A 2D Shape Boundary Detection Algorithm for VLSI Implementation

Elmar Melcher, João Marques de Carvalho, Marcelo de Barros,
Lírida Naviner, Jean François Naviner,
Valteir R. da Silva[1],
Fred Henrique Souza Paes[1],
Ricardo A. S. Moreira[1],
Jeanne E. de P. Braquehais

Universidade Federal da Paraíba
Departamento de Engenharia Elétrica
58.109-790 Campina Grande - PB, Brasil
elmar@dee.ufpb.br

Abstract. In this work a parallel architecture is proposed for VLSI implementation of a data-flow algorithm for 2D boundary (or contour) detection. The algorithm works on the gradient image and uses a set of primitive paths to generate all possible contour paths on a neighborhood defined by a 3×3 window. The objective is to determine whether or not the neighborhood central pixel belongs to a continuous boundary line passing across the window. Test results show that one-pixel wide continuous boundary lines can be extracted using this algorithm.

1 Introduction

Polygonal modeling has traditionally been one of the most popular methods for shape representation and still is largely used in applications where shape recognition of two-dimensional objects or surfaces is needed [SR92, SM92].

Polygonal models have the advantage of being a local representation, i.e., they preserve local shape features therefore allowing for object recognition, even in the presence of partial occlusion. Additionally, they can be made insensitive to rotation, translation, and scaling, (a requirement for any practical recognition system) and are much less computationally expensive than higher-order polynomial approximations. As a drawback, the representation provided by polygonal models is usually not as compact as those based on global features such as moments or transform descriptors. A more complete analysis of the issues involved in those and other forms of shape representation has been made by Pavlidis [Pav78]. Further details about the advantages of polygonal modeling can also be found in the literature [AF86, KD82].

In order to operate properly, algorithms for polygonal modeling of 2D objects require shapes with continuous and well defined boundaries. Generation of this boundary is the objective of the preprocessing phase to which the original image of the object to be modeled is normally submitted. As part of a typical preprocessing operation initially a discrete gradient operator is employed to generate a gradient image, upon which *boundary tracking segmentation* can be performed [MM92].

Most of the work so far reported on algorithms for boundary extraction on digital images assume a sequential software implementation, either on a general purpose computer or on an specialized signal processor. This type of approach is inadequate if real-time high speed operation is desired, due to the computationally intensive character of low-level image processing. Applications such as vision systems for mobile robots may require 512×512 image frames with 256 gray levels (8 bits) to be processed at a rate of 30 frames per second. Real-time operation at this rate would require a processing time of 120 nanoseconds for each image element or *pixel*. This performance figure becomes clearly out of reach for any sequential general purpose computer when one considers the amount of multiplications, additions and other operations usually involved in each output pixel computation. The obvious solution is to design parallel algorithms and architectures which can be implemented in specialized integrated circuits called *ASIC's* using CAD-based VLSI design tools.

The availability of very powerful and ease to use VLSI design tools has fueled the development of several real-time image processing systems, particularly for low-level feature detection and extraction

[1] bolsistas PIBIC/CNPq/UFPB

applications. Bhanu et al. have designed and implemented a real-time segmentation processor which makes use of a gradient relaxation algorithm (iterative) to assign pixels into classes, based on their gray value and the gray values of neighboring pixels [BHS90]. Ranganathan et al. have proposed a VLSI architecture which convolves images with eight 15×15 kernels in order to implement a technique for corner detection which is based on the concept of *half-edge* and on the first derivative of Gaussian [RNM91]. Cheng et al. utilized the theory of dynamic programming to develop a backtracking method for curve detection and designed an associated VLSI architecture which solves the problem in $O(n)$ time, where n is the length of the curve to be found [CTL90]. All these architectures make use of pipelining and parallelism in order to achieve real-time performance. More recently, Melcher at al. have used the CAD toll called ALMA, a finite state machine generator, to design a VLSI architecture for polygonal modeling of two-dimensional shapes. This circuit uses parallelism to simultaneously perform segmentation and vertex extraction, therefore maximizing speed [MCa94]. A good critical survey of parallel architectures and algorithms for image processing has been done by Cypher and Sanz [CS89].

In this work a parallel architecture is proposed for VLSI implementation of a data-flow algorithm for 2D boundary (or contour) detection. The algorithm works on the gradient image and uses a set of primitive paths to generate all possible contour paths on a neighborhood defined by a 3×3 window. The objective is to determine whether or not the neighborhood central pixel belongs to a continuous boundary line passing across the window.

