JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 2, 302–319 (1985)

# A Systolic Parallel Processor for the Rapid Computation of Multiresolution Edge Images Using the $\nabla^2 G$ Operator

JAMES J. CLARK AND PETER D. LAWRENCE

*Department Of Electrical Engineering,*
*University of British Columbia,*
*Vancouver, British Columbia, Canada*

Received February 13, 1984

This paper describes the application of current parallel processor technology to an important problem in computer vision; the computation of a multiresolution set of edge images from a video camera signal. The edge operator of choice for many image analysis systems is the $\nabla^2 G$ or Laplacian of a Gaussian operator. This operator locates edges by finding the zero crossings of the $\nabla^2 G$ filtered image. A hardware system for the production of a zero crossing "pyramid" is proposed. The hardware processor utilizes a set of "systolic" array processors which implement two-dimensional digital lowpass and bandpass filters as well as the zero crossing detectors. A multilevel interleaved system is described which allows concurrent processing of two sets of image descriptions and ensures that the component processing elements are utilized to the fullest. 

## INTRODUCTION

There has been much interest, recently, in the use of multiple levels of image resolution in image analysis systems. A review of early work in the field of "hierarchical" image processing is provided in [1]. Description of more recent research in this area can be found in [2].

Recent work in stereopsis [3,4] and motion detection [5] have utilized zero crossings of bandpassed images as their basic descriptive element. As processing power available to machine vision system designers increases, so too will $\nabla^2 G$ zero crossings become the edge operator of choice for these systems. Many image processing tasks can be done most efficiently with hierarchical algorithms [2]; thus it seems likely that machines for the computation of hierarchical image descriptions based upon zero crossings of $\nabla^2 G$ filtered images will become standard machine vision hardware.



0743-7315/85 $3.00

## A Hierarchical Zero Crossing Extraction System

The specifications of a typical hierarchical image analysis system may be as follows:

—Four spatial frequency channels, having center frequencies $\omega_0$, $2\omega_0$, $4\omega_0$, and $8\omega_0$.

—Basic low-level image primitives to be used are oriented zero crossings of the spatially bandpass filtered channels.

—The major analysis operations performed by the system are to be done hierarchically using the zero crossing image representation.

In this section we discuss the implementation of the subsystem responsible for the production of the hierarchical zero crossing image representation. This subsystem can be broken up into two sections: the spatial filtering to produce the set of spatial frequency channels, and the zero crossing detection. The spatial filtering is performed as shown in Fig. 1. The lowpass filter and subsampler sections form a two-dimensional decimator or sampling rate re-

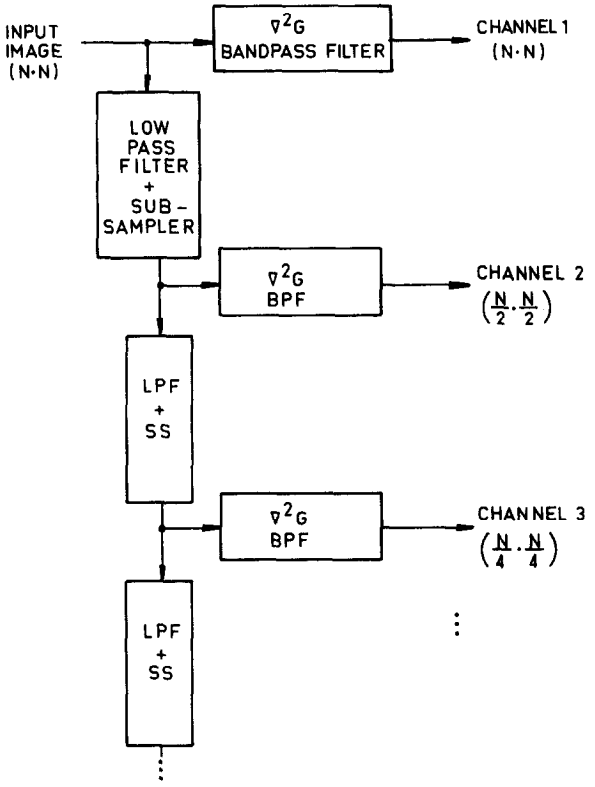


Fig. 1. The filter structure for the computation of a multiresolution set of spatial frequency channels ($\nabla^2 G$ filters with different bandpasses).

ducer. The lowpass filter restricts the maximum frequency of the image to one-half its previous maximum, to limit the aliasing error when the filtered image is subsampled. Each decimation stage reduces the number of image samples by a factor of 4 (by 2 in each of the horizontal and vertical directions). Each lowpass filter section has exactly the same set of filter coefficients. Each successive stage of the decimator is followed by a $\nabla^2 G$ bandpass filter. Even though the coefficients for each of these $\nabla^2 G$ filters are the same, the apparent frequency response of these filters with respect to the input have different center frequencies because of the sampling rate reduction.

This scheme of spatial frequency channel production offers distinct advantages over the direct method in which the input signal is filtered by four separate bandpass filters, each having a different frequency response. The first, and probably least important, advantage is that only one set of filter coefficients is required for all the lowpass filters and for all the $\nabla^2 G$ filters. A more important advantage lies in the fact that the center frequency of the prototypical bandpass filter, with respect to the input, is fairly high, on the order of $\pi/2$ radians.

In designing finite wordlength digital bandpass filters the number of coefficients required to approximate an ideal filter response to a given accuracy is inversely proportional to the center frequency. For example, in the direct method we would require a filter size on the order of $8N \times 8N$ for the lowest (fourth) spatial frequency channel filter (given that the highest frequency filter was of size $N \times N$), compared to the $N \times N$ size filter required in the hierarchical scheme for all the channels. Of course we must take into account the lowpass filters in the hierarchical case but these too will be of constant, not exponential, size. In addition, the structure of our hierarchical filtering system facilitates the pipelining of computation, as will be seen in a later section.

The filtering method described in this paper can be compared with the technique developed by Crowley and Stern [6]. They compute the Difference of Low-Pass (or DOLP) transform of an image, which, for Gaussian lowpass filters, closely approximates the $\nabla^2 G$ filter. Their method uses subsampling to reduce computation and also uses the separability of the Gaussian lowpass filter to reduce the amount of computation. Their method produces bandpass filtering at resolution levels that are a factor of $\sqrt{2}$ apart, in comparison to our method which produces bandpass filters with resolution a factor of 2 apart. In some cases this may be all right, but most current hierarchical algorithms use a resolution reduction factor of 2. For these cases the Crowley and Stern method needs to do twice as much computation as is actually required. The method described in this paper uses the fact that it can be implemented in a pipelined fashion to increase the speed of computation. It is not clear whether the method of Crowley and Stern can be similarly configured.

If we let the lowpass filter prototype have a frequency response $L(\omega_1,\omega_2)$ and the bandpass filter have a frequency response $B(\omega_1,\omega_2)$ then the frequency responses of the spatial frequency channels, referred to the input, are as follows:

$$H_1(\omega_1,\omega_2) = B(\omega_1,\omega_2)$$
$$H_2(\omega_1,\omega_2) = B(2\omega_1,2\omega_2)L(\omega_1,\omega_2)$$
$$H_3(\omega_1,\omega_2) = B(4\omega_1,4\omega_2)L(\omega_1,\omega_2)L(2\omega_1,2\omega_2)$$
$$H_4(\omega_1,\omega_2) = B(8\omega_1,8\omega_2)L(\omega_1,\omega_2)L(2\omega_1,2\omega_2)L(4\omega_1,4\omega_2).$$

Also, due to the signal sampling, we have that $B(\omega_1,\omega_2) = B(\omega_1 + 2\pi k,\omega_2 + 2\pi l)$ and $L(\omega_1,\omega_2) = L(\omega_1 + 2\pi k,\omega_2 + 2\pi l)$ for $k,l = \pm 1, 2, 3, \ldots$.

The prototype two-dimensional lowpass filter was designed by transforming a one-dimensional lowpass filter using the McClellan transformation [8]. This transformation takes a one-dimensional filter with transfer function $F_1(\omega)$ and produces a two-dimensional filter with transfer function

$$F_2(\omega_1,\omega_2) = F_1(f(\omega_1,\omega_2)),$$

where

$$f(\omega_1,\omega_2) = \arccos[0.5(\cos(\omega_1) + \cos(\omega_2) + \cos(\omega_1)\cos(\omega_2) - 1)].$$

This transformation preserves the optimality (if present) of the one-dimensional filter in the two-dimensional design. The one-dimensional filter was designed using the Remez exchange algorithm [9] to produce an optimal half-band lowpass filter (optimal in the sense that the peak approximation error to an ideal lowpass filter is minimized using a minimax criterion).

The peak sidelobe level of the lowpass filter is set by the number of coefficients used in the filter. For $N = 25$ this level is about $-33$ db. One result of the McClellan transformation is that the resulting filter displays octant symmetry. Thus $L(\omega_1,\omega_2) = L(\omega_2,\omega_1) = L(\omega_1,-\omega_2) = L(-\omega_1,\omega_2) = L(\omega_2,-\omega_1) = L(-\omega_2,\omega_1) = L(-\omega_1,-\omega_2) = L(-\omega_2,\omega_1)$. This means that, for $N$ odd, there are only $(N + 1)^2/8 + (n + 1)/4$ unique filter coefficients instead of $N^2$. This can result in a large savings in computation and increased throughput if the symmetry is taken advantage of.

The prototype bandpass filter is, as mentioned earlier, a $\nabla^2 G$ filter with transfer function

$$B(\omega_1,\omega_2) = k(\omega_1^2 + \omega_2^2)\exp(-\sigma^2(\omega_1^2 + \omega_2^2)).$$

This type of filter has a bandwidth of about 1.2 octaves. The value of $\sigma$ used

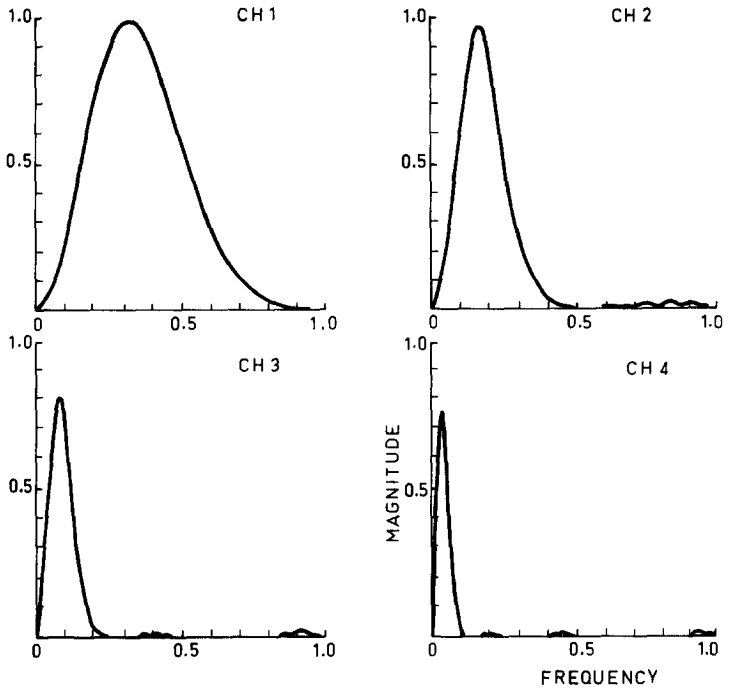


FIG. 2. The magnitude response of the four $\nabla^2 G$ filters. One of the space axes has been suppressed.

in our prototype bandpass filter was $\sqrt{2}$. This value was chosen to trade off between high bandwidth (lower number of filter coefficients) and low aliasing error (due to the sampling of the ideal continuous filter).

The frequency response of the four spatial filters is shown in Fig. 2. One of the spatial frequency axes has been suppressed ($\omega_2 = 0$) for clarity. The peak sidelobe levels are less than $-33$ db in all cases.

Zero crossings are detected by scanning along horizontal lines (rasters) for the following occurrences:

| +\| − | +\| 0 | −\| + | −\| 0 | 0\| + | 0\| − |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |

When one of these is found, a zero crossing is assigned to the position of the left pixel in cases 1 and 3, and to the zero pixel in the other cases. To provide some measure of noise immunity, we ignore all zero crossings whose contrast falls below a given threshold. The threshold used will depend on the expected signal-to-noise ratio of the $\nabla^2 G$ filtered images (which in turn will depend on the processor word length and camera characteristics). In our 8-bit system we used a threshold of 20/255 as the majority of noise-like zero crossings fell below this threshold.

The hierarchical zero crossing image description system described above was simulated on an Amdahl 470 V/8 mainframe computer. Eight bits were used to represent the filter coefficients and image data. Some results of this simulation can be found in [7].

## A Proposed Hardware Implementation Using Systolic Processors

In designing vision systems for industrial applications, one is faced with two, somewhat conflicting, requirements. These are the need for high throughput and the need for small, economical systems. The first requirement is commonly dealt with by using large mainframe computing systems and the second requirement is satisfied by using small computing systems such as microcomputers. Using mainframe computers is very expensive and is an inefficient use of computing resources, since they are usually not designed for dedicated applications. Mini- and microcomputers are economical but are much too slow for complex low-level vision processing.

The most efficient means of implementing industrial vision systems is in special-purpose dedicated hardware. These systems can be made as fast as, or faster than, a mainframe computer, and yet have a cost and size comparable to those of a minicomputer. Drawbacks to using special-purpose hardware include large development costs (since one is not using predesigned components such as a minicomputer) and lack of computational flexibility.

A hardware structure that is special purpose, but can have low design cost and relatively high computational flexibility, is the so-called "systolic" array processor proposed by Kung [10]. Systolic array processors are computational engines which consist of regular arrays of simple computational building blocks. Data and results flow through the array in a serial fashion through the "cells," much like the flow of blood in a human body (hence the term "systolic").

We will now describe a systolic implementation of the hierarchical zero crossing extraction system described earlier. This convolver is an adaptation of the systolic convolver described by Kung and Picard [11]. Like that system, the one described here uses a 1D array to perform the 2D computation and every computational element in the array is used in every clock cycle. In the design discussed here, however, the full utilization of the processors is not obtained through the use of redundant data streams, as in the system of Kung and Picard. Instead, it is the interleaving of two independent data streams which allows the full processor utilization. Also, in the design presented here, there is only one input data path and one result data path, and the input and result data streams both move through the array at the same speed. This simplifies the control of the array. Our systolic convolver, because of the nature of the filtering method (i.e., that the filter kernel is the same at all resolution levels), is able to process the filtering operations at all resolution

levels concurrently by interleaving the data streams at the various levels. Thus it is perfectly suited to our application of computing multiresolution edge images.

Throughout the design, a word length of 8 bits is used for the filter coefficients and image data (some intermediate results may have longer word lengths). Using 8 bits for the $\nabla^2 G$ filter coefficients means that, for $\sigma = \sqrt{2}$, the filter will have a size of $11 \times 11$ coefficients. All other coefficients, outside this $11 \times 11$ region, will be zero. The size of the lowpass filter is largely insensitive to the word length, and is mainly dependent on the required sidelobe attenuation. We took, as a design criterion, a peak sidelobe level of less than $-30$ db. For a lowpass filter size of $25 \times 25$ the peak sidelobe level is $-33$ db.

A systolic array processor implementation of a two-dimensional convolver (for implementing the digital filters) is shown in Fig. 3. The systolic convolver consists of an array of $N \times N$ (25 in this case) computational elements or cells. The $\nabla^2 G$ filter would have 121 cells ($11 \times 11$), and the lowpass filter would have 625 cells ($25 \times 25$). There are four different types of cells (all based on a single cell type).

Input data flow in one direction through the cells and results flow in the other direction. The filter coefficients do not move and each cell is associated with a certain filter coefficient. The shift registers buffer intermediate results and allow the convolution "window" to overlay the proper region of the image (this, in effect, transforms the linear parallel processor into a neighborhood parallel processor). Some signals are added to account for edge effects, which occur when the convolution window overlaps the sides of the image.

The convolution is produced by each cell multiplying its input $x(n)$ by its resident filter coefficient $w(n)$, adding the result to the output $y(n)$ from the previous stage, and passing this sum along to the next cell in the array. By

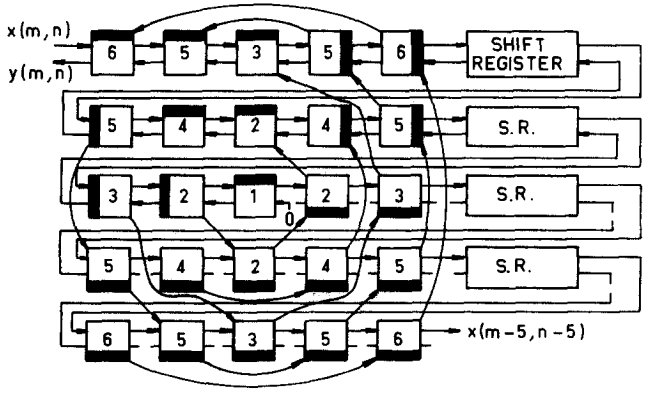


FIG. 3. A $5 \times 5$ systolic convolver using the octal symmetry of the filter. The positions of the black bars on the cells distinguish between different cell types.

the time the first input value $x(1)$ has percolated to the last cell of the array, the first valid output $y(1)$ is available at the output of the first cell in the array. The total delay between data first entering the array and the first valid output is thus $N^2 + (N - 1)S$ cycles, where $S$ is the length of the shift register. However, because of the pipelined nature of the computation, after this delay time a valid output is generated during each cycle. The shift register is used to store the portion of the image line that is not currently in the convolver, hence $S = M - N$, where $M$ is the image line length.

