ISSN 2320-6802

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY

WINGS TO YOUR THOUGHTS.....

A Novel Open Source Morphology Using GPU Processing With LTU-CUDA

G. Jagannathan[1], R. Subash[2], K. Senthil Raja[3]

[1]Ranganathan Engineering College,
Coimbatore-641109, India
jaganjagan.91@gmail.com

[2]Ranganathan Engineering College,
Coimbatore-641109, India
subash.f1@gmail.com

[3]Ranganathan Engineering College,
Coimbatore-641109, India
rksenthilraja27@gmail.com

Abstract: *A mathematical morphology is used as a tool for extracting image components that are useful in the representation and description of region shape. The mathematical morphology operations of dilation, erosion, opening, and closing are important building blocks of many other image processing algorithms. The data parallel programming provides an opportunity for performance acceleration using highly parallel processors such as GPU. NVIDIA CUDA architecture offers relatively inexpensive and powerful framework for performing these operations. However the generic morphological erosion and dilation operation in CUDA NPP library is relatively naive, but it provides impressive speed ups only for a limited range of structuring element sizes. The vHGW algorithm is one of the fastest for computing morphological operations on a serial CPU. This algorithm is compute intensive and can be accelerated with the help of GPU. This project implements vHGW algorithm for erosion and dilation independent of structuring element size has been implemented for different types of structuring elements of an arbitrary length and along arbitrary angle on CUDA programming environment with GPU hardware as GeForce GTX 480. The results show maximum performance gain of 20 times than the conventional serial implementation of algorithm in terms of execution time.*

Index Terms *- Morphological image processing, erosion, dilation, GPU, NVIDIA, CUDA.*

1. INTRODUCTION

1.1. MORPHOLOGY

Morphological image processing is powerful non-linear image analysis tool for the analysis of spatial structure based on pre-defined spatial structures known as structuring elements [1]. The fundamental operations of morphological image processing are erosion and dilation [10, 12, and 13]. The objective of this work is to produce a freely available GPU capability for fast morphological operations so that fast GPU processing can be readily available to those in the morphological image processing community [2]. CUDA [7] is a device architecture developed by NVIDIA that allows general parallel computation to be performed on their CUDA enabled GPU [8] graphics cards. NVIDIA provides the NPP library at no cost that contains fundamental morphological operations.

1.2. OPERATIONS

1.2.1 Dilation

Dilation "grows" or "enlarges" objects in a binary image [3]. The manner and extend of this growth image is controlled by the structuring element (fig 1.1).

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \Phi\}$$

This equation is based on reflecting B about its origin and shifting this reflection by z. The dilation of A by B then is the set of all displacements z, such that and A overlap by at least one element.

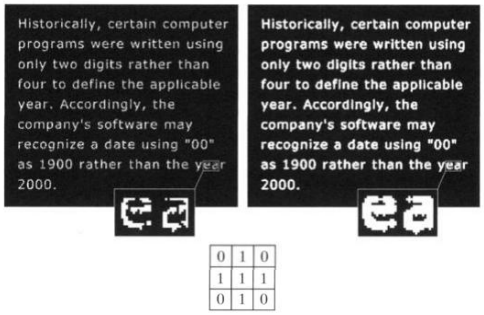


Figure 1.1: Brocken character image with max length of gaps 2 pixels, dilation to bridge gaps.

1.2.2 Erosion

Erosion "shrinks" or "removes" objects in a binary image [3]. With A and B as sets, the erosion of A by B is defined as (fig 1.2)

$$A - B = \{z \mid (B)_Z \subseteq A\}$$

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

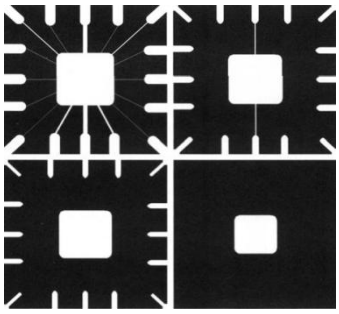


Figure 1.2: Example of erosion

The erosion of A by B is the set of all points z such that B, translated by z, is contained in A, where set B is a [4] structuring element.