The rest of the work is organized as follows: Section 2 describes the algorithm for boundary detection. Sections 3 and 4 present VLSI hardware implementations of the proposed algorithm. Section 5 shows results obtained by simulating actual circuit operation. Finally, conclusions are drawn in section 6.

2 Algorithm

For hardware implementation of image processing, the most performant algorithms are those of the data-flow type. The image pixels come in serially, pixel by pixel, one line after the other. For every incoming image pixel, a data-flow algorithm produces one output pixel. The output pixel is determined by a function of the corresponding input pixel and neighboring pixels. These pixels form a window. Apart from the function, the performance and the data storage requirements of data-flow algorithms depend on the size of the window, as shown in table 1. The storage size and delay times are for images with 8 bits per pixel, 512 pixels per line, 512 lines per image, 30 images per second.

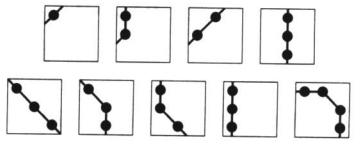

Figure 2: Primitive paths for pathfinder I

The performance characteristics of data-flow algorithms makes them the best choice for real time image processing. The boundary detection algorithm proposed in this paper consists of three processing stages, as shown in figure 1. Each stage will be described in detail in the following paragraphs.

2.1 Robert's Gradient

Robert's Gradient is based on a 2×2 window and approximates the image gradient at pixel coordinates (m, n) by:

$$P_g(m,n) = |P_o(m-1,n-1) - P_o(m,n)| + |P_o(m-1,n) - P_o(m,n-1)| \quad (1)$$

where P_o is a pixel gray level value of the original image, and P_g is a pixel gray level value of the gradient image.

2.2 Local Maximum and Pathfinder I

This phase works with two criteria. If both criteria are validated, the center pixel of the window becomes the output pixel, otherwise the output pixel becomes zero. Note that the output image at this phase is not a binary image. It has as many gray levels as the input image.

The local maximum criterion is based on a 5×5 window. It is validated if the center pixel value is greater than 10% of the full scale value and if it is among the five biggest values in the window.

The pathfinder criterion works on a 3×3 window. This criterion determines whether the center pixel of the window belongs to a continuous border line passing across the window. To accomplish this, all possible border paths across a 3×3 window must be checked. The possible paths are derived from the primitives in figure 2 by mirror and rotation operations. Eliminating repetitive patterns, a total of 44 distinct paths was obtained. Note that the paths in figure 2 do not contain any sharp corners, which would difficult subsequent border tracking.

window pixel x pixel	storage space required	image delay
2×2	1 line + 2 pixel = 4112 bits	1 pixel = $0.1\mu s$
3×3	2 lines + 3 pixels = 8216 bits	1 line + 2 pixels = $61.7\mu s$
5×5	4 lines + 5 pixels = 16424 bits	2 lines + 4 pixels = $123.3\mu s$

Table 1: Window size and performance of data-flow algorithms

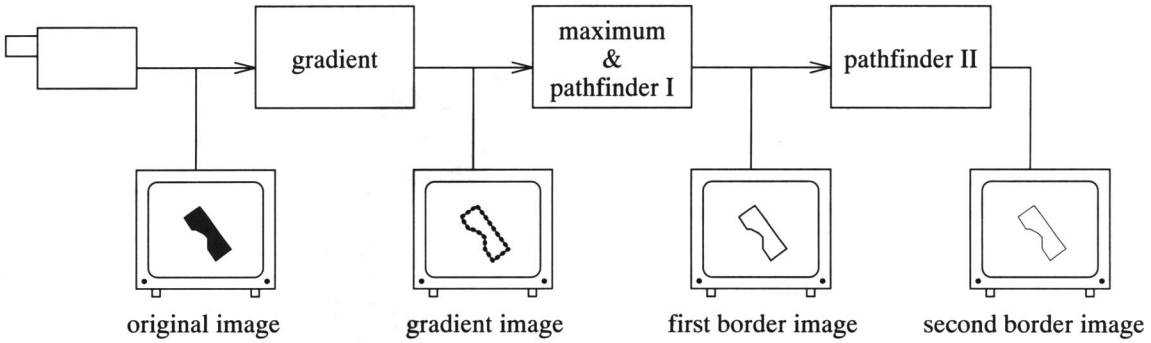

Figure 1: The image processing stages of the proposed algorithm

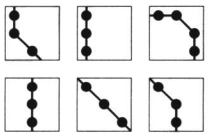

Figure 3: Primitive paths for pathfinder II

The pathfinder criterion is validated if the center pixel belongs to one of the six paths with the largest value. The value of a path is determined by the weighted sum of its pixel values:

$$S_p = \frac{12}{N_p} \sum_{i=1}^{N_p} P_g(m_i, n_i) \qquad (2)$$

where N_p is the number of pixels of the path and m_i and n_i are the image coordinates of the path in the window. Note that S_p is always an integer value.

2.3 Pathfinder II

The pathfinder II algorithm works only with paths containing at least 3 pixels. These 28 paths were derived from the primitives in figure 3.

The same equation (2) is used to calculate the weighted sum for each path. If the center pixel of the window belongs to the path with the highest sum, the output pixel is set to 1.

If the two paths with the largest weighted sums have equal value and the center pixel of the window belongs to one of those paths, an arbitrary choice has to be made: if in this situation the sum of the three pixels in the upper right corner of the window is bigger than the sum of the three pixels in the lower left corner, the output pixel is set to 1. In all other cases the output pixel is set to 0. The criterion assures that only one of the pixels of the different paths is set to 1, providing a one pixel wide border line. In the examples shown in section 5, this situation occurs at about 4% of the border pixels.

3 ASIC Hardware Implementation

The implementation of a data-flow algorithm in an ASIC is relatively easy compared to the implementation of other algorithms. Virtually no control circuitry is needed because the sequential part of the circuit deals only with the uniform pixel flow. In many cases the design can be done automatically by using high level synthesis tools [dMCa90].

For these reasons and in order to be concise, this section only estimates the complexity of the hardware, i.e. the chip area required. The estimation is based on the ES2's RAM generator [Eur90] in ECPD12 technology and an average density of 10.000 transistors per mm^2. The estimation results are summarized in table 2.

The gradient operator needs two subtractors and one adder/subtractor, i.e. only 675 transistors.

The local maximum operator has to compare 24

operator	storage		computation	
	bits	area mm^2	transistors	area mm^2
gradient	4112	2.40	675	0.07
maximum	16424	9.63	2082	0.21
pathfinder I	.	.	30660	3.07
pathfinder II	8216	4.81	20570	2.06

Table 2: Complexity of the hardware implementation

	time in ns	area in CLBs	use in %
Add/Sub	15	5	2,5
Comparator	15	5	2,5
Multiplexor	6	4	2

Table 3: Complexity of FPGA Implementation: delay time, number of Configurable Logic Blocs needed, percentage of FPGA used.

peripheral pixel values to the center pixel value. The result bits "greater than or equal" of the comparisons have to be added and compared to 5. That is done by 24 comparators, an adder tree of 24 operands of 1 bit each, and one additional comparator. This sums up to 2082 transistors.

The pathfinder I operator uses the same storage memory as the maximum operator. The computation here is done by 136 adders for the weighted sums, one comparison tree that determines the maximal weighted sum among the 12 paths that include the center pixel, 32 comparators that compare this value to the weighted sums of the remaining paths, an adder tree for 32 operands of 1 bit each, and one additional comparator.

The pathfinder II operator needs 92 adders for the 28 weighted sums, one comparison tree that determines the maximal weighted sum among the 12 paths that include the center pixel, 16 comparators that compare this value to the weighted sums of the remaining paths, and one logic NOR with 16 inputs.

The total chip area is thus estimated to $22mm^2$. Note that the chip area is almost directly proportional to the number of pixels per line and to the number of bits per pixel.

Figure 4: Original image of toy block (a), gradient image (b), first border (c) and second border (d)

4 Low-Cost FPGA Implementation for Hardware Validation

In order to perform an evaluation of the hardware implementation proposed, a low-cost prototyping platform is under development. It consists of a highly programmable prototyping board with an adapted (high throughput) interface with the host machine of the image processing system. This board holds two XC4013 Xilinx FPGA (Field Programmable Gate Array) circuits [Xil94].

Such an approach allows a system level validation of the designed architecture step by step. Each step is performed by customizing the prototyping board with the specific architectures of gradient, maximum, pathfinder I and pathfinder II modules. This is made by programming sequentially the FPGA circuits. After each step, the intermediate processing results are stored back in the host machine and then used as data entry for the following step.

To allow validation of real time image processing operators, the FPGA implementation of the algorithms needs special adaptation at algorithm, architecture and technology levels [dB94]. So, the elementary operators of boundary detection algorithm (comparators, multiplexers, adders and subtractors) were implemented by a "full-custom-like" design approach and present the characteristics in table 3 when using XC4013 SRAM FPGA circuits from Xilinx.