Because the filter has octal symmetry, not all cells are required to perform multiplications. Only those cells drawn with a black bar on the top edge of the cell in Fig. 3 are required to perform multiplications. There are $(N + 1)^2/8 + (N + 1)/$ of these "type 1" cells. Input values from every cell having the same coefficient (denoted by numbers on the cells in Fig. 3) are ripple summed (i.e., asynchronously) in a cyclical manner (illustrated by the arrows in Fig. 3) into the cell containing the multiplier, wherein the sum is multiplied by the filter coefficient. Due to this use of symmetry, not all cells in the array are identical. There are four different types, as shown in Fig. 4. The internal structure of each cell is shown in Fig. 5. Filter coefficients are input serially along the $x$ data path and then strobed into the coefficient register by pulsing the coefficient load signal.

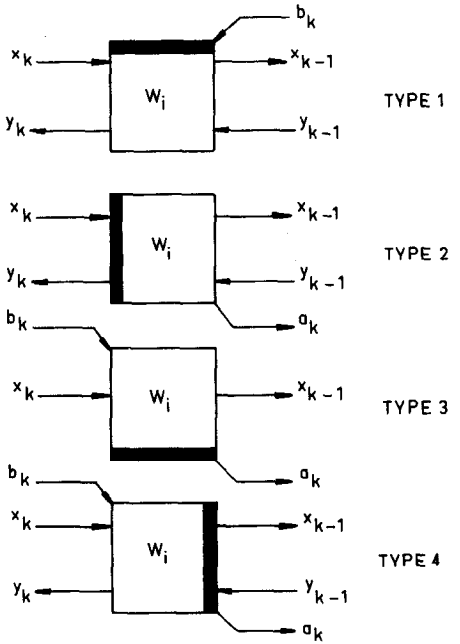


FIG. 4. Data flow for the four different types of computational cells.

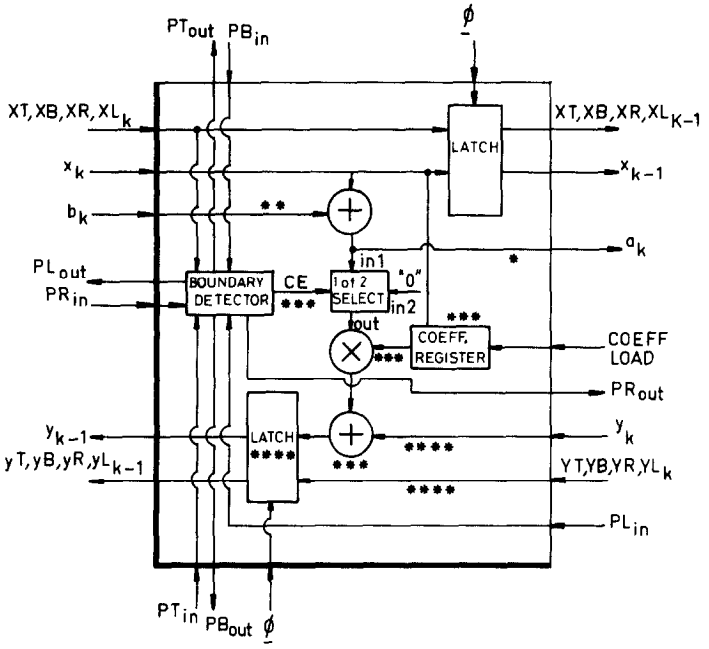


FIG. 5. The structure of the computation cells. The asterisks indicate those data lines or processor cell subelements that are to be omitted for a given cell type.

The computation, in this design, occurs in two stages. During clock phase $\phi_1$ ($\phi_1$ and $\phi_2$ are nonoverlapping clock signals) the multiplier in alternate cells is active and produces $y_{k-1}$. On the trailing edge of $\phi_1$ the input data $x_k$ and the result of computation $y_{k-1}$ are strobed into the latches of the cell. The other cells remain passive during $\phi_1$ and become active during $\phi_2$ (and the first cells become passive). In this manner valid outputs are generated every two cycles (a cycle being the time in which one of the two clock phases is active). The output stream is of the form *aabbccdd* . . . , where a given output is repeated for two clock cycles. Note that the multipliers are active for only half of the time.

To allow the multipliers to work almost continuously we can use an interleaved structure, in which two data streams are interleaved into a single data stream and processed with a multiplexed cell. In this structure, data and results are clocked into one set of latches during $\phi_1$ and into another set of latches during $\phi_2$. Outputs are passed to adjacent cells from those latches that are not being strobed during a given cycle. This ensures that no race conditions are set up and that the data are stable when processed. The output data stream is now of the form $a_1b_1a_2b_2$ . . . , where $a_1a_2$ . . . belongs to one data stream and $b_1b_2$ . . . belongs to the other. Notice that a valid output value is available every clock cycle, although for a given data stream a valid output

value is available every second cycle. The multiplier in each cell now is active during each clock cycle. An interleaved scheme such as this allows the processing of stereo images concurrently which minimizes the distortions induced by the motion of objects during the time between the processing of the two stereo images.

This interleaving provides a constraint on the length of the shift registers. In order for each cell having the same coefficient to be operating on the same data stream (note that alternate cells operate on alternate data streams) at the same time, the shift register length must be even (for $N$ odd). This also means that the image line length must be odd if the filter size is odd. The subsampling is done in such a way that, if the image line length is 255, then the subsampled line lengths will be 127, 63, and 31.

We could implement the system shown in Fig. 1 with a systolic processor for each of the filter blocks. However, the processors for the lower levels (lower spatial frequency) would be underutilized since, due to the subsampling and the synchronous serial flow of data, the lower-level processors would be active only a fraction of the time.

To reduce the processor idle time we can use a single systolic processor for all levels of each lowpass filter, bandpass filter, and zero crossing detector. This is done by multiplexing the computation between levels. Due to the subsampling we need only compute on level 2 once for every four computations at level 1, once on level 3 for every four computations on level 2, and so on. Thus the data stream would have the form $a_{11}b_{11}a_{12}b_{12}a_{13}b_{13}a_{14}b_{14}a_{21}b_{21}a_{15}b_{15}\ldots$, where $a_{ij}$ is the $j$th data word of the first (of the two stereo images) data stream on level $i$.