A 486x486 binary image[5] of a wire-bond mask and images eroded using square structuring elements of sizes 11x11, 15x15, and 45x45 pixels whose components were all ones in Fig 1.4. Two vertical lines were tinned but not removed since their width was greater than 11.

In image processing [5] applications, dilation and erosion are used most often in various combinations. An image undergoes a series of dilations and/or erosions using the same or different structuring elements.

1.2.3 Opening and Closing

Opening generally smoothes the contour of an object and eliminate thin protrusions [3]. The opening of a set *A* by structuring element *B* is defined as

$$A \circ B = (A - B) \oplus B$$

Therefore, the opening *A* by *B* is the erosion of *A* by *B*, followed by a dilation of the result by *B*.

Closing also tends to smooth sections of contours but fusing narrow breaks and long, thin gulfs and eliminating small holes and filling gaps in the contour. The closing [3] of a set *A* by structuring element *B* is defined as

$$A \bullet B = (A \oplus B) - B$$

Therefore, the closing *A* by *B* is the dilation of *A* by *B*, followed by an erosion of the result by *B*.

1.3 CONTRIBUTION

This paper presents a freely available implementation of fast morphological erosion and dilation [14] based on CUDA/NPP using the vHGW algorithm. Erosion and dilation are implemented for a base set of structuring elements from which larger structuring elements can be constructed via structuring element decomposition. A test framework using MATLAB is provided in LTU-CUDA. MATLAB is not known as a particularly fast implementation environment for programming it demonstrates vHGW constant time performance as the structuring element size increases. Therefore performance results from MATLAB provide a useful relative comparison.

2. ALGORITHM

Here, generalized vHGW algorithm [9] is also based on splitting the input pixel to overlapping segments of size 2p-1. The angle of structuring element is accepted from user. This value decides which pre-processing [13] is to be performed.

vHGW ALGORITHM

1 Image rows are partitioned into segments of length p with $(p-1)/2$ columns of overlap on each side to form a window of size $2p-1$, centered at $p-1, 2p-1, 3p-1, \ldots$

2 For each pixel $k = 0..(p-1)$ in a given window w, a suffix max array R is created for the pixels left of center
$R[k] = max(w[j]) : j = k..(p-1)$
and a prefix max array S is created for the pixels right of cent-re $(p-1)..(2p-2)$,
$S[k] = max(w[p-1+j]) : j = 0..k$

3 For each pixel $\frac{(p-1)}{2} <= j < p + \frac{(p-1)}{2}$ in w (the segment of length p) the dilation result is
$result[j] = max(R[m], S[m])$, where $m = j - \frac{p-1}{2}$

- Horizontal structuring element
- Vertical structuring element
- Structuring element along arbitrary angle

The result of a dilation using a square structuring element of various sizes up to 63x63, showing constant time computation regardless of structuring element size on an image of 1024x1024.

3. IMPLEMENTATION

The project was implemented on a NVIDIA Geforce GTX 480 with CUDA 480 cores running NVIDIA driver under Windows 7 (64 bits) operating system.

Some important features of Geforce GTX 480 affecting implementation are specified below

- Computing Capability: 2.0x
- Graphics Clock: 607 MHz
- Processor Clock: 1215 MHz
- Maximum Dimensionality: 3

Then the implementation continues with the steps of vHGW algorithm for dilation with 1D structuring element of size p=2N+1. Erosion follows the same process using minimum value arrays. The total number of pixels to be padded is (p-1)/2 at each side of rows, left and right, in order to calculate prefix and suffix values smoothly.

- Dilation: pad value=-128
- Erosion: pad value=127

This project implements vHGW algorithm with shared memory arrays for the max arrays *prefix[i]* and *suffix[i]*, and 2 threads per window w, one for each max array. Shared memory usage is an important issue in the implementation.

