, the state of any given pixel in the output image is determined by applying a rule to the corresponding pixel and its neighbors in the input image. The rule used to process the pixels defines the operation as dilation or erosion. Following table-I lists the rules for both dilation and erosion.

To process border pixels, the morphological functions assign a value to these undefined pixels, as if the functions had padded the image with additional rows and columns. The value of these padding pixels varies for dilation and erosion operations.

TABLE I. RULES FOR OPERATIONS

Operation	Rule
Dilation	The value of the output pixel is the maximum value of all the pixels in the input pixel's neighbourhood. In a binary image, if any of the pixels is set to the value 1, the output pixel is set to 1.
Erosion	The value of the output pixel is the minimum value of all the pixels in the input pixel's neighbourhood. In a binary image, if any of the pixels is set to 0, the output pixel is set to 0.

A. Sequential Implementation

To perform dilation and erosion on images it is required to read actual image data from image files. In order to avoid unnecessary complexities involved in decoding various image formats, 8-bit BMP Image format is selected. 8-bit BMP image format stores 8-bit bitmap of every pixel, hence obtaining image data is very much eased. Same operation can be applied for 16, 32 and 64 bit BMP images.

For performing morphological dilation and erosion, *structuring element* is a key component. In this paper, square shaped structuring element of *N* x *N* sizes is applied on images. Size of structuring element, for example 3x3, 5x5, is varied to examine behavior of morphological dilation and erosion with variation in the size of structuring element. In serial implementation, whole input image is visited pixel-by-pixel and values of output pixels are decided by applying dilation and erosion rules. Below is the pseudo code for CPU implementation of morphological dilation and erosion for processing of an *N* x *M* image:

Algorithm 1: Pseudo Code for morphological dilation and erosion on CPU

```
1.   Repeat Step 2 to 15 for X:= 1 to N
2.     Repeat Step 3 to 15 for Y:= 1 to M
3.       If (boundary_pixel) == True
4.         Operation for assigning boundary pixel
5.       Else Repeat Step 6 to 10
6.         Assign List:= [Neighbouring pixels]
7.         For I:= 1 to N
8.           For J:=1 to M
9.             If pixel(I,J)==1
10.              Assign List:= inImg(X+I-sz, Y+J-sz)
11.      If Operation.dillation == True
12.        Val := max(List)
13.      Else If Operation.erosion == True
14.        Val := min(List)
15.      outImg(X,Y) :=val;
```

In the above code we realize two functions inImg and outImg. The purpose of inImg is to input a pixel value from sample image whereas the purpose of outImg is to store back the pixel value in the final output image, rest of the steps are

self-explanatory. Step 4 is basically for padding additional rows and columns to the existing boundary pixels.

B. GPGPU based Implementation

For performing the Morphological Image Processing in parallel consider the 2D matrix in Fig. 4. There are five identified regions in the matrix that may be treated as logically separate structures. These structures can be processed in parallel.

1	1	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0
1	1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	0	0
1	1	1	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1

Fig. 4. 2D data for Morphological processing

The amenability of Morphological Image Processing to data parallel programming provides an opportunity for performance acceleration using highly parallel processors, such as GPGPUs. Implementation of dilation and erosion on GPGPU can provide impressive speedups over CPU implementations for a range of structuring element sizes.

The parallelization of morphological dilation and erosion, lies in implementing kernels (functions to be executed on GPGPU) which can perform dilation and erosion on single pixel and launch these kernels with as many threads as there are pixels in input image. Obviously, each and every thread cannot run in parallel due to resource limitations, threads schedulers inside GPGPU will schedule threads according to available resources and marked increase in performance of morphological operations will be noticed. While launching kernel in parallel threads, limit of maximum number of threads imposed by GPGPU should be considered. Pseudo code for CUDA kernels to perform morphological dilation and erosion is given below:

Algorithm 2: Pseudo code - Kernel function for morphological dilation

1. *Repeat Step 2 to 4 for I := 1 to N*
2. *Repeat Step 3 to 4 for J := 1 to M*
3. *If pixel(I,J)==1*
4. *Assign List:= inImg(X+I-sz, Y+J-sz)]*
5. *Val := max(List)*

All the statements inside loop I are executed in parallel by creating parallel threads through the following statement.

*int x = blockDim.x * blockIdx.x + threadIdx.x*

Algorithm 3: Pseudo code - Kernel function for morphological erosion

1. *Repeat Step 2 to 4 for I := 1 to N*
2. *Repeat Step 3 to 4 for J := 1 to M*
3. *If pixel(I,J)==1*
4. *Assign List:= inImg(X+I-sz, Y+J-sz)]*
5. *Val := min(List)*

In order to process m threads a conditional statement must be used as under;

If (x < m) { // Statements inside loop B }

These statements will generate ***M*** threads that will execute the body statements of loop I. Again in loop B the EDC subsection of algorithm is executed in parallel on m threads to decrease the overall complexity of the algorithm.

The image matrix first needs to be transferred to the device memory using cudaMallocPitch and cudaMemcpy2D functions. The final image matrix is transferred to the host after processing using the same cudaMemcpy2D function.

IV. DISCUSSION

The CPU and the GPGPU device utilized in this implementation are listed in Table - II.

TABLE II. SPECIFICATION (A) HOST (B) DEVICE

CPU	
Processor	Core 2 Quad CPU @ 4 x (2.66 GHz)
Memory	8 GB
OS	Windows 7 (32-bit)
GPGPU	
Device	nVIDIA Tesla C1060
Thread Block Size	[512, 512, 64]
IDE	CUDA
Compute Capability	1.3
Driver Version	4.2
Multiprocessor Count	30
Clock Rate (KHz)	1296000

Fig. 5 represents the initial test image. It contains various shapes to demonstrate behavior of morphological operation in detail. Circles of various sizes, rectangle are representing symmetrical shapes, while triangle and semi-circles are representing asymmetrical shapes. Little objects, of random shapes and sizes, at random locations are intended to mimic noise in image.

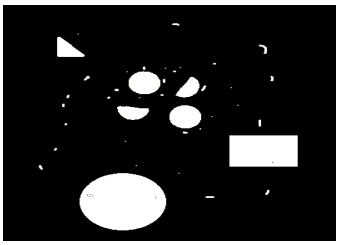

Fig. 5. Initial test image

When morphological erosion is performed on above image with square shaped structuring element of various sizes, following images are obtained, Fig. 6.

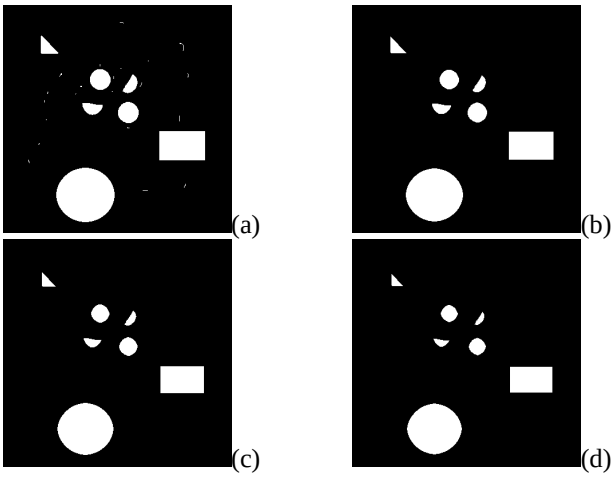

Fig. 6. Morphological erosion with window size (A) 3 (B) 5 (C) 7 and (D) 9

Fig. 7 represents the resulted images after applying the morphological dilation with square shaped structuring element of various sizes. Morphological erosion and dilation is applied to eliminate noise from the test image. When sample images was eroded with structuring element of size 3, noise with smaller size disappeared. when size of structuring element was increased to 5, all objects mimicking noise tend to disappear. But it is worth noting, application of erosion shrinks the objects in the sample image. If application of erosion was intended to filter noise then shrinking of shapes is not acceptable, as noise filter should only remove noise while retaining actual objects as it is. When Morphological dilation was applied on sample image then result was worst. Though, morphological dilation is one of the two fundamental morphological operations, but wrong choice of algorithm can devastate whole image processing. On application of dilation, each object in the image tends to expand and result get worst with larger noise objects when compared to sample image.

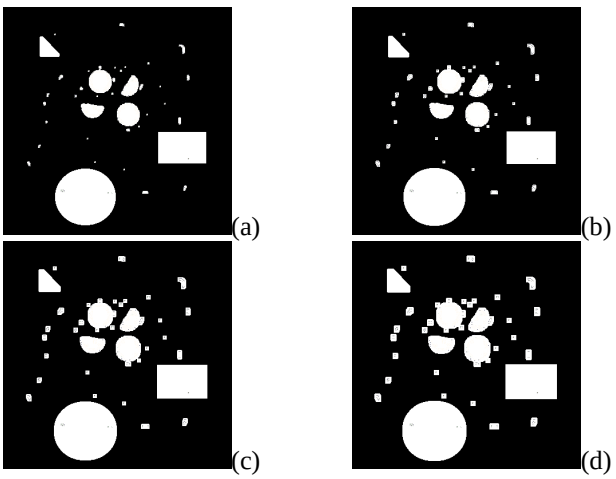

Fig. 7. Morphological dillation with window size (A) 3 (B) 5 (C) 7 and (D) 9

If it is intended to remove noise from same input image while retaining actual size of objects then advanced morphological operations can be used. In this case, Morphological opening, Erosion followed by Dilation, can give quite favorable results. When morphological opening is applied on sample image with structuring element of various sizes, following images are obtained. Fig 8 shows the effects of applying morphological erosion and dilation on sample image respectively. Structuring element used was square shaped with size 3, 5, 7 and 9. As explained earlier, erosion tends to remove pixels while dilation adds pixels.