The internal structure of a two-level interleaved processor is shown in Fig. 6. The circles letters represent points at which data values are tabulated in Table I for 16 successive clock cycles. Table I traces the data flow and illustrates how the convolution operation is performed. The clock phases used in our four-level system, which is a straightforward extension of Fig. 6, are shown in Fig. 7. One can see how a given processor acts on data at a given level once for every four computations at the next-highest level.

Each shift register in Fig. 3 (in the $x$ or $y$ data path, that is) is of size $N_S = 2[(255 - N) + (127 - N) + (63 - N) + (31 - N)] = 952 - 8N$. Data are shifted in on the rising edge of both $\phi_1$ and $\phi_2$.

Figure 8a details the circuitry responsible for handling edge effects. When the convolution window overlaps the image boundaries, a wraparound effect will occur that will result in an incorrect output. This effect is shown in Fig. 8b. To take care of this problem, we use the circuit depicted in Fig. 8a. Each data signal $x$ and $y$ has associated with it four flags; $T$, $B$, $R$, and $L$. $T$ is high when the data signal corresponds to a pixel in the last line of the image, $B$ corresponds to a pixel in the first line of the image, and $R$ and $L$ correspond to pixels in the first and last columns of the image, respectively. These signals are latched and propagated along with the $x$ and $y$ values. In order to eliminate

TABLE I

DATA FLOW FOR A TWO-LEVEL INTERLEAVED SYSTOLIC CONVOLVER

| $\phi_1$ | $\phi_2$ | $\phi_A$ | $\overline{\phi}_A$ | $\phi_B$ | $\overline{\phi}_B$ | $A$ | $B$ | $C$ | $D$ | $E$ | $F$ | $G$ | $H$ | $I$ | $J$ | $K$ | $L$ | $M$ | $N$ | $O$ | $P$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 0 | $a$ | 0 | $aw$ | 0 | 0 | 0 | $a$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | $b$ | $aw$ | $aw$ | 0 | $bw$ | 0 | $a$ | 0 | $b$ | 0 | $a$ | 0 | 0 | $aw$ | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | $c$ | 0 | $aw$ | $cw$ | $bw$ | 0 | $a$ | $c$ | $b$ | 0 | 0 | 0 | 0 | $aw$ | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | $d$ | $cw$ | $aw$ | $cw$ | $bw$ | $dw$ | $a$ | $c$ | $b$ | $d$ | $c$ | 0 | 0 | $aw$ | $cw$ | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | $e$ | $dw$ | $aw$ | $(ew + cw)$ | $bw$ | $dw$ | $a$ | $e$ | $b$ | $d$ | $d$ | 0 | $dw$ | $aw$ | $cw$ | $cw$ |
| 0 | 1 | 0 | 0 | 0 | 1 | $f$ | $(ew + cw)$ | $aw$ | $(ew + cw)$ | $bw$ | $(fw + dw)$ | $a$ | $e$ | $b$ | $f$ | $e$ | 0 | $dw$ | $aw$ | $ew$ | $dw$ |
| 1 | 0 | 0 | 0 | 1 | 0 | $g$ | $(fw + dw)$ | $aw$ | $(gw + ew)$ | $bw$ | $(fw + dw)$ | $a$ | $g$ | $b$ | $f$ | $f$ | 0 | $fw$ | $aw$ | $ew$ | $ew$ |
| 0 | 1 | 0 | 0 | 0 | 1 | $h$ | $(gw + ew)$ | $aw$ | $(gw + ew)$ | $bw$ | $(hw + fw)$ | $a$ | $g$ | $b$ | $h$ | $g$ | 0 | $fw$ | $aw$ | $gw$ | $fw$ |
| 1 | 0 | 0 | 0 | 1 | 0 | $i$ | $(hw + fw)$ | $aw$ | $(iw + gw)$ | $bw$ | $(hw + fw)$ | $a$ | $i$ | $b$ | $h$ | $h$ | 0 | $hw$ | $aw$ | $gw$ | $gw$ |
| 0 | 1 | 0 | 0 | 0 | 1 | $j$ | $(iw + gw)$ | $aw$ | $(iw + gw)$ | $bw$ | $(jw + hw)$ | $a$ | $i$ | $b$ | $j$ | $i$ | 0 | $hw$ | $aw$ | $iw$ | $hw$ |
| 1 | 0 | 1 | 0 | 0 | 0 | $k$ | $bw$ | $(kw + aw)$ | $(iw + gw)$ | $bw$ | $(jw + hw)$ | $k$ | $i$ | $b$ | $j$ | $b$ | $bw$ | $hw$ | $aw$ | $iw$ | $aw$ |
| 0 | 1 | 0 | 1 | 0 | 0 | $l$ | $(kw + bw)$ | $(kw + aw)$ | $(iw + gw)$ | $(lw + bw)$ | $(jw + hw)$ | $k$ | $i$ | $l$ | $j$ | $k$ | $bw$ | $hw$ | $kw$ | $iw$ | $bw$ |
| 1 | 0 | 0 | 0 | 1 | 0 | $m$ | $(jw + hw)$ | $(kw + aw)$ | $(mw + iw)$ | $(lw + bw)$ | $(jw + hw)$ | $k$ | $m$ | $l$ | $j$ | $j$ | $bw$ | $jw$ | $kw$ | $iw$ | $iw$ |
| 0 | 1 | 0 | 0 | 0 | 1 | $n$ | $(mw + iw)$ | $(kw + aw)$ | $(mw + iw)$ | $(lw + bw)$ | $(nw + jw)$ | $k$ | $m$ | $l$ | $n$ | $m$ | $bw$ | $jw$ | $kw$ | $mw$ | $jw$ |
| 1 | 0 | 0 | 0 | 1 | 0 | $o$ | $(nw + jw)$ | $(kw + aw)$ | $(ow + mw)$ | $(lw + bw)$ | $(nw + jw)$ | $k$ | $o$ | $l$ | $n$ | $n$ | $bw$ | $nw$ | $kw$ | $mw$ | $mw$ |
| 0 | 1 | 0 | 0 | 0 | 1 | $p$ | $(ow + mw)$ | $(kw + aw)$ | $(ow + mw)$ | $(lw + bw)$ | $(pw + nw)$ | $k$ | $o$ | $l$ | $p$ | $o$ | $bw$ | $nw$ | $kw$ | $ow$ | $nw$ |

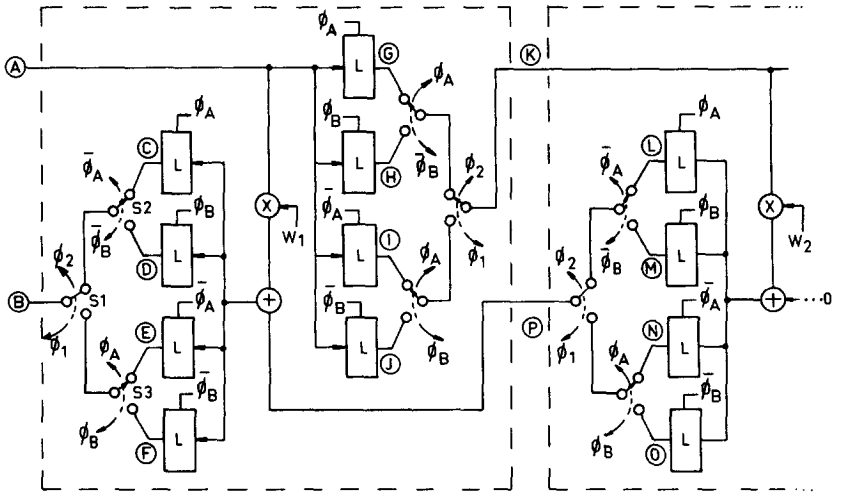


FIG.. 6. A two-level dual-image data stream (stereo) systolic convolver processing element.

the wraparound effect, we disable the computations performed on the cross-boundary points. To do this we must first determine which cells in our array contain these cross-boundary points and then use a zero multiplicand instead of the pixel value in the convolution computation for this cell. In this manner, the image will be processed as if it were surrounded by a region of zeros. Cross-boundary points are determined by the signals *PR*, *PL*, *PT*, and *PB*. These signals ripple through the array as shown in Fig. 8 and are generated

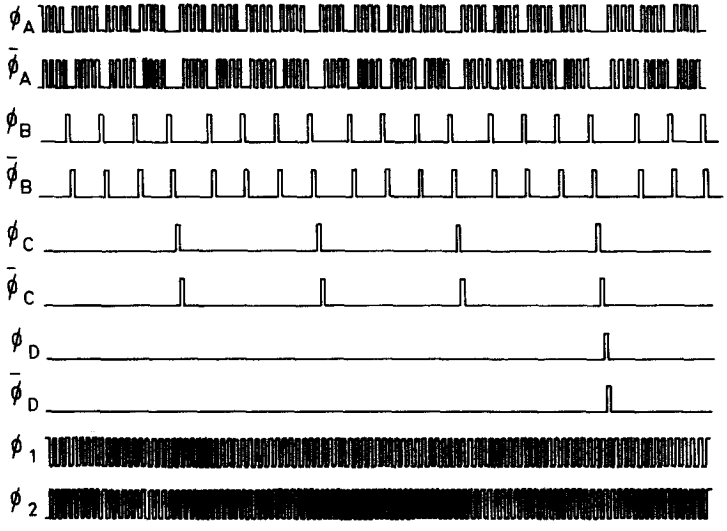


FIG. 7. The 10 clock phases for a four-level dual-image data stream systolic convolver.

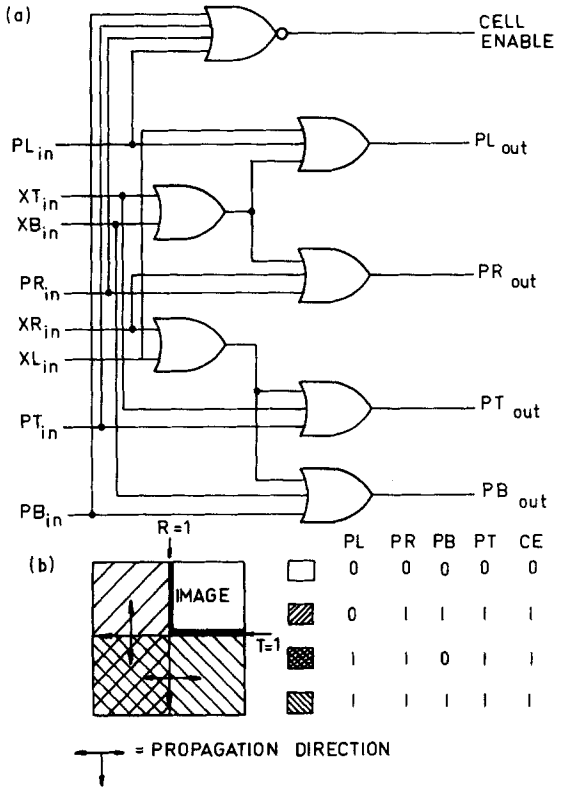


FIG. 8. (a) The circuitry for handling edge effects. (b) An example of the propagation of the image edge signals, required to inhibit the processing of cross-edge pixels.

from the $L$, $R$, $T$, $B$, and $P$ signals from neighboring cells. The logic equations for generating these signals are as follows:

$$PL_{\text{out}} = (L_{\text{in}})\text{OR}(PL_{\text{in}})\text{OR}(T_{\text{in}})\text{OR}(B_{\text{in}})$$

$$PR_{\text{out}} = (R_{\text{in}})\text{OR}(PR_{\text{in}})\text{OR}(T_{\text{in}})\text{OR}(B_{\text{in}})$$

$$PT_{\text{out}} = (T_{\text{in}})\text{OR}(PT_{\text{in}})\text{OR}(L_{\text{in}})\text{OR}(R_{\text{in}})$$

$$PB_{\text{out}} = (B_{\text{in}})\text{OR}(PB_{\text{in}})\text{OR}(L_{\text{in}})\text{OR}(R_{\text{in}}).$$

The cell computation is "zeroed" when $(PT_{\text{in}})\text{OR}(PB_{\text{in}})\text{OR}(PR_{\text{in}})\text{OR}(PT_{\text{in}})$ is logically high. The cells along the boundary of the convolver array ground the $P$ signals that would otherwise not be connected to any adjacent cell.