5 Simulation Results

Operation of the border detection hardware implementation was simulated by a program written

in C language. The choice of C is justified by the fact that it allows fast simulations at functional level, providing fast turn-around time for debugging and parameter adjustment. A hardware description language like VHDL runs slower and is more complicated to debug. The use of C language made it possible to involve a team of undergraduate students in the project, who where actually making their first real-world experience with both C language and VLSI implementation.

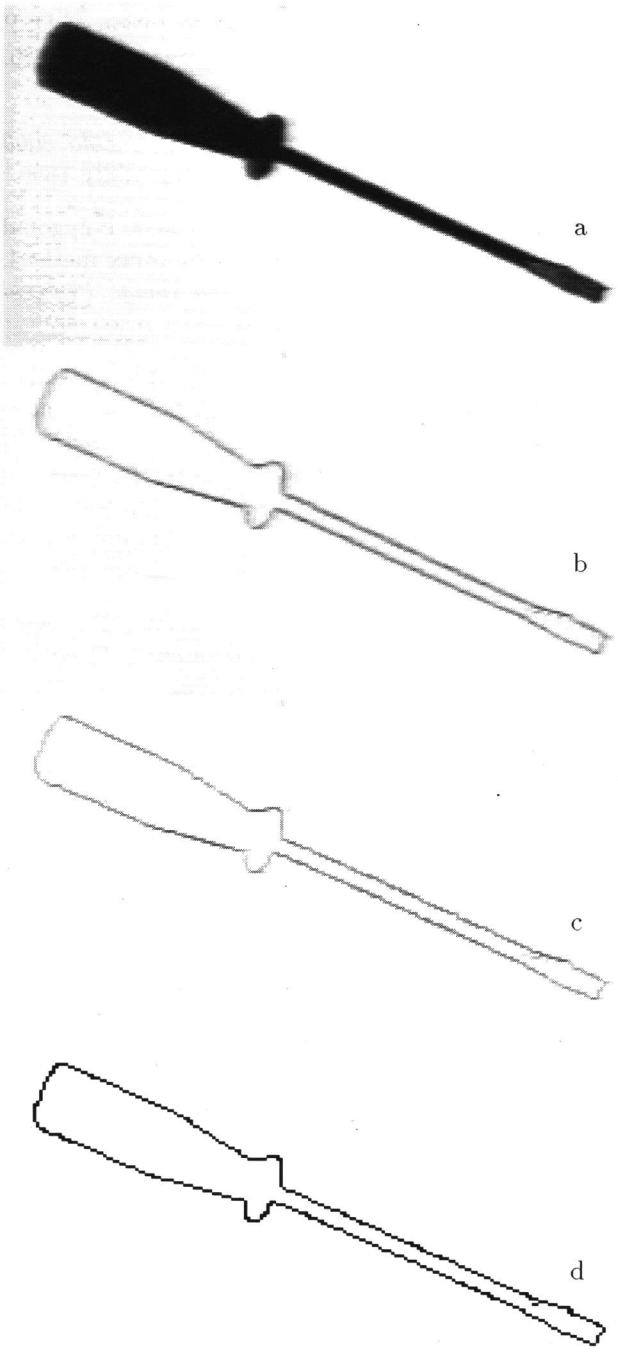

Figure 5: Original image of the screwdriver (a), gradient image (b), first border (c) and third border (d)

Images of two objects were used to validate the algorithm: An arch-shaped toy block, figure 4 and a screwdriver, figure 5.

The toy block is made of 64×64 pixels. 16 grey levels (4 bits) are used.

Figure 5a shows a shape with sharp contrast. The gradient's image 5b reaches strong grey level values all around the border and the contour line is relatively thin. This makes the task of border extraction rather easy, resulting in a perfectly continuous line no more than one pixel wide in figure 5d.

The screwdriver is made of 220×128 pixels. 32 grey levels (5 bits) are used.

In comparison to the toy block, the screwdriver in figure 5a has poor contrast, mainly due to its round shape in perpendicular sections of the image plane and because of the shadow caused by light that falls onto the image plane in angle from the left side. To avoid the shadow, the light source should be located next to the camera. The contrast is particularly bad at the tip of the screwdriver and at the top of the handle.

The gradient image (figure 5b) reaches good (high) grey level values at a few points only and the contour line is wider than in figure 4b. After the second processing stage, employing the maximum operator and pathfinder I, the border is much finer (figure 5c), but still far from perfect.