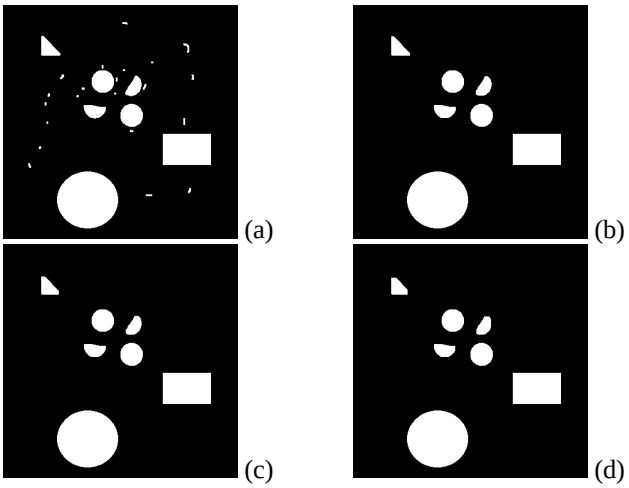

Fig. 8. Morphological opening with window size (A) 3 (B) 5 (C) 7 and (D) 9

Above set of images (fig. 8) clearly shows, morphological opening successfully removed noise objects while retaining original object size. But when size of structuring element is increase to 7 and 9, edges of semi-circles and triangle get distorted. This is because square shaped (symmetrical) structuring element is not well suited for semi-circle and triangle (asymmetric) objects.

V. RESULTS AND CONCLUSION

The tests were made on different images with varying window sizes. however only limited results have been shown here. The results for sequential and parallel implementation can be summarized in tabulated form in Table - III.

TABLE III. SIMULATION RESULTS (A) CPU (B) GPGPU

Morphological operation time in (ms)			
Window Size	**CPU**	**GPGPU**	**Speedup**
Erosion			
3	6.66	0.077	86.49
5	6.66	0.178	37.42
7	20	0.334	59.88
9	30	0.534	56.18
Dilation			
3	6.66	0.103	64.66
5	10	0.208	48.08
7	20	0.364	54.95
9	40	0.572	69.93

Note that the values calculated in above table-III are through time stamp function in C++ and CUDA. These values might have a tolerance of 5 percentile depending upon the current contents of cache. The simulation results for erosion mentioned in table-III can be plotted against each other on a line graph as represented in fig. 9.

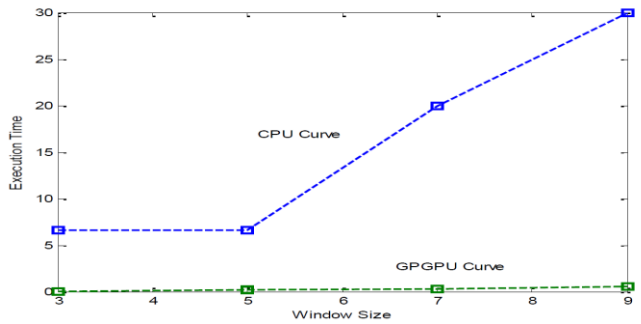

Fig. 9. Comparison Chart for table III simulation (Erosion)

Similarly the simulation results for dilation mentioned in table 3 can be plotted against each other on a line graph as represented in fig. 10.

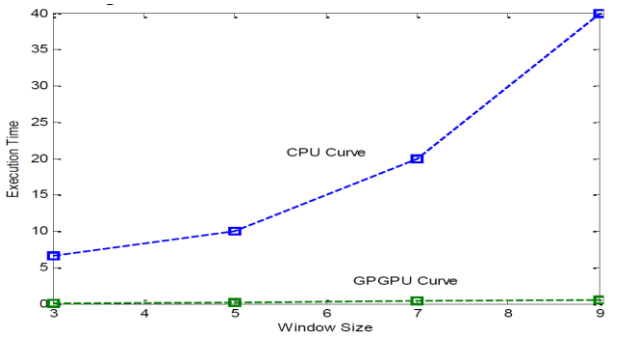

Fig. 10. Comparison Chart for table III simulation (Dilation)

In fig. 11 the simulation results for Erosion and Dilation are depicted. The x-axis refers to the window size which is according to the variations mentioned in Table 3, whereas y-axis refers to the execution time in milliseconds.