If multiple threads are used for single result then it is necessary to have inter-thread communication. Shared memory is however limited in size making it only viable for smaller images and structuring elements. Multiple windows can be addressed in one CUDA block to achieve good thread utilization and scheduling.

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

3.1 SUPPORTED OPERATION

LTU-CUDA provides the following operations and structuring elements primitives from which a variety of other shapes can be constructed including squares, rectangles, diamonds, octagons, and 8-sided approximations to a circular disc.

1. Erosion and dilation using 8 bit or 32 bit images.
2. Flat horizontal, vertical, and diagonal degree line structuring elements using GPU vHGW.
3. Two 3 by 3 mask structuring elements implemented using loop free code providing a 10–20% speed improvement over NPP.
4. Other structuring elements (non flat, non-filled) are currently implemented using a generic structuring element kernel equivalent to the NPP generic algorithm.

3.2. TESTING CONDITIONS

MATLAB uses structuring element decomposition for many areas based structuring elements, including the square and disc structuring elements used in this work. The following MATLAB [11] functions and structuring element decompositions were used with LTU-CUDA being tested with the equivalent decomposition [6].

- strel('line', angle, length) produces linear structuring elements with angle equal to create horizontal, vertical and 45 degree diagonal lines.
- strel('square', side_length) produces a square structuring element, which is decomposed into two structuring elements, horizontal and vertical lines of length size_length.
- strel('disc', radius) produces an 8-sided approximation of a circular disc decomposed into a combination of horizontal, vertical and degree linear structuring elements.

As we use structuring element decomposition [10] for MATLAB and LTU-CUDA it is only reasonable to apply structuring element decomposition for NPP where this would improve performance.

4. RESULTS

The modified vHGW algorithm was tested on different images. Only greyscale image type was considered (fig 4.2). Due to simple format, header of image was read and written step by step in program. Otherwise, Image library e.g. Free Image also could be used for this purpose. The colour depth or bits per pixel was considered 8bits per pixel.

Image size we have considered ranges between 512 * 512 and 4098 * 4098. Also, different types of structuring elements have been applied for these images. The performance gain of minimum 2x and maximum of 20x has been achieved.

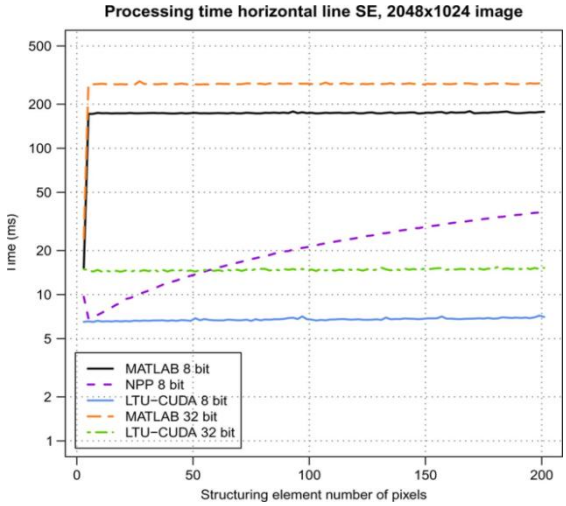
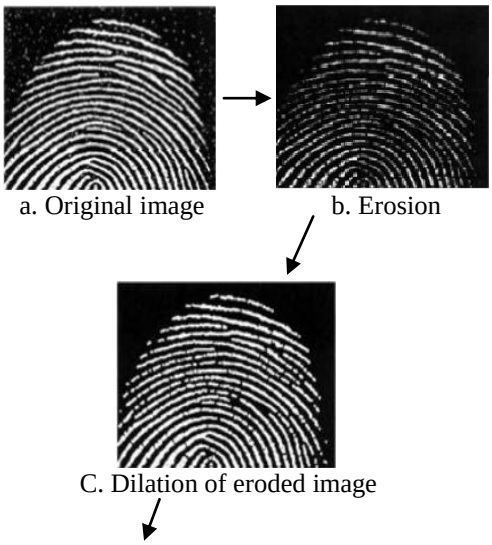