The subsampler design is shown in Fig. 9. The subsampler stores every other image line at a given level (except the lowest level) and outputs every other sample in this image line to the next-lower level. Hence, there is a net sample rate reduction of 4 : 1. The multiplexer shown on the output of the

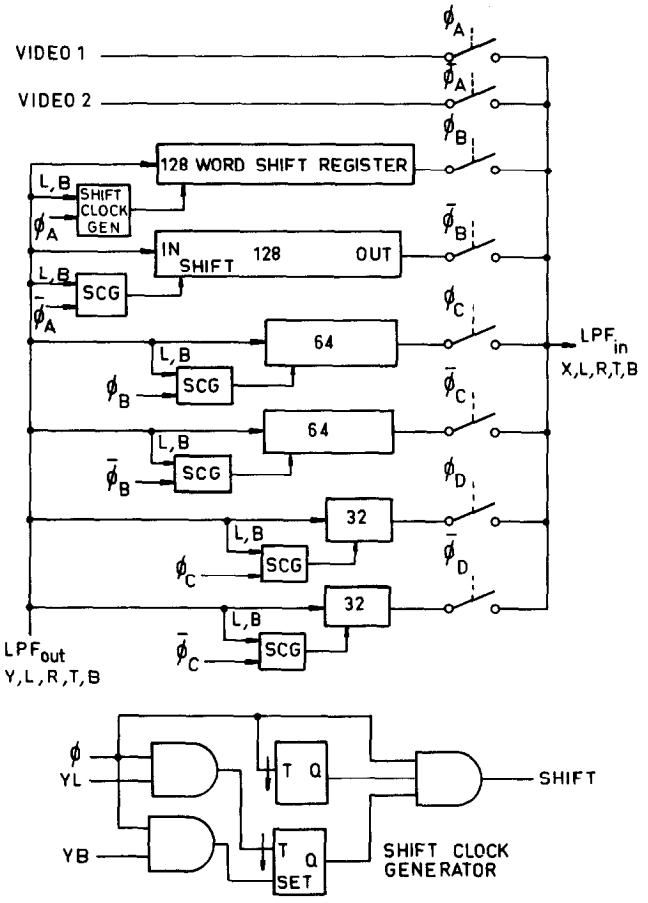


FIG. 9. The design of the subsampler.

subsampler performs the transmission of these samples to correct (next-lowest) level. Gating of the higher-level clock is performed to enable the sampling of alternate input lines. The *B* signal resets the subsampler so that the first image line is sampled and not the second, thus ensuring that the *T* and *B* signals are not lost in the subsampling process.

The overall operation of the hierarchical zero crossing extraction system is shown in Fig. 10. Note that, due to the input multiplexing between the video signals and the lowpass filter outputs, the video signals are sampled irregularly (or semiregularly). This means that, if the video sources output data regularly, there must be a FIFO type of input buffer that stores an input line so that it can be sampled irregularly. This FIFO should be long enough to contain one entire input image line.

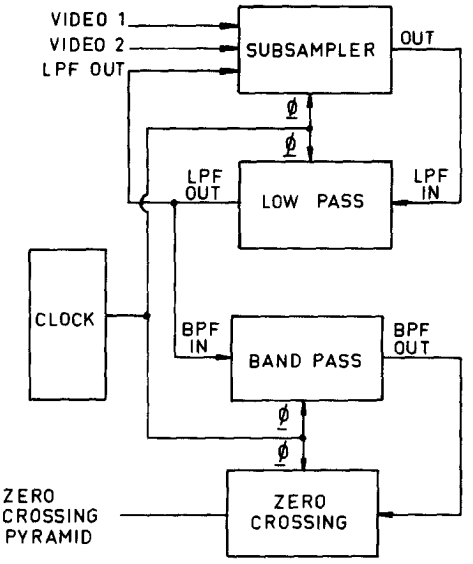


FIG. 10. Block diagram of a hierarchical edge detection system.

## PERFORMANCE EVALUATION

We can calculate the number of clock cycles (here defined as one-half of the $\phi_1$ or $\phi_2$ cycles) needed to complete an entire image filtering operation with the systolic array processor, as follows.

We will define the filtering time, $T_F$, as the amount of time between the first valid output of the bandpass filter for image 1, level 1, and the last valid output of the bandpass filter (i.e., for image 2, level 4, line 31, pixel 31). We further define the initial delay time, $T$, as the amount of time between input of the first data word into the lowpass filter array and the output of the first valid data value from the bandpass filter (for image 1, level 1).

Let $T_L$ be the time required for one image line to be processed. This time will be the same for both the lowpass and bandpass filters and will include the time required for processing one line for all image resolution levels, since, due to the interleaved nature of the computation, a line at a given level is not processed fully until the corresponding lines at the lower resolution levels have been fully processed. This can be seen in Fig. 9, keeping in mind that each separate clock signal indicates when a given image resolution level is being acted on. Let $T_S$ be the delay time added by the subsampling process. Let $N_L$ be the lowpass filter size and $N_B$ the bandpass filter size.

It can be shown that the line processing time is given by the relation

$$T_L = 2 \times (255 + 64 + 16 + 4) = 678.$$

The delay time can be written as

$$T_D = (N_L - 1)T_L + (N_S - 1)T_L + 2(N_L + N_B) + T_S$$

and is the time required for the passage of data once through the lowpass filter, the subsampler, and the bandpass filter. $T_S$ is equal to $2T_L$, as two image lines are effectively stored in the subsampler. After the delay time there is a valid output data value every clock cycle. Since there are a total of $N_V = 2(255^2 + 127^2 + 63^2 + 31^2)$ valid output pixel positions in an image pair the total filtering time, $T_F$, is equal to $N_V$. For $N_L = 25$ and $N_B = 11$ we have that

$$T_L = 678, \quad T_S = 1{,}356, \quad T_D = 24{,}480, \quad T_F = 172{,}168.$$

In Table II we have tabulated the filtering time, $T$, the throughput $(1/T_F)$, and the effective cycle time that would be obtrained for a number of different systems. The effective cycle time is calculated by dividing the total filtering time by the number of cycles in the systolic implementation ($T_F = 172{,}168$). This yields a measure of comparison between two systems that are performing the same convolution using different algorithms. These systems include: a VAX 11/750 running a Fortran 77 convolution routine (unoptimized, inefficient), a 2D convolution routine (CONV2D) running on an FPS-100 array processor, the systolic processor shown above using a CMOS shift and add multiplier (designed using 5-$\mu$m design rules, with cycle time obtained by Spice simulation), and the systolic processor using a SN74S558 45-nsec (typical) combinatorial multiplier made by Monolithic Memories Inc. Note that the numbers for the FPS-100 and VAX 11/750 convolution programs do not include the time required for I/O, which can eat up a large portion of the total execution time. Cycle time is chiefly dependent on the multiplier speed and clever multiplier design (such as pipelining the multiplier itself) may lower the processor cycle time. Reduction of the filter sizes or coefficient word lengths will reduce circuit complexity and processor cycle time; however, these reductions will be obtained at the expense of degraded performance (in the form of aliasing error, filter ringing, quantization error, and so forth).

TABLE II
A COMPARISON OF FILTERING TIMES FOR VARIOUS COMPUTATIONAL SYSTEMS

| | VAX 11/750 | FPS-100 | Shift/add | SN74S558 |
|---|---|---|---|---|
| Filtering time | 2,295 sec | 0.603 sec | 0.120 sec | 0.0077 sec |
| Effective cycle time | 13,300 $\mu$sec | 3.5 $\mu$sec | 0.70 $\mu$sec | 0.045 $\mu$sec |
| Throughput (image pair/sec) | 0.00044 | 1.66 | 8.30 | 129 |

## SUMMARY

Algorithms for the digital implementation of a system for the extraction of a multiresolution zero crossing image representation are detailed. A hardware system design of this system utilizing a pipelined, multilevel, interleaved systolic array processor is described. Currently, work is under way at the University of British Columbia in implementing such a system in CMOS, using the facilities of the Canadian VLSI Implementation Centre (VLSIIC) coordinated by Queens University. It has been found that the individual type 1 cells using fast combinatorial multipliers cannot, at present, be fabricated entirely on one $0.5 \times 0.5\text{-cm}^2$ die using 5-$\mu$m design rules. A CMOS shift and add multiplier has been designed which will allow an entire type 1 cell to be placed on such a die, but the slow speed of this type of multiplier compared to combinatorial designs rules out its use for most real-time applications. The type 4 cell has been implemented in NMOS, along with a clock generator circuit that can produce the 10 separate clock signals shown in Fig. 7. It is hoped that we will, in the near future, be able to implement a complete type 1 cell, with a combinatorial multiplier, in CMOS using the facilities of the VLSIIC.

The system described in this paper is not limited to computation of multiresolution edge images. It can be used in any filtering type of application that processes data at a number of resolutions. Some examples are image compression [12], image segmentation [13], and motion analysis [14]. The key contribution of the work presented here is the development of an efficient system for performing the multiresolution filtering required by these applications.

## ACKNOWLEDGMENTS

The authors gratefully acknowledge the support and services provided by the Forest Engineering Research Institute of Canada (FERIC) in this study. Funding was also supplied by the Natural Sciences and Engineering Research Council of Canada (NSERC) under Grant A-4924 and by the B.C. Science Council under Grant 52(RCG,8) and G.R.E.A.T. awards.

## REFERENCES

1. Tanimoto, S. L. Regular hierarchical image and processing structures in machine vision. In Hanson, A., and Riseman, E. (Eds.). *Computer Vision Systems*. Academic Press, New York/London, 1978
2. Rosenfeld, A. (Ed.). *Multiresolution Image Processing and Analysis*. Springer-Verlag, New York/Berlin, 1984.
3. Marr, D., and Poggio, T. A computational theory of human stereo vision. *Proc. Roy. Soc. London Ser. B.* **204** (1979), 301–328.
4. Grimson, W.E.L. "A computer implementation of a theory of human stereo vision. *Philos. Trans. Roy. Soc. London Ser. B* **292** (1981), 217–253.
5. Marr, D., and Ullman, S. Directional selectivity and its use in early visual processing. *Proc. Roy. Soc. London Ser. B.* **211** (1981), 151–180.

6. Crowley, J. L., and Stern, R. M. Fast computation of the difference of low-pass transform. *IEEE Trans. Patt. Anal. Mach. Intell.* **6**, 2 (1984), 212–222.
7. Clark, J. J., and Lawrence, P. D. A hierarchical image analysis system based upon oriented zero crossings of bandpassed images. In Rosenfeld, A. (Ed.). *Multiresolution Image Processing and Analysis*. Springer-Verlag, New York/Berlin, 1984.
8. McClellan, J. H. The design of two-dimensional digital filters by transformations. *Proc. 7th Ann. Princeton Conf. Inf. Sci., Syst.*, 1973.
9. McClellan, J. H., Parks, T. W., and Rabiner, L. R. A complete program for designing optimum FIR linear phase digital filters. *IEEE Trans. Audio. Electroacoust.* **AU-21** (1973), 506–526.
10. Kung, H. T. Why systolic architectures? *Computer* **15** (Jan. 1982), 37–46.
11. Kung, H. T., and Picard, R. L. Hardware pipelines for multi-dimensional convolution and resampling. *Proc. IEEE Conf. on Computer Architecture for Pattern Analysis and Image Database Management*, 1981, pp. 273–278.
12. Burt, P. J., and Adelson, E. The Laplacian pyramid as a compact image code *IEEE Trans. Comm.* **31** (1983), 532–540.
13. Hanson, A. R., and Riseman, E. M. Segmentation of natural scenes. In Hanson, A., and Riseman, E. (Eds.). *Computer Vision Systems*. Academic Press, New York/London, 1978, pp. 129–163.
14. Mutch, K. M., and Thompson, W. B. Hierarchical estimation of spatial properties from motion. In Rosenfeld, A. (Ed.). *Multiresolution Image Processing and Analysis*. Springer-Verlag, New York/Berlin, 1984.