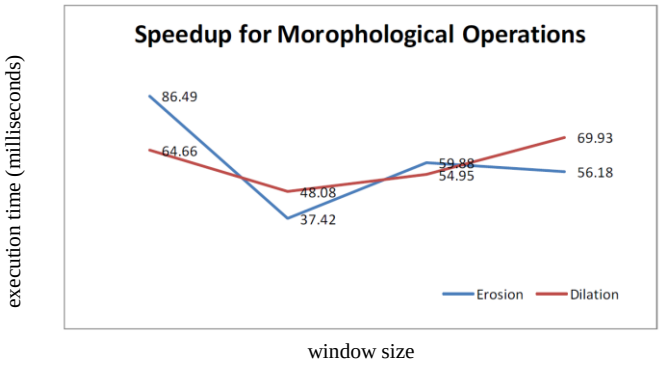

Fig. 11. Chart for speedup for Morphological operations

From the Fig 11 and the simulation results summarized in table III, it is evident that GPGPU offers much better performance for any data set. As the size of the data set increases GPGPU starts showing a steep improvement in the processing of the currently applied algorithm.

Results obtained clearly show the performance enhancement for morphological dilation and erosion process using the proposed parallel implementation on GPGPU. The speedup of almost 85 for erosion operation and around 70 for dilation operations were obtained. It is also noted that speedup obtained demands on two things. Firstly, the size of structuring element or window size. In can be observed in the fig. 11. The other is GPGPU processing capability. this enforce that optimal sized structuring element should be selected to exploit the full capability of GPGPU. In short, dilation and erosion are fundamental morphological operations and form basis of many morphological image processing algorithms. Their speed enhancements can lead to speed enhancement of any algorithms in which they are used.

REFERENCES

[1] Thurley, M.J.; Danell, V., "Fast Morphological Image Processing Open-Source Extensions for GPU Processing With CUDA," in Selected Topics in Signal Processing, IEEE Journal of , vol.6, no.7, pp.849-855, Nov. 2012

[2] W.-C. Fang, T. Shaw, J. Yu, B. Lau, and Y.-C. Lin, "Parallel Morphological Image Processing with an Opto-electronic VLSI Array Processor," in Proceedings of the 1993 IEEE International Conference on Acoustics, Speech, and Signal Processing: Plenary, Special, Audio, Underwater Acoustics, VLSI, Neural Networks - Volume I, Washington, DC, USA, 1993, pp. 409–412.

[3] Theys M D, Morn R M, Allemang M D, Siegel H J, "Morphological Image Processing on Three Parallel Machines ", In the proceedings of Sixth Symposium on the Frontiers of Massively Parallel Computing, 1996. Proceedings Frontiers '96.

[4] P. Kumar, K. Palaniappan, A. Mittal, and G. Seetharaman, "Parallel Blob Extraction Using the Multi-core Cell Processor," in Advanced Concepts for Intelligent Vision Systems, vol. 5807, J. Blanc-Talon, W. Philips, D. Popescu, and P. Scheunders, Eds. Springer Berlin Heidelberg, 2009, pp. 320–332.

[5] Slabaugh, G.G.; Boyes, R.; Xiaoyun Yang, "Multicore Image Processing with OpenMP [Applications Corner]," in Signal Processing Magazine, IEEE , vol.27, no.2, pp.134-138, March 2010

[6] J. Plaza, R. Pérez, A. Plaza, P. Martínez, and D. Valencia, "Parallel morphological/neural processing of hyperspectral images using heterogeneous and homogeneous platforms," Cluster Computing, vol. 11, no. 1, pp. 17–32, 2008.

[7] H. Sugano and R. Miyamoto, "Highly Optimized Implementation of OpenCV for the Cell Broadband Engine," Comput. Vis. Image Underst., vol. 114, no. 11, pp. 1273–1281, Nov. 2010.

[8] Baig, H.; Jeong-A Lee; Jieun Lee. Performance evaluation of CPU-GPU and CPU-only algorithms for detecting defective tablets through morphological imaging techniques. 7th Iberian Conference on Information Systems and Technologies (CISTI), 2012, Pages: 1 - 6

[9] Teng Li; Yong Dou; JingFei Jiang; Jing Gao. Efficient parallel implementation of morphological operation on GPU and FPGA. International Conference on Security, Pattern Analysis, and Cybernetics (SPAC), 2014,

[10] Lipeng Wang; Hanbin Wang. Implementation of a Soft Morphological Filter Based on GPU Framework. 5th International Conference on Bioinformatics and Biomedical Engineering, (iCBBE) 2011, Pages: 1 - 4

[11] Crozet, S.; Geraud, T.. A first parallel algorithm to compute the morphological tree of shapes of nD images. IEEE International Conference on Image Processing (ICIP), 2014, Pages: 2933 – 2937.

[12] Valencia, D.; Plaza, A. Efficient implementation of morphological opening and closing by reconstruction on multi-core parallel systems. First Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing, 2009. WHISPERS '09. Pages: 1 - 4.