Figure 4.1: performance comparison over matlab

The Morphological Image Processing [10], [12] enhances the degraded noisy and / or incomplete latent fingerprints. In physics and related fields, computer techniques routinely enhance images of experiments in areas such as high-energy plasmas and electron microscopy. Similarly successful applications of image processing concepts can be found in astronomy, biology, nuclear medicine, law enforcement, and defence. The results for the vHGW vertical line structuring element in Fig 4.1 is not as desired and show interesting results for LTU-CUDA and MATLAB [11].

LTU-CUDA shows a significant dependency on structuring element size although still significantly better than NPP, and MATLAB shows a substantially faster result (nearly 4 times faster) than MATLAB processing horizontal lines. Both results are related to memory handling. CUDA is more preferred programming environment to program graphical processing units (GPUs). Algorithms in morphological image processing have large scope for parallelization.

NVIDIA [15] provides the NPP library at no cost that contains fundamental morphological operations.



INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

d. Dilation e. Closing

Figure 4.2: Noise removal using erosion and dilation

The presented results were produced using morphological erosion on a NVIDIA GTX 470 [15] with 448 CUDA cores running NVIDIA driver 270.41.19, AMD Athlon II X4 620 CPU running at 3067 MHz, and with MATLAB version 7.10.0 R2010a [11]. Performance results were generated under linux 32 bit (ubuntu 10.04LTS).

The results for vHGW horizontal line structuring element in Fig. 4.3 show the performance of the vHGW algorithm both in LTU-CUDA and MATLAB independent of structuring element size.

The results for the vHGW vertical line structuring element in Fig 4.3 is not as desired and show interesting results for LTU-CUDA and MATLAB.

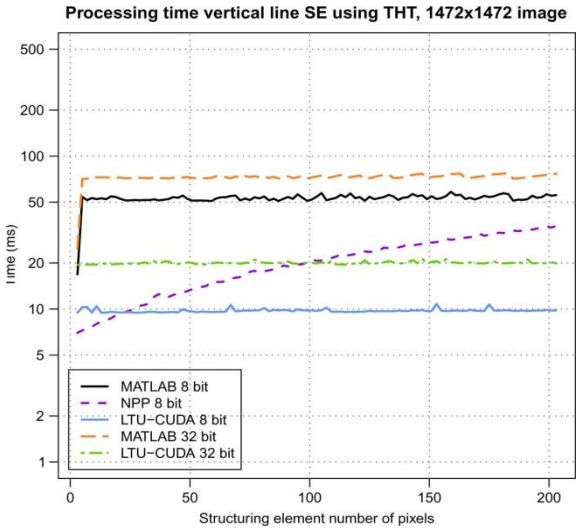


Figure 4.3: Processing time for structuring element

Performance results (fig 4.3) for NPP and LTU-CUDA include not only the morphological image processing time, but also all operations required to replace the functionality provided by MATLAB including memory allocations and transfers to the GPU and back. Fig shows improve performance over MATLAB.

Performance Increase : LTU-CUDA as a multiple of NPP and MATLAB

structuring element	MATLAB 8		NPP 8bit		MATLAB 32	
	mean	stdev	mean	stdev	mean	stdev
horizontal line	25.5	2.38	3.18	1.26	18.5	1.73
vertical line	4.73	1.38	1.66	0.25	3.29	0.58
vertical-THT line$^{\phi}$	5.45	0.42	2.13	0.83	3.64	0.25
square THT †	13.0	1.21	3.01	1.34	15.5	1.43
diagonal line	33.1	17.3	9.22⋆	4.32⋆	13.1	7.21
disc ‡	54.1	23.2	14.7	7.84	40.3	18.3

Figure 4.4: Performance increase of LTU_CUDA

LTU-CUDA shows a significant dependency on structuring element size although still significantly better than NPP, and MATLAB shows a substantially faster result (nearly 4 times faster) than MATLAB processing horizontal lines. Both results are related to memory handling. The LTU-CUDA results in Fig. 4.4 are memory bound rather than computationally bound as a result of how the image is laid out in CUDA global memory CUDA memory bottleneck for vertically oriented operations is of interest.

