Efficient Binary Morphological Algorithms on a Massively Parallel Processor

Andreas I. Svolos, Charalampos G. Konstantopoulos, and Christos Kaklamanis
Computer Engineering & Informatics Dept.
Univ. of Patras, GR 265 00 Patras, Greece
svolos@cti.gr

Abstract

One of the most important features in image analysis and understanding is shape. Mathematical morphology is the image processing branch that deals with shape analysis. The definition of all morphological transformations is based on two primitive operations, i.e. dilation and erosion. Since many applications require the solution of morphological problems in real time, researching time efficient algorithms for these two operations is crucial. In this paper, efficient parallel algorithms for the binary dilation and erosion are presented and evaluated for an advanced associative processor. Simulation results indicate that the achieved speedup is linear.

Keywords: *mathematical morphology, dilation, erosion, massively parallel processor, associative processor, hypercube*

1. Introduction

Shape is one of the most significant features employed in the field of image analysis and understanding. It has been widely used for the identification of the various objects in a scene, the segmentation of an image into its constituent homogeneous regions, or even the description of the elements of which a textured surface is composed. These problems are very frequently encountered in image processing applications, such as industrial machine vision inspection and recognition [5], [9], visually guided robot [9], biomedical image analysis [2] etc.

Mathematical morphology is the branch of image processing that deals with shape analysis and characterisation. It is based on set theory [8], [13]. The image is considered to be a set of ordered N-tuples $E^N \subset Z_+^N$, where Z_+ is the set of all positive integers including 0. The first two dimensions give the Cartesian co-ordinates of the pixels, whereas the other $N-2$ represent the information associated with each pixel. In the case of the binary images this information is implicit. A binary image is represented by a set of ordered pairs $E^2 \subset Z_+^2$ which correspond to pixels belonging to the objects of the image. By convention, these pixels have value 1. The rest of the pairs (pairs not in set E^2) correspond to background pixels and are assumed to have value 0. A large number of problems encountered in the analysis of grey level images turn out to be problems in the binary image field. Binary mathematical morphology [4] is the branch that deals with shape analysis in binary images. Mathematical morphology is also extended to three or more dimensions (e.g. grey level images).

Morphological operations are usually employed in the early stages of image processing applications, so it is essential these operations be completed as quickly as possible. Also, many of the image processing applications mentioned in a previous paragraph are time-critical and therefore, require the solution of morphological problems in real time. Thus, the efficient implementation of morphological operations, in terms of computational time, is very important.

In order to achieve real-time performance, hardwired approaches such as the ASICs (Application Specific Integrated Circuits) have been developed [17]. However, they impose several restrictions such as limiting the size and shape of the structuring element. Additionally, they are application-specific and so, they cannot be employed in the execution of image processing functions other than the morphological ones. A number of parallel architectures have also been employed in order to achieve real-time computation of the morphological operations. One type is the array processor with a 2D mesh interconnection of the processing elements [3], [14]. In this architecture, each processing element may correspond to a different pixel of the processed image, while the nearest-neighbour network allows the fast computation of transformations in the local neighbourhood of each pixel. However, the bit serial processing performed in each element of a massively parallel array, imposes constrains on the full utilisation of the inherent parallelism in the morphological transformations. Another type of architecture that has been widely used is the pipeline architecture [5], [9]. In this architecture, the image is transmitted serially through a number of processing elements each of which ap-

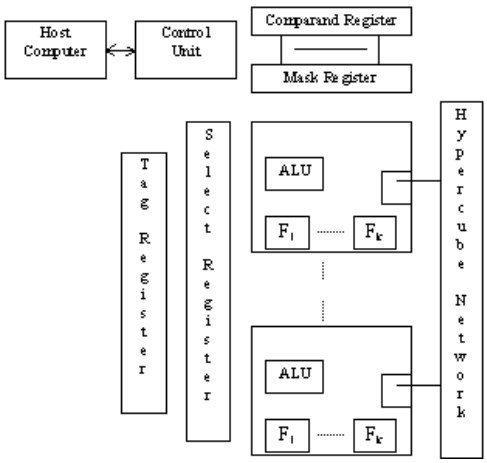

Figure 1. The employed advanced associative processor architecture.

plies a different transformation to the data, in parallel with the others. This architecture provides parallelism regarding the operations that need to be performed on the image data but not the data themselves, which are processed serially for each operation. This is a serious drawback in many cases of morphological processing, in which the amount of data that need to be processed is very large, while the number of applied transformations is usually small. A different approach which belongs to the VLIW (Very Long Instruction Word) architectures is the Mediaprocessor [17]. This is actually a DSP (Digital Signal Processor) with several parallel arithmetic and load/store units which can offer fine-grained parallelism in image processing applications. However, in morphological operations the parallelism is limited to only the structuring element giving a computational time complexity of at least $\Omega(N^2)$, where $N \times N$ is the size of the processed image.

Recently, a new advanced architecture that belongs to the associative processors has been developed for image processing [6] (see Figure 1). Due to the rapid evolution of VLSI technology and the large regularity of this type of architecture, the cost of manufacturing associative memories with parallel access not only to the memory words but also to the individual bits in each word has been dramatically decreased [11]. It has been also shown to be feasible the addition of arithmetic/logic circuits in each memory word, transforming it into a processing element [12]. These circuits are able to execute logical operations, such as OR and AND, in parallel on the bits in each processing element, thereby allowing the implementation of a word-parallel, bit-parallel, massively-parallel computing system. Additional circuits can be incorporated allowing the interconnection of the processing elements in various topologies, such as the 2D mesh or the hypercube. Konstantopoulos et al. [6] selected the hypercube network among other geometries due to its generality and its proven efficiency in a large number of applications, especially in image processing. Careful design of the relevant image processing algorithms can lead to minimisation of the hardware complexity of the network interface in each processing element, e.g. the number of required buffers.

The basic operations in mathematical morphology are dilation and erosion. These operations are defined in the next section. The definition of all the other morphological transformations are based on these two primitive operations. In this paper, two efficient parallel algorithms for the binary dilation and erosion are presented and evaluated for the associative processor described in [6]. It is shown that these algorithms can take full advantage of the capabilities of this advanced architecture and that the proposed associative processor is more efficient than other architectures for the binary morphological transformations. It is also shown that there is a trade-off between the hardware complexity of the processing elements and the communication cost of the algorithms.

The proposed algorithms can support any arbitrary structuring element size and shape. Certain convex structuring elements can be decomposed into a series of smaller elements in order to reduce the computational time [16]. These decomposition algorithms can easily be incorporated in the proposed algorithms to further improve the time performance.

2. Binary mathematical morphology

2.1. Definitions

As we said in the previous section, there are two primitive operations from which all the other morphological operations can be derived: dilation and erosion. Similarly, the definition of these two operations can be based on three elementary set operations, namely the union and intersection of two binary images as well as the translation of a binary image. Let $A, B \subset R^2$ be two sets of ordered pairs that represent two binary images. The translation of binary image A by vector $b = (i, j)$, which is denoted by A_b , is defined as:

$$A_b = \{x + b \in R^2 | x \in A\} \quad (1)$$

The dilation of binary image A by binary image B, denoted by $A \oplus B$, is defined as:

$$A \oplus B = \{x \in R^2 | \exists a \in A, \exists b \in B, x = a + b\} \quad (2)$$

In all morphological operations, the first operand (set A) represents the image to be processed while the second

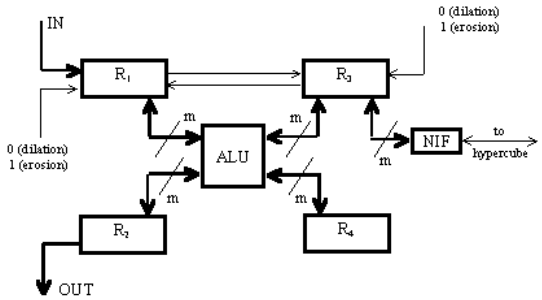

Figure 2. Structure of the processing elements.

operand (set B), which is commonly referred to as the structuring element, represents a shape parameter, i.e. an elementary shape such as a circle. It is proven that

$$A \oplus B = \bigcup_{b \in B} A_b \text{ [4]} \tag{3}$$

This property, which shows that the dilation operation can be broken down into a sequence of set unions, is very useful in the parallel computation of dilation.

The erosion of binary image A by the structuring element B, denoted by $A \ominus B$ is defined as:

$$A \ominus B = \{x \in R^2 | x + b \in A, \forall b \in B\} \tag{4}$$

It is proven that

$$A \ominus B = \bigcap_{b \in B} A_{-b} \text{ [4]} \tag{5}$$

Thus, similarly to the dilation operation the erosion of A by B can be decomposed into a number of set intersections. This decomposition is very important and will be used for its parallel computation.

2.2. Parallel algorithms for binary dilation and erosion on the associative processor

Before we present the parallel algorithms for dilation and erosion, we need to show how a 2D mesh network can be mapped into the hypercube network of the parallel architecture employed here. The reason is that the 2D mesh topology is the most efficient for the image translation needed at each computational step. Saad and Schultz [10] proved that a 2D mesh network with nearest-neighbour and wraparound links (torus topology) can be embedded in a hypercube with dilation 1 (dilation is defined as the maximum number of edges of the host geometry that are needed for mapping one edge of the embedded geometry). Actually, a 2D mesh network with size $N \times N$, $N = 2^n$ can be mapped into a 2n-dimensional hypercube without any collisions on the links. The mapping is based on the Binary Reflected Gray Code (BRGC).

In addition, we need to have a closer look into the internal structure of each processing element of the employed associative processor (see Figure 2). The general outline of this architecture is shown in Figure 1. Konstantopoulos et al. [6] describe its general characteristics in greater detail. Each element contains 4 registers m bits long, namely R_1, R_2, R_3, and R_4. The ALU in each element allows the parallel computation of the logical operations AND and OR between registers (R_1, R_2), (R_1, R_3), (R_2, R_3), and (R_3, R_4). In addition, registers R_1 and R_3 constitute a shifter capable of performing left as well as right shifts. Finally, the processing elements can communicate over the hypercube network through their R_3 register. Each R_3 register is connected to the network via the NIF (Network Interface) module.

Let A be the binary image to be processed with size $N \times N$, $N = 2^n$. Also, let B be the structuring element. Then, the associative processor needs to be composed of at least $P = \frac{N^2}{m}$ processing elements, each of which stores a horizontal segment of m bits of image A. These bits are initially stored in register R_1. As we said in Section 2.1, properties 3 and 5 are employed for the parallel implementation of binary dilation and erosion, respectively. Each computational step consists of an image translation dictated by the structuring element followed by a logical operation (OR for dilation and AND for erosion) between the translated image and the partial result computed so far, which implements the corresponding set operation (union or intersection). Registers R_1 and R_3 perform the necessary horizontal translations by shifting their contents, accordingly. We mentioned earlier that these two registers form a shifter 2m bits long. Register R_3 is employed solely for the efficient utilisation of the hypercube network. Without this register, each horizontal translation of image A by one pixel would require the transmission of messages of only one bit over the network. However, a communication step over the hypercube costs much higher, in terms of time, than a register shift. Register R_3 in each processing element is employed to temporarily hold the portion of the translated image which is to be stored in register R_1 of its nearest neighbour determined by the direction of translation. In this way, register R_3 acts as a buffer. Only when this buffer is full, are its contents sent over the network to its nearest neighbour. Then, the contents of R_1 in each processing element are updated in order to keep up with the performed translations. Thus, the number of messages that need to be transmitted is smaller and therefore, the use of register R_3 as a buffer reduces the communication cost of the algorithms.

Register R_2 is employed to store the partial image computed so far. When the algorithm concludes, the contents of R_2 in all processing elements form the final image (the

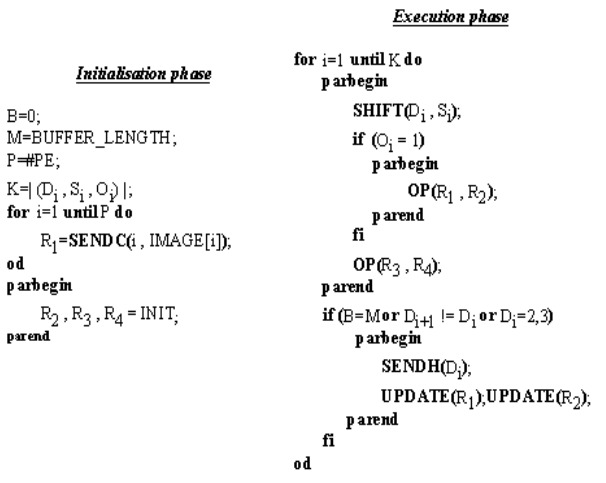

Figure 3. The proposed algorithms for the binary dilation and erosion.

image that results from the application of the morphological operation). Finally, register R_4 is necessary to temporarily store partially computed results which are transmitted along with the contents of register R_3, in order for R_2 to be updated. Therefore, this register plays the same role as register R_3.

The structuring element B is stored and processed in a host computer to which the associative processor is interfaced via the control unit (see Figure 1). It is decomposed into a sequence of triplets (D,S,O), where D is the direction of translation, S is the number of pixels by which the image has to be moved in direction D, and O is a flag which determines whether the appropriate logical operation between the partially computed image (R_2) and the translated image (R_1) needs to be performed in the current computational step (an efficient algorithm for performing this decomposition, which minimises the required shifts, will be presented elsewhere). The computed triplets (D,S,O) are then transferred and stored inside the control unit of the associative processor. Since the structuring element has small dimensions (usually much smaller than the dimensions of the image to be processed), the required memory space is small. The control unit translates the triplets into the appropriate sequence of instructions and data. The word-parallel, bit-parallel access capability of the associative architecture enables the rapid transfer of these data and instructions to all processing elements for execution.

The proposed algorithms are based on a synchronous computational model. They are composed of alternating non-overlapping steps of computation and communication. Figure 3 shows the initialisation phase and the execution phase of the algorithms for the binary dilation and erosion. In the pseudocode of the initialisation phase, variable B shows how much of the buffer (register R_3) has been used after the last time it was emptied. Variable M gives the length of the registers (m bits) while variable P gives the number of processing elements. Also, variable K gives the length of the sequence of the (D,S,O) triplets. Each such triplet corresponds to a single computational step and therefore, this number determines the number of computational steps of the algorithms. Function **SENDC(·)** sends the corresponding portion of the initial image to each processing element. Finally, registers R_2, R_3, and R_4 initially get the value INIT, which is different for each of the two binary operations, i.e. it is 0 for dilation and 1 for erosion.

In the pseudocode of the execution phase, function **SHIFT(·)** performs the image translation dictated by the (D_i, S_i) pair, where i is the current computational step. Function **OP(·)** performs the appropriate parallel logical operation (OR for dilation and AND for erosion). This sequence of execution is repeated until either register R_3 is full (B=M) or a translation of the image in a different direction is needed ($D_{i+1} \neq D_i$). When one of these two conditions is fulfilled, the processing elements perform a communication step by executing function **SENDH(·)** which sends the contents of their registers R_3 and R_4 over the hypercube network to their nearest neighbour determined by direction D_i (possible directions are left, right, up, and down). Finally, function **UPDATE(·)** uses the received data in the processing elements to update the contents of their registers R_1 and R_2, accordingly[1]. The language construct **parbegin**. . .**parend** encloses the instructions which are to be executed by all processing elements. All the other instructions are executed by the control unit.

3. Results

In this section, simulation results are presented from the comparison of the time performance of a serial processor, a massively parallel array processor with a 2D mesh interconnection network, and the associative processor employed in this paper. Figure 4 a) shows the speedup of the associative processor over the serial processor in binary morphological processing for various image sizes N. It also shows the number of processing elements P of the associative processor that were needed in each simulation, i.e. for each different image size. Figure 4 b) gives the corresponding speedup over the array processor. In this case, the speedup is independent of the image size, as will be explained later on. The binary dilation algorithm was employed in producing all simulation results. Binary erosion gave similar performance results.

[1]The implementation of the above functions is beyond the scope of this paper.

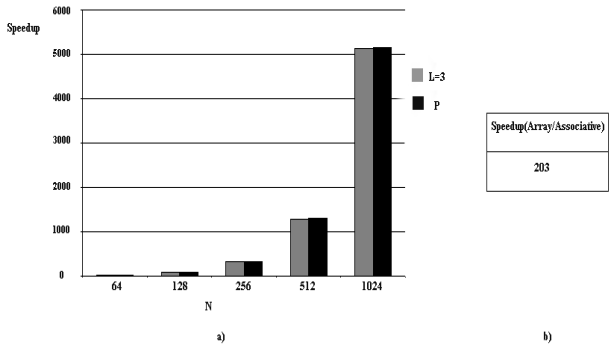

Figure 4. a) The speedup of the associative over the serial processor for various image sizes along with the number of processing elements. b) The speedup of the associative over the array processor.

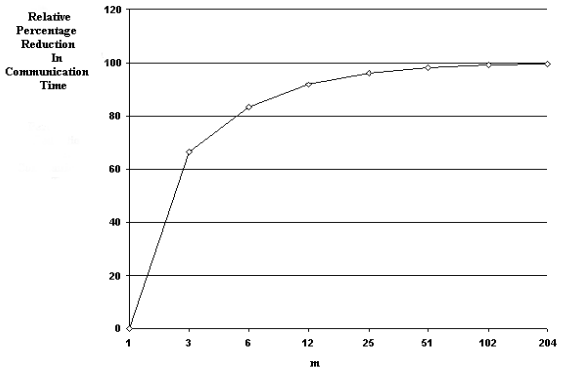

Figure 5. The percentage reduction of the total communication time relative to its maximum value against the buffer length m.

All architectures were simulated on the Parallaxis simulator [1], [15]. This simulator was designed and implemented at the University of Stuttgard, Germany. This is actually a language for data-parallel programming which is also machine independent across different SIMD architectures. However, when the code is compiled by a conventional C compiler, simulation code is obtained. An interesting feature of this language is that the interconnection network of the parallel architecture can be easily specified by the programmers.

The size of the structuring element was 5×5. Similar results were taken for other sizes, e.g. 3×3 and 7×7. Both the associative and the array processor contained the same number of processing elements P in each simulation. P was scaled up according to relation $P = N \cdot \lceil \frac{N}{m} \rceil$, where m is the number of bits of the processed image stored in each processing element of the parallel architectures. Parameter m was kept constant in all simulations, therefore the scaling of the processing power was directly proportional to the size of the processed image. Also, m gives the length of buffers in each processing element of the associative processor. In deriving the illustrated results, m was 204 bits.

Figure 5 shows how m affects the communication time performance of the proposed dilation algorithm on the associative processor. Actually, it presents the percentage reduction of the communication time in each simulation relative to the worst such time, as m increases. The size of the analysed image was 256×256, while the size of the structuring element was again 5×5.

4. Discussion

From Figure 4 a), we can clearly see that the speedup derived from the employment of the associative processor over the serial processor is very close to the number of processing elements P used in each simulation, i.e. for each different image size. Given P processors we would like any parallel algorithm to run P or even $\Theta(P)$ times as fast as the best corresponding serial algorithm. In this case, according to [7], the parallel algorithm achieves linear speedup. No algorithm can ever attain greater than linear speedup. The simulation results clearly indicate that the proposed parallel binary dilation algorithm running on the associative processor has a $\Theta(P)$ speedup. Therefore, it can be considered near optimal in terms of computational time. We also notice that as the size of the binary image increases the speedup increases greatly. This is due to the fact that the proposed associative architecture exploits fully the intrinsic parallelism in this image processing task.

Comparing the time performance of the parallel dilation algorithm on the associative processor with the corresponding performance of the same algorithm on the array processor, the former was found to be about 200 times faster in all simulations (see Figure 4 b)). Since all the other parameters, such as the number of processing elements P and the storage capacity m, were the same for both processors, the above speedup resulted only from the parallel computation of the logical operations OR and AND in the processing elements of the associative processor. In this case, the size of the processed image has no effect on the speedup. The derived speedups were not significantly affected by the size of the structuring element for small sizes. This is the usual case, since even when the structuring element is large, there exist algorithms that can decompose it to smaller sizes [16].

Parameter m plays an important role in the determination of the computational complexity of the proposed algorithms and especially its constituent communication time. Figure 5 shows this relationship in the form of the percentage reduction of the communication time relative to the maximum communication time, as m increases. In the case of the associative processor, m gives the length of the associated buffers in each processing element. The longer the buffers the smaller the number of messages that are needed to be sent over the hypercube network during the execution of the binary dilation algorithm. This results in a smaller total communication processing time (the time needed to initiate a communication step and prepare the data for transmission). This time is actually the dominant factor in the communication delay of the algorithms. In Figure 5, the maximum communication time happens when m=1, that is when the network is engaged for each one-bit shift of the processed image. On the contrary, when m=204 the percentage reduction of the communication time is 99.52%. However, by increasing m the circuit complexity of each processing element is also increased (the processing element becomes larger). Therefore, there is a trade-off between the hardware complexity of the processor and the communication complexity of the algorithms.

In all simulations for the estimation of the speedups the maximum value of m was 204 bits. This value has not been chosen randomly. According to Scherson et al.[12], it is feasible to implement an associative processor chip where each processing element consists of 1K bits of memory split into 5 bit planes and the associated control and networking circuitry. This gives a register length of up to 204 bits.

5. Conclusions

The results presented in Section 3 showed that the massively parallel associative processor employed in this paper is the most efficient architecture for morphological processing. Clearly, the presented parallel algorithms for binary dilation and erosion exhibited a near optimal speedup. This was proven to be due to the fact that these algorithms take full advantage of the unique features of this advanced architecture. More specifically, the algorithms can perform efficient image translations over the single-stage hypercube interconnection network of the associative processor by mapping the 2D mesh topology into this network. Also, the ability to perform logical operations in parallel on the bits in each processing element leads to an optimal computational complexity. Moreover, the incorporation of two additional registers in each processing element makes the utilisation of the hypercube more efficient and thus, it reduces the communication complexity. Finally, it was shown that there is a trade-off between the communication complexity of the presented algorithms and the hardware complexity of each processing element, which is related to its buffer length.

References

[1] http://www.informatik.uni-stuttgart.de/ipvr/bv/p3.

[2] J. Dengler, S. Behrens, and J. F. Desaga, "Segmentation of microcalcifications in mammograms", *IEEE Trans. on Medical Imaging*, vol. 12, pp. 634–642, 1993.

[3] M. Duff, "Parallel processors for digital image processing", *Advances in Digital Image Processing*, Plenum, New York, pp. 265–279, 1979.

[4] R. M. Haralick, S. R. Sternberg, and X. Zhuang, "Image analysis using mathematical morphology", *IEEE Trans. on Patt. Anal. Mach. Intell.*, vol. 9, pp. 532–550, 1987.

[5] M. J. Kimmel, R. S. Jaffe, and J. R. Mandeville, et al., "MITE: Morphic image transform engine. an architecture for reconfigurable pipelines of neighbourhood processors", *Proc. IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, pp. 493–500, 1985.

[6] K. G. Konstantopoulos, A. E. Svolos, D. N. Serpanos, and D. G. Maritsas, "An efficient associative processor for two primitive numerical operations employed in computer simulation applications", *Proc. SCS Advanced Simulation Technologies Int. Conf.: High Performance Computing*, pp. 210–215, 1998.

[7] T. Leighton, *Introduction to parallel algorithms and architectures: Arrays-Trees-Hypercubes*, Morgan Kaufmann Publishers Inc., 1992.

[8] G. Matheron, *Random Sets and Integral Geometry*, Wiley, New York, 1975.

[9] D. L. McCubbrey and R. M. Lougheed, "Morphological image analysis using a raster pipeline processor", *Proc. IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, pp. 444–452, 1985.

[10] Y. Saad and M. H. Schultz, *Topological properties of hypercubes*, Technical Report 389, Dept. of Computer Science, Yale University.

[11] I. D. Scherson and S. Ilgen, "A reconfigurable fully parallel associative processor", *J. Parall. Distrib. Comp.*, vol. 6, pp. 69–89, 1989.

[12] I. D. Scherson, D. A. Kramer, and B. D. Alleyne, "Bit-parallel arithmetic in a massively parallel associative processor", *IEEE Trans. on Computers*, vol. 41, pp. 1201–1209, 1992.

[13] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, New York, 1982.

[14] S. R. Sternberg, "Parallel architectures for image processing", *Proc. IEEE COMPSAC*, pp. 712–717, 1979.

[15] T. Bräunl, S. Feyrer, and W. Rapf, et. al., *Parallele Bildverarbeitung*, Addison-Wesley, 1995.

[16] J. Xu, "Decomposition of convex polygonal morphological structuring elements into neighbourhood subsets", *IEEE Trans. on Patt. Anal. Mach. Intell.*, vol. 13, pp. 153–162, 1991.

[17] G. York, R. Managuli, and Y. Kim, "Fast binary and grayscale mathematical morphology on VLIW", *Proc. Of SPIE: Real-Time Imaging IV*, pp. 45–55, 1999.