PARALLELIZING INFINITE IMPULSE RESPONSE FILTERS

RACHID BENLAMRI
Computer Engineering Department
Etisalat College of Engineering
Po. Box 980 Sharjah
UNITED ARAB EMIRATES

Abstract: In this paper, we present a parallel algorithm for edge detection based on Infinite Impulse Response filter (IIR filter). In particular, the Infinite size Symmetric Exponential Filter (ISEF) that is an optimal IIR filter and computationally efficient smoothing filter is studied. The proposed algorithm exploits efficiently all aspects of potential parallelism (spatial parallelism, temporal parallelism and systolism) inherent in the studied edge detection algorithms. The designed concurrent algorithm is expressed in terms of a collection of concurrent processes communicating and synchronizing in an efficient way in order to speed up the low-level operations.

Key-words: Edge detection, Infinite Impulse Response filter, ISEF filter, Spatial parallelism, Temporal parallelism.

1 Introduction

Computer vision has been an area where the computational demand is far above the capacity of a conventional sequential computer. This is due to the large amount of data to be processed by the time-consuming vision tasks. To overcome this problem, computer vision can take advantages of parallel architectures. In fact, parallel systems play a central role in the future of computing. This vital role is increasing rapidly as the costs involved in improving semi-conductor circuit speed and density become higher. In this paper, we describe a parallel algorithm for speeding up edge detection based on the Infinite Impulse Response filter (IIR filter).

The low-level tasks operate on an input image and are characterized by uniform and local computation over the whole image array. These characteristics lead to efficient parallel implementation of such algorithms and ease control and scheduling. The processing model of the IIR filters presents a potential parallelism (spatial parallelism, temporal parallelism and systolism). The proposed concurrent algorithm exploits efficiently all these types of parallelism inherent in the segmentation process. Thus, it is successfully used for speeding up the low-level operations that provide less noisy and very well localized edges.

The remaining of the paper is organized as follows. Section 2 describes edge detectors based on IIR filters and in particular a detailed description of ISEF filter is introduced. The potential parallelism inherent in ISEF filter and the description of the proposed parallel algorithm are presented in section 3. Finally, conclusions from the work are drawn and further research work is suggested.

2 IIR Filter Based Edge Detection

Local intensity discontinuities, commonly referred to as edges, are important attributes for an image because they correspond in general to the important changes of physical or geometrical properties of objects in the scene and they are widely used as primitives in pattern recognition and image matching. These edges can be detected by maxima of gradient or the zero crossing of the second derivatives calculated by some differential operators. Many methods for edge detection in noisy images have been proposed such as Robert gradient, Sobel operators and many others [1-4]. However, these early differential operators are very sensitive to noise because numerical differentiation of images is an ill-posed problem in the sense of Hadamard [5]. Differentiation needs to be regularized by a regularization filtering operation before differentiation. Therefore, a preprocessing such as

smoothing is in general necessary to reduce the noise.

A well-known smoothing filter is the Gaussian filter and the edges can be detected by a Laplacian-Gaussian filter. But there is an essential difficulty of the Laplacian-Gaussian filter, which is the contradiction between smoothing effect and the precision of localization. To overcome this difficulty, many exponential filters which are Infinite Impulse Response filters were proposed [6,7]. In order to present the processing model of such IIR filters, the Infinite Size Exponential filter (ISEF filter) is studied in more details. This filter proposed by Shen and Castan is an optimal linear filter based on one-step model (a step edge and white noise) and the multi-edge model. Moreover, this optimal smoothing filter is a symmetric exponential filter of an infinitely large window size and can be realized by a very simple and computationally efficient recursive algorithm. A theoretical analysis for the performance of this filter [8] has shown that this filter is superior to the other current filters. In addition, this filter presents many advantages such as it has a constant time of execution for different sizes of the operator and is readily amenable to parallel implementation.

In 1-D, the normalized Infinite size Symmetric Exponential Filter (ISEF) has the form [7,9]:

$$\begin{aligned} f_L(x) &= C.a_0.(1-a_0)^{|x|} \\ &= f_1(x) \otimes f_2(x) \\ &= C.(f_1(x) + f_2(x) - a_0.\delta(x)) \end{aligned}$$

where: $C = 1/(2-a_0)$

$\otimes$ means the convolution

$$f_1(x) = \begin{cases} a_0.(1-a_0)^x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

$$f_2(x) = \begin{cases} 0 & x > 0 \\ a_0.(1-a_0)^{-x} & x \leq 0 \end{cases}$$

Because the exponential filter is an IIR filter, the functions $f_1(x)$ and $f_2(x)$ are realized by a computationally efficient recursive algorithm. Assume that $I(x,y)$ is the input image, $I_1(x,y) = I(x,y) \otimes f_1(x)$ and $I_2(x,y) = I(x,y) \otimes f_2(x)$, the recursive algorithms are as follows:

$$\begin{aligned} I_1(x,y) &= I_1(x-1,y) + a_0.(I(x,y) - I_1(x-1,y)) \\ I_2(x,y) &= I_2(x+1,y) + a_0.(I(x,y) - I_2(x+1,y)) \end{aligned}$$

In 2-D, as the exponential function is separable, the exponential filter can be written as follows:

$$f(x,y) = f_L(x).f_L(y)$$

Therefore, the first and second directional derivatives of input images can be calculated by the recursive algorithms f_1 and f_2 and calculated simultaneously as shown below [7,8]:

$$\begin{aligned} I_x(x,y) &= \frac{\partial}{\partial x}(I(x,y) \otimes f(x,y)) \\ &= I(x,y) \otimes f_1(y) \otimes f_2(y) \otimes (f_2(x) - f_1(x)) \end{aligned}$$

$$\begin{aligned} I_{xx}(x,y) &= \frac{\partial^2}{\partial x^2}(I(x,y) \otimes f(x,y)) \\ &= I(x,y) \otimes f_1(y) \otimes f_2(y) \otimes (f_2(x) + f_1(x)) \\ &\quad - 2.\mathrm{I}(x,y) \otimes f_1(y) \otimes f_2(y) \end{aligned}$$

Similarly:

$$\begin{aligned} I_y(x,y) &= I(x,y) \otimes f_1(x) \otimes f_2(x) \otimes (f_2(y) - f_1(y)) \\ I_{yy}(x,y) &= I(x,y) \otimes f_1(x) \otimes f_2(x) \otimes (f_2(y) + f_1(y)) \\ &\quad - 2.\mathrm{I}(x,y) \otimes f_1(x) \otimes f_2(x) \end{aligned}$$

With this algorithm, we can calculate simultaneously the first and second directional derivative I_x and I_{xx} (or I_y and I_{yy}) of input image as shown on fig.1. Using the differential operators derived from the 2-D exponential filter, Shen and Castan have proposed several methods for edges detection [7]. One uses maxima of gradient (GEF), another uses the zeros crossing of second directional derivative along the gradient (SDