IEEE UPINLBS'2016
Nov. 3~4, 2016, Shanghai, P.R. China

A Flexible Processing Circuit of Morphological Transform for Obstacle Detection

Rongdi Sun*, Peilin Liu, Zhenqi Wei
School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, China
*Email: the.sun@sjtu.edu.cn

Abstract—**Simply but effectively, morphological transform can be used for obstacle detection in visual navigation. Since the resolution of captured images increases, this paper presents a high-speed circuit for grayscale morphological transform to satisfy real-time processing requirement. The design adopts pipelined architecture with by-pass lines to exploit the most flexibility and scalability. It supports flat structure elements of any shape and size by easy configuration. The overall architecture is implemented based on a Xilinx Virtex-4 FPGA chip with low resource overhead and a high synthesized frequency of 290MHz. It achieves a good performance over 100fps for processing 1080P images.**

Keywords—Morphology; pipelined architecture; grayscale image; arbitrary structure element

I. INTRODUCTION

Morphological transform is a shape-based image processing method which is developed from set theory. It combines various set operations to process a binary or grayscale image [1]. Lots of applications in machine vision adopt morphological transform to extract local geometrical information. Specifically, morphological transform can be used for obstacle detection in visual navigation due to its low complexity in computation. For a 2D grayscale image, morphological transform performs a sequence of neighborhood operations in a siding window named structure element. Structure elements denote the size of neighborhood and involved pixels. Two fundamental morphological operators, dilation and erosion, can be defined as max and min filters in brightness space. As the size of structure elements grows, this computing process becomes more time consuming. Moreover, a structure element may not have a regular shape like square. However, the raw image data are captured and stored pixel by pixel. The asymmetry of structure elements increases the complexity of computing units to read image data in particular sequences.

Many optimized architectures for morphological transform have been proposed from different perspectives including algorithm and hardware. One method is to explore the decomposition property of morphological operations. The core concept of decomposition algorithms is to combine several special structure elements to form the general one. Xu [2] proved that any 8-convex polygon can be built by dilations with the thirteen structuring elements restricted in 4-neighborhood. Normand [3] proposed a method to decompose arbitrary discrete H-convex polygons into unions and dilations of the four two-pixel structuring elements (2PSE) and applied it to construct structure elements for binary morphology. Furthermore Déforges [4] used 2PSE to develop a fast recursive algorithm for greyscale image morphological transform and designed a specific pipeline architecture.

Another method is so called partial-result-reuse (PRR). Wang and He [5] presented an algorithm for a fast 1-D grayscale morphological filters with set structuring elements. It records the extreme value of current window and performs an opening or closing with a single pass procedure. Lam [6] designed an algorithm for 2-D flat structure elements which employs the result of previous searching area determined by a domain-selection method. These algorithms can reduce the computational complexity of morphological operations. Some hardware implementations also learn from the concept of PRR. Ong [7] designed an architecture with a feedback loop path and a decoder/encoder pair comparator. The feedback loop path can reuse partial results to reduce the number of add/subtract units. The decoder/encoder pair comparator using a modified decoding function can reduce the gate count and propagation delay when the size of morphological operations increases. Coltuc and Pitas [8] proposed an optimized PRR architecture by exploring the intrinsic algebraic properties of the max/min operators. And in [9], the authors extended PRR architecture to support various structure elements with the help of systolic arrays.

Both decomposition and PRR methods can improve the computational efficiency but still remain some issues. For example, decomposition often limits the shape of structure elements in convex polygons. Besides, a structure element may have different decomposed results although a unique principle is followed. The PRR method usually uses a small structure element to be duplicated several times to generate a large one, which can also be further duplicated to generate other larger one [9]. This property is easy to be implemented in hardware only for structure elements with self-affinity. By employing systolic arrays PRR-based architectures can deal with more shapes of structure elements. However, it is difficult to configure a reasonable data path, which results in underutilization of large amounts of hardware resources. To address these problems, in this paper we propose a pipelined architecture of morphological transform for grayscale image. The data path can be easily configured just by a binary string to perform any morphological transform with arbitrary shape of structure elements. The hardware implementation on FPGA is able to operate at 290MHz and exhibits high performance in real-time processing for 1080P images.

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

(a) Square

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

(b) Disk

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

(c) Diamond

Fig. 1. Structure elements with different shapes.

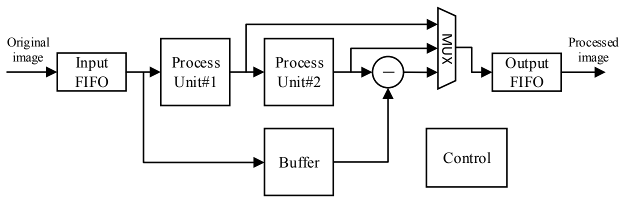

Fig. 2. Architecture of the morphological transform processing circuit.

The rest of this paper is organized as follows. Section II reviews morphology operations and the image processing flow. Section III presents the circuit architecture and elaborates how this design supports user defined structure elements. In Section IV, the architecture is verified by FPGA implementations with various configurations. At last, Section V concludes the entire work in this paper.

II. Primer on Morphology

Mathematical morphology are initially operations on a set of points, which involves two objects: image and structure element. The structure element is usually defined as a small-size binary window sliding over the entire image pixel-by-pixel. In a structure element, the distribution of element "1" determines which pixels in the image will be used for each morphological transform (Fig. 1). Morphological transform includes different neighborhood operations, and it means that the result of each pixel depends on itself and the surrounding pixels. Among all the neighborhood operations two basic operators are dilation and erosion. Suppose I is the point set of a grayscale image and S is the flat structuring element for simplification. The dilation of I by S can be expressed through equation (1):

$$I \oplus S(x, y) = \max\{I(x\text{-}i, y\text{-}j) | (i,j) \in D(S), (x\text{-}i, y\text{-}j) \in D(I)\} \quad (1)$$

where D means the domain of definition. Correspondingly, the flat erosion is defined as (2):

$$I \ominus S(x, y) = \min\{I(x+i, y+j) | (i,j) \in D(S), (x\text{-}i, y\text{-}j) \in D(I)\} \quad (2)$$

And the combination of these two operators will generate closing (3) and opening (4):

$$I \cdot S = I \oplus S \ominus S \quad (3)$$

$$I \circ S = I \ominus S \oplus S \quad (4)$$

Deduced from the equations above, the closing operation will fill the holes or cracks in the region and the opening operation can smooth a rough sketch.

In some specific scenarios, for example, to extract small targets in a background with noise, top-hat transform is applied by subtracting the result of opening operation from the original image. Respectively, bottom-hat transform is to subtract the result of closing operation by the original image. Motivated by the fact that morphology transform is based on these two fundamental operators (dilation and erosion), the proposed hardware architecture implement different morphology operations as many as possible with limited resource overhead.

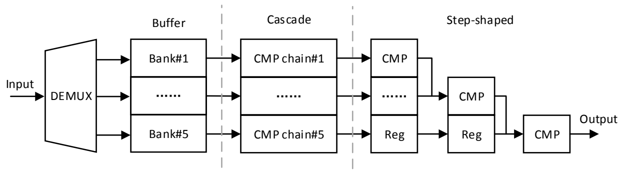

Fig. 3. Architecture of process unit.

III. Configurable Architecture

The data flow of morphological transform indicates that it is suitable for stream processing. Thus we develop a pipeline architecture, which is easy to be set into various operation mode. As showed in Fig. 2, the overall architecture consists of input and output FIFOs, two process units, a buffer, a subtracter, a multiplexer and a control unit. The control unit is initialized by setting image size, structure element, function of the process units and so on. Both of the process units can perform dilation or erosion operations. The buffer organized in multi-banks serves to store raw image data if necessary. Only when performing some compound operations, with data from process unit#1 and the buffer, is the subtracter used. The multiplexer selects a data source from process unit#1, process unit#2 or the subtracter. Different selections are mapped to different morphological operations. Certainly process unit#2, the buffer and the subtracter may be disenabled based on operation types to save power. To take top-hat transform as example, we can divide the computation into three steps. First, the original image data from the input FIFO are sent to process unit#1 as well as the buffer, and process unit#1 computes the result of erosion. Second, process unit#2 performs dilation operation with data from process unit#1 and generates the result of opening. Third, the subtracter outputs the final result, which is the equal of original image minus opening image, and writes it to the output FIFO.

A. Basic Processing Unit

Commonly raster scan images are transmitted line by line continuously, while morphology operations are basically comparison within a 2D window (square structure elements) passing over the entire image. Naturally we can decompose the window into separate rows, process each row in parallel and compare the result of each row to figure out the final results (maximum or minimum in the window). As show in Fig. 3, the input data are first allocated in a round robin way to different banks of the inside buffer. Each bank stores a row of image and the number of banks equals to the height of structure elements. Following is the compare module including several cascade comparator chains and a step-shaped comparator chain. To

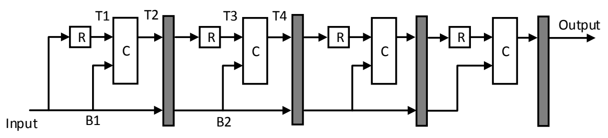

Fig. 4. Architecture of a casade comparator chain

reduce the start-up delay, image data are transmitted to the compare module once enough data are available to compute the transform of the first pixel. For example, if a 5x5 structure element window is employed, bank#1~bank#3 are read simultaneously when the first pixel of the third row has been written into bank#3. When processing pixels on image borders, extreme value (0xFF or 0x00 for 8-bit) will be filled into the blanks as the virtual pixels outside the image to cover the region of the structure element window.

A cascade comparator chain in Fig.4 includes a number of comparators that run in series (C). A pipeline register (dark rectangles in Fig.4) is inserted between two comparators in adjacent stages to cut off the long combinational logic path and improve circuit timing. The input data flow through two separate links corresponding to two input ports of comparators. The top input port of each comparator is delayed by a register (R). Thus for each comparator one source is a partial result while the other is an unused pixel. Suppose all comparators are set to max mode and a row of image data are input to the comparator chain pixel-by-pixel. In the second clock cycle, the node B1 will have the second pixel (P2) and the node T1 will have the first pixel (P1) due to the delay-register. So T2 will produce the maximum of P1 and P2, denoted by M(1,2). Both P2 and M(1,2) will be written into the following pipeline register in the third cycle. Thus the node B2 will have P2 and B1 will have P3. Then in the fourth cycle M(1,2) will pass through a register to T3, and B2 will receive the value of P3 from B1. Thus T4 will generate M(M(1,2),P3)=M(1,2,3). The similar operations will be performed on the next stage and so on. Finally, M(1,2,3,4,5) will come to the output, and it will be updated to M(2,3,4,5,6) in the next clock cycle. The same comparator chains are placed in parallel to process 5 rows of image data synchronously. Afterwards the output data of each row are sent to a step-shaped comparator chain to calculate the final result.

B. Modified Comparator Chain

Except square structuring elements, other shapes like disk and diamond are also useful for morphological transform. Sometimes structure elements without symmetry may be applied for special purpose. For these situations, we have to modify the basic logic units to support user-defined structure elements. We set two constraints when introducing extra logics.

- Keep the pipeline timing fixed. For irregular structure elements the pipeline takes the same number of clock cycles to complete the task as the square structure elements.
- Avoid complex analysis of structure elements shape. Programmers have no need to redesign the algorithm flow and they just input structure elements and configure comparator mode.

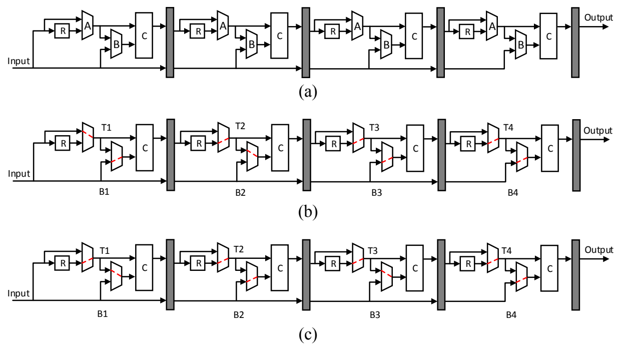

Fig. 5. Configurable comparator chain (a) and the data path for structure elements "01011" (b) and "10101"(c)

Following the principles above, we add a multiplex (MUX) to each input port of the comparators in comparator chains. In Fig. 5(a) two types of MUX are labelled by A and B respectively. MUX A is used to control the data transmission delay while MUX B is to determine data sources of two input ports of a comparator. If MUX B selects the top input data, the two inputs will be the same thus we can regard the comparator as a direct connection. By configuring MUX A and MUX B, the comparator chain can operate with arbitrary structure elements.

To elaborate the details of computing flow, we take a 1D structure element "01011" for example. Fig. 5(b) shows the data path for this structure element. When the first pixel P1 arrives, both the node T1 and B1 keep the same value P1 so the first comparator will output P1. After P1 is written into the pipeline register, it takes one cycle for P1 to arrive at T2, and now B2 has the value of P2. The second comparator will also output P1 regardless the value of B2, because it selects the same input data. Note that B3 has the value of P3 when P1 comes to T3, thus the third comparator will output M(1,3). T4 and B4 will hold the value of M(1,3) and P4 respectively in two cycles, and the fourth comparator will output M(M(1,3),P4). Finally, the output port will generate the correct result M(M(2,4),P5)=M(2,4,5) in the tenth cycle.

TABLE I. DATA OF EACH NODE AT EACH CYCLE

Node / Cycle	In	T1/B1	T2/B2	T3/B3	T4/B4	Out
1	1	1/1	-	-	-	-
		-/1				
2	2	2/2	-/1	-	-	-
		1/2				
3	3	3/3	(1,1)/2	-/1	-	-
		2/3	-/2			
4	4	4/4	(2,2)/3	-/2	-/1	-
		3/4	(1,1)/3			
5	5	5/5	(3,3)/4	(1,1)/3	-/2	-
		4/5	(2,2)/4	-/3		
6	6	6/6	(4,4)/5	(2,2)/4	-/3	-
		5/6	(3,3)/5	(1,3)/4		
7	7	7/7	(5,5)/6	(3,3)/5	(1,3)/4	-
		6/7	(4,4)/6	(2,4)5	-/4	
8	8	8/8	(6,6)/7	(4,4)/6	(2,4)/5	-
		7/8	(5,5)/7	(3,5)/6	(1,3)/5	
9	9	9/9	(7,7)/8	(5,5)/7	(3,5)/6	(1,3,4)
		8/9	(6,6)/8	(4,6)/7	(2,4)/6	-
10	10	10/10	(8,8)/9	(6,6)/8	(4,6)/7	(2,4,5)
		9/10	(7,7)/9	(5,7)/8	(3,5)/7	(1,3,5)

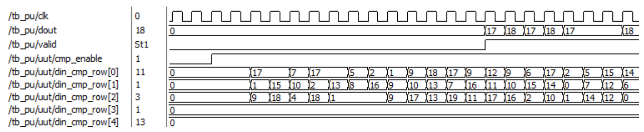

Fig. 6. Simulation of dilation with a 5x5 diamond structure element.

The values of each node at different cycles in a comparator chain are listed in Table I. The top half of each table cell is for structure elements "01011" and the bottom half is for "10101". From the table we can see that the valid result will always be output in a fixed number of clock cycles. It indicates that the modified architecture keeps pipeline depth the same.

Now a critical issue has come to us that how to configure the MUX A and MUX B by the only known structure elements. By observing the data flow in the pipeline step by step, we conclude several general rules.

- Count the number of "0" from the MSB in a structure element string with the length of N until met with the first "1", denote this number by M.
- In the input-to-output direction, configure the first M MUX A to select the straight-forward path, and the rest to select the register-delay path. If M=N, ignore the last one.
- Use the lower N-1 bit of the structure elements string to configure MUX B. "1" stands for selecting the upper path while "0" selecting the bottom, i.e. "1" means the comparator performs compare operation and "0" means it directly outputs the input data.

The proposed method has been verified in both Fig. 5 and Table 1. In this way users only need to write the structure elements strings by row and the hardware will undertake configuration.

C. Parallel Processing

As mentioned above, several pipelines are built to process the 2D morphology transform in parallel. On the condition of arbitrary structure elements, the configuration strings are not fixed for each comparator chain. Taking the structure elements window showed in Fig. 1(c) into consideration, we use a long 5-syllable binary string "00100_01110_11111_01110_00100" to represent it. Each syllable of the string corresponding to the configuration of a comparator chain. Assume the buffers store image data from the 1st row to the 5th row and the compare module is computing the transform of the 3rd row. Since the input data are stored in the buffers in a round robin way, next time the 6th row will replace the 1st row. Due to one-to-one connections between the buffers and the comparator chains, the configuration strings of each chain have also to be rotated. We use a row counter to make the configuration string perform circular shift by syllable-step. Thus the binary string will be "00100_00100_01110_11111_01110" when processing the 4th row.

If a syllable in the string is "00000", which means the current computation does not involve the data of this row. At this time the step-shaped comparator chain has to ignore the input of this row. Similarly, we also attach a MUX to the output end of each previous comparator chain, and generate a "use" signal by computing OR of each bit in a syllable to control the MUX. When "use" is invalid the output of this comparator chain will be replaced with an extreme value. Thus, the connected comparator in the step-shaped chain will always select data from another input port as if the data marked with a string "00000" is always skipped.

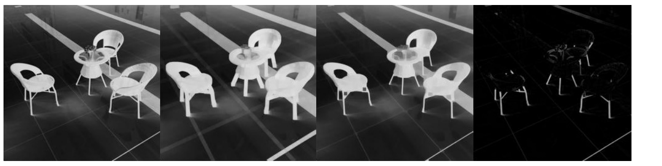
(a) image (b) dilation (c) opening (d) tophat

Fig. 7. Morphological transform with a 7x7 diamond structure element.

D. Handling with Image Borders

Image borders can be grouped into two categories. When it comes to the top or bottom of an image, only a few RAMs have valid image data, while the rest should be filled with extreme values as row-wise extension of the image. On the other hand, the start or end of a row also should be extended with extreme values to (N-1)/2 pixels, where N stands for the width of structure elements. Instead of writing extra pixels into RAMs, we directly replace the data at the input ports of comparator chains when processing pixels on the edge. In the meantime, the input FIFO and buffers at previous stages stopp reading and writing for N-1 cycles. Fig. 6 shows the simulation result of the proposed architecture for dilation by a 5x5 diamond structure element (Fig.1 c). We can see two zeroes are sent to each the comparator chain.

IV. Evaluation of Recource Cost and Performance

The hardware overhead mainly depends on the size of images and structure elements related with applications, so we will roughly deduce the design parameters. From Section III we know that process unit #1 starts running (reading from the RAM) when the first pixel of the (N+1)/2th row has been buffered in the RAM, where N is the width and height of structure elements. During the period of an entire image, process unit #1 has to stop reading the input FIFO for H-(N+1)/2 times, N-1 cycles at a time. This feature leads to data accumulation of the input FIFO because the raw data are written to the input FIFO uninterruptedly for an image. Moreover, the number of banks in a RAM equals to the height of structure elements and each bank stores a row of image data, either for the two process units or the buffer. Thus, the total on-chip memory size can be estimated by (5):

$$(H-(N+1)/2)(N-1)+3WN \tag{5}$$

where W and H are the width and height of images. The first term represents the input FIFO depth and the second represents storage capacity of RAMs in the two process units and the buffer. Note that the output FIFO is not counted in (5) because it relies on reading signals from off-chip controller. Intuitively we keep the output FIFO depth the same as the input. Besides,

TABLE II. COMPARISON BETWEEN PROPOSED ARCHITECUTRE AND OTHER WORKS

Architecture	Chein	Déforges	This work
Arbitrary structure element	Partially	Partially	Yes
Comparator counts[a]	6	12	48
Delay cycles	6W+6	6W+6	3W+20
Configuration complexity	Medium	High	Low

[a] in one process unit

the number of function units can be calculated according to the size of structure elements by (6):

$$2N(N-1)+2(N-1)+1 \quad (6)$$

in which the first two terms means the number of comparators in the two process units, and the last constant "1" corresponds to the subtracter.

In practice we implement this work in a Xilinx Virtex-4 FPGA supporting up to 1280x1080 images, 16bit data and 7x7 structure elements. It consumes 58 BRAMs for FIFOs and multi-bank RAMs, and 5962 LUTs for logic units. Since a BRAM capacity is fixed to 18Kbit, we have to use two blocks for each memory bank (30.72Kbit). The synthesized frequency has been improved to 290MHz according to the report. The delay between data input and the result output in one process unit is given by (7):

$$W(N-1)/2+2N+(N-1) \quad (7)$$

where the first term is data preparing time before computing, the last two terms are the delays in cascade and step-shaped comparator chains respectively. Even at the frequency of 100MHz, the architecture still meets real-time requirements to process 1080P images. Fig. 7 shows tophat transform of a 256x256 image with the 7x7 diamond structure element.

Table II compares two recent works by Déforges [4] and Chein [9] with the proposed architecture on the condition of dilation by 7x7 structure element. Chein's design has the least comparator and it does not change along with the size of structure elements. However, it would support only self-affine structure elements if no systolic arrays were used. Déforges's work supports convex polygon structure elements by decomposition method but it is complex in configuration. The proposed architecture costs the most comparators but operates with the least startup delays. It stores the raw data inside process units for a while and then begins to compute the results continuously. By simple configuration it can perform different morphological transform by arbitrary flat structure elements.

V. CONCLUSION

Obstacle detection is one of the most significant steps in visual navigation. Although many powerful algorithms have been developed for this task, morphological transform are still used when accurate recognition is not required due to its low complexity. In this paper, we propose a flexible architecture of morphological transform for processing high-resolution images in real time. It supports arbitrary structure elements by simple configuration. An approach to configure the data path with binary strings is demonstrated by analysis and simulation. The processed data are generated per clock cycle without a break after a short delay because of the fully pipelined architecture. This work is verified in a Xilinx Virtex-4 FPGA and the report shows a synthesized frequency of 290MHz. It can perform morphological transform on 1080P images at 100fps.

ACKNOWLEDGMENT

This research work is supported by the Important National Science and Technology Specific Project of China under Grant No. 2016ZX03001022-006, the National Natural Science Foundation of China under Grant Nos. 91438113, 61402283, 61573242, 61401501 and 61304225, the National Science and Technology Major Project under Grant No. GFZX0301010708, and the Shanghai Science and Technology Committee under Grant Nos. 16DZ1100402 and 15511105100.

REFERENCES

[1] R.M. Haralick, S.R. Sternberg, and X. Zhuang, "Image analysis using mathematical morphology," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 9, no. 4, pp. 532-550, Jul. 1985.

[2] J. Xu, "Decomposition of convex polygonal morphological structuring elements into neighborhood subsets," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 13, no. 2, pp. 153-162, Feb. 1991.

[3] N. Normand, "Convex structuring element decomposition for single scan binary mathematical morphology" in International Conference on Discrete Geometry for Computer Imagery, 2003, pp. 154-163.

[4] O. Déforges, N. Normand, M. Babel, "Fast recursive grayscale morphology operators: from the algorithm to the pipeline architecture," Journal of Real-Time Image Processing, vol. 8, no. 2, pp. 143-152, Jun. 2003.

[5] D. Wang, D.C. He, "A fast implementation of 1-D grayscale morphological filters," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 41, no. 9, pp. 643-636, Sep. 1994.

[6] R.W.K. Lam,C.K. Li, "A fast algorithm for morphological operations with flat structuring element," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 45, no. 3, pp. 387-391, Mar. 1998.

[7] S. Ong, M.H. Sunwoo. "A new cost-effective morphological filter chip," in IEEE Workshop on Signal Processing Systems, 1997, pp. 421-430.

[8] D. Coltuc, I. Pitas. "Fast computation of a class of running filters," IEEE Transactions on Signal Processing, vol. 46, no. 3, pp. 549-553, Mar. 1998.

[9] S.Y. Chien, S.Y. Ma, L.G. Chen, "Partial-result-reuse architecture and its design technique for morphological operations with flat structuring elements," IEEE Transactions on Circuits and Systems for Video Technology, vol. 15, no. 9, pp. 1159-1169, Sep. 2005.