5. CONCLUSION

Graphical Processing Units (GPUs) are available with large computing power. The computing system gets large number of Giga floating point operations per second (Gigaflops) for lesser cost than traditional high performance machines and they have large market presence to develop product based on them. CUDA is more preferred programming environment to program graphical processing units (GPUs). Algorithms in morphological image processing have large scope for parallelization. There is need for accelerating these algorithms. The vHGW algorithm has been implemented not only for horizontal, vertical and diagonal structuring elements, but also elements along the arbitrary line and of the arbitrary length. This large performance improvement was achieved with CUDA implementation. The performance improvement varies with the input size of image, size and shape of a structuring element. Large performance gain is achieved with larger size of images due to occupancy factor of GPUs.

6. FUTURE WORK

Gray scale morphology has been implemented using CUDA. This work can be extended for color morphology. Morphological image processing is an area which possesses a lot of potential for parallel computing. Similar work can be applied on various algorithms in this area. Only flat structuring elements are considered here. The work can be extended for non flat elements also. Applications in this area can reduce the large amount of time for computation. With the advancements in GPUs, it is possible to get much higher performance gain.

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

REFERENCES

[1] R.C.Gonzalez, R.E. Woods, "Digital Image Processing", 2nd Edition.
[2] V.Danell, Matthew J. Thurley, "Fast Morphological Image Processing Open Source Extensions with CUDA", IEEE Journal of Signal Processing, Vol. X, No. X, XXXXXX 2012
[3] J.Gil and R.Kimmel, "Efficient dilation, erosion, opening and closing algorithms", Pattern Analysis and machine Intelligence, vol. 24, no. 12, pp. 1606-1617, dec. 2002
[4] http://www.cis.rit.edu/class/simg782/lectures /lecture_03/lec782_05_03.pdf
[5] www.wikipedia.org/mathematical_morphology
[6] Pierre Sollie, Edmond J. Breen, and Ronald Jones "Recursive implementation of erosions and dilations along discrete lines at arbitrary angles", Pattern analysis and machine intelligence, IEEE Transactions on, vol 18, no 5, pp. 562-567, may 1996
[7] "optimising CUDA – part II", NVIDIA developer website, 2009
[8] http://en.wikipedia.org/wiki/GPU
[9] Luke Domanski, Pascal Vallotton, Dadong Wang, "Parallel vHGW image morphology on GPUs using CUDA", CSIRO, Mathematical and Informational Sciences, Biotech Imaging
[10] E. R. Dougherty and R.A. Lotufo, „hands on Morphological Image Processing. SPIE - The International Society for Optical Engineering, 3002, vol. tt59.
[11] MATLAB, "version r2010a", Natick Massachusetts 2010.[Online].Available: http://www.mathworks.com
[12] D. B. Kirk and W. mei W. Hwu, Programming Massively Parallel Processors: A Hands-on Approach (Applications of GPU Computing Series). Morgan Kaufmann, 2010.
[13] M. Vanherk, "A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels," *Pattern Recogn. Lett.*, vol. 13, no. 7, pp. 517–521, Jul. 1992.
[14] M.Van Droogenbroeck, "On the Implementation of Morphological Operations", Math. Morphology and its applications to image processing, J. Serra and P. Sollie, eds. Dordrecht: Kluwer Academic Publishers, 1994, pp. 241-248
[15] NVIDIA, "NVIDIA CUDA C Programming Guide – 4.2", NVIDIA developer website, April 2012, [Online]. Available: http//developer.download.nvidia.com