Two stages with the pathfinder II operator were needed to reach an acceptable result in figure 5d. The first stage resets the pixels that are not considered to belong to the border line but lets the border pixels unchanged. Only the second stage reduces the image to a binary image. The border line is perfectly continuous, but there are a few locations where the border line is more than one pixel wide. There also is a spurious border that appears near the tip of the screwdriver. Comparing with the original image we recognize the contour of the flat part of the srewdriver's tip at that location.

Still, the quality of the contour in figure 5d is clearly sufficient for a subsequent vertex extraction procedure.

6 Conclusion

The simulation results show that the hardware implementation of the proposed algorithm is capable of producing good results even for images with poor contrast. In both cases a continuous, well defined, one pixel wide contour was extracted by the circuit, which will ease considerably the task of the vertex ex-

traction algorithm, last stage of the polygonal modeling operation.

A data-flow solution for the vertex extraction procedure is rather more complicated than a sequential one. Sequential algorithms progress pixel by pixel along the shape contour, testing each new pixel in order to detect vertex for the polygonal approximation. A list of the boundary points has to be available beforehand, which precludes its use on high-speed real-time operation. In the other hand, data-flow algorithms work on the image as it is being acquired line by line. Therefore, they represent the only possible solution if real-time performance is to be achieved. However, data-flow algorithms look at the image through a rectangular window of a given size, 3×3 or 5×5, for instance. This fact makes it difficult to determine to which shape a contour segment belongs, when more than one object (or shapes) are present in the image scene being modeled. A compromise solution to the above problem may be a hybrid architecture, where vertex detection would be followed by a non data-flow procedure in charge of assigning detected vertex to objects.

The development of an algorithm for VLSI implementation of the complete polygonal modeling procedure is presently under way. Once it is completed and integrated with the circuit described in this paper, a complete high-speed system for polygonal modeling of 2D shapes will be available, capable of performing at the rates required for real-time applications.

References

[AF86] N. Ayache and O.D. Faugeras. HYPER: A new approach for the recognition and positioning of two-dimensional objects. *IEEE Transactions on Pattern Analysis and Machine Inteligence*, 8(1):44–54, 1986.

[BHS90] B. Bhanu, B.L. Hutchings, and K.F. Smith. VLSI design and implementation of a real-time image segmentation processor. *Machine Vision and Applications*, 3:21–44, 1990.

[CS89] R. Cypher and J.L.C. Sanz. SIMD architectures and algorithms for image processing and computer vision. *IEEE Transac. Acoustics, Speech and Signal Proc.*, 37:2158–2174, 1989.

[CTL90] H.D. Cheng, C. Tong, and Y.J. Lu. VLSI curve detector. *Pattern Recognition*, 23:35–50, 1990.

[dB94] Marcelo Alves de Barros. "Low Level image Processing Operators on FPGA: Implementation Exemples and Performance Evaluation". *Proc. of the 12th International Conference on Pattern Recognition*, Oct. 1994.

[dMCa90] H. de Man, F. Catthoor, and alii. "Architecture-Driven Synthesis of techniques for VLSI Implementation of DSP Algorithms". *Proc. of the IEEE*, Feb. 1990.

[Eur90] European Silicon Structures. *Solo 2000 Family Libraries*, 3rd edition, July 1990.

[KD82] Y. Kurozumi and W. A. Davis. Polygonal approximation by the minimax method. *Computer Graphics and Image Processing*, 19:248–264, 1982.

[MCa94] Elmar U.K. Melcher, João M. Carvalho, and alii. VLSI circuit for polygonal modelling of 2D shapes designed using the finite state generator ALMA. In *Anais do IX SBMICRO*, Congresso da Sociedade Brasileira de Microeletrônica, pages 359–368, Rio de Janeiro - RJ, Agosto 1994.

[MM92] A.D Marshall and R.R. Martin. *Computer Vision, Models and Inspection*. World Scientific, London, U.K., 1992.

[Pav78] T. Pavlidis. Survey: A review of algorithms for shape analysis. *Computer Graphics and Image Processing*, 7:243–258, 1978.

[RNM91] N. Ranganathan, S.J. Nichani, and R. Mehrotra. A VLSI architecture for a half-edge based corner detector. *Machine Vision and Applications*, 4:165–181, 1991.

[SM92] F. Stein and G. Medioni. Structural indexing: Efficient 2-d object recognition. *IEEE Transactions on Pattern Analysis and Machine Inteligence*, 14(12):1198–1204, 1992.

[SR92] I. K. Sethi and N. Ramesh. Local association based recognition of two-dimensional objects. *Machine Vision and Applications*, 5:265–276, 1992.

[Xil94] Xilinx. *"The XC4000 Programmable Gate Array Data Book"*. San Jose, CA, USA, 1994.