A Vertex Detection Algorithm for VLSI Implementation

E. Melcher[1], J.M. de Carvalho[1], P.C. Cortez[2], L. Naviner[1], J.F. Naviner[1]

[1]Universidade Federal da Paraíba - DEE - CCT
Caixa Postal 10105, 58.109-970 Campina Grande - PB, Brasil
carvalho@dee.ufpb.br
[2] Universidade Federal do Ceará - DEE - CT

Abstract. This paper describes a data-flow vertex detection algorithm for polygonal modeling of 2D contours. VLSI implementation by a parallel architecture is also proposed. This work is part of a real-time shape recognition system using polygonal models, presently under development.

Keywords: Shape Recognition, Polygonal Modeling, VLSI.

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. 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 [1].

Computation of a gradient image is the initial stage of the polygonal modeling operation. Once the gradient image is available, it can be submitted to boundary detection and vertex extraction, which produces the desired polygonal model. The development of data-flow VLSI implementations for the gradient computation and boundary detection operation have been previously reported [2, 3].

The performance characteristics of data-flow algorithms make them the best choice for real time image processing. 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 as a function of the corresponding input pixel and a set of neighboring pixels.

2 Vertex Detection Algorithm and Architecture

In this work a parallel architecture is proposed for VLSI implementation of a vertex extraction data-flow algorithm for polygonal modeling. The algorithm works on a 2D boundary (or contour) and uses a set of primitive paths to generate all possible contour paths crossing a neighborhood defined by a 3×3 window. All the primitive and generated paths possess a vertex at the central pixel of window. The objective is to determine by comparison whether or not the central pixel of a 3×3 neighborhood of the input image is also a vertex.

steps k	window size	storage (bits)	image delay (μs)
1	3×3	1027	$61.7 \mu s$
2	5×5	2053	$123.3 \mu s$
3	7×7	3079	$184.8 \mu s$
4	9×9	4105	$246.1 \mu s$

Table 1: Performance of data-flow algorithms

In order to effectively implement the algorithm proposed in a data-flow scheme all possible contours have to be pre-calculated. The hardware implementation of the logic that detects if an image window corresponds to one of the vertex patterns is then straightforward.

The procedure for vertex detection, used to calculate the set of primitive patterns, has been previously developed by Cortez [4]. It utilizes a 3×3 mask which is shifted k steps forward and backward over the contour being modeled, starting from an initial central point. The method uses the Hough Transform to estimate the changes of curvature that occur at each step, in order to decide whether or not the initial point is a vertex. The greater the number of steps taken the more precise will be the curvature estimation, at the expense of more calculations. A value of $k = 2$ was found to produce satisfatory results. The performance and the data storage requirements of data-flow algorithms depend on the size of

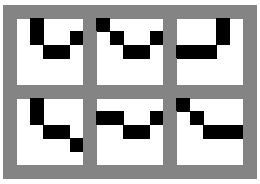

Figure 1: Contour primitives for $k = 2$

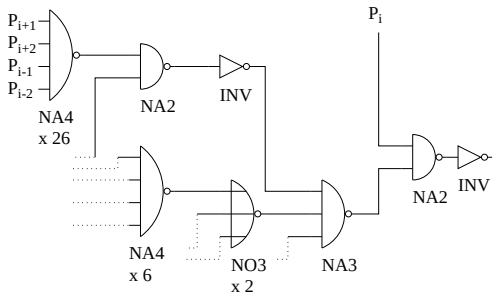

Figure 2: Schematic of a VLSI implementation for $k = 2$

the window defining the neighborhood, as shown in Table 1. The storage size and delay times are for images with 1 bit per pixel, 512 pixels per line, 512 lines per image, 30 images per second.

In this work, all possible contours for $k = 2$ steps were examined in a sistematical way in order to select only the contour patterns that satisfy the criteria defined by Cortez for a vertex located at the center pixel. In Figure 1 are shown the only six primitive patterns selected. All the patterns that satisfy the criteria can be derived from these by mirrowing and rotation by $\frac{\pi}{2}$. Eliminating identical patterns, a list of exactly 26 patterns is obtained.

Figure 2 shows a resumed schematic of a CMOS VLSI implementation. For each pattern, an AND gate detects if that particular pattern is present in the image window. All the AND outputs are ORed in order to determine if any of the patterns has been detected. Note that the center pixel is present in all the possible contours and thus can be factored out in the logic implementation.

The vertex operator for $k = 2$ needs 32 4-input NANDs, one 3-input NAND, 2 2-input NANDs, 2 3-input NORs and 2 inverters. These gates need 286 transistors occupying a chip area of approximately $0.1mm^2$ in a $0.7\mu m$ CMOS technology. The overall gate delay can be estimated to $5ns$.

3 Conclusion

The small chip area accupied and the short gate delay of an implementation for $k = 2$ suggests the use of higher values, for example $k = 4$, in order to determine the curvature of the contour more accurately. In this manner, the decision about the contour point under consideration being or not a vertex can be made more reliably.

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 vertices 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 which shape a contour segment belongs to, 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 vertices 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

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

[2] E. Melcher, J. M. de Carvalho, and *alli. A 2D Shape Boundary Detection Algorithm for VLSI Implementation.* SIBGRAPI'95. São Carlos, SP, October 1995.

[3] E. Melcher, J. M. de Carvalho, and *alli. A Novel Gradient Operator Suited for VLSI Implementation of 2D Shape Recognition.* SBCCI'96. Recife,PE, March 1996.

[4] P.C. Cortez and J.M. de Carvalho. *Modelagem Poligonal de Contornos 2D Usando a Transformada de Hough.* XIII SBT. Águas de Lindóia, SP, September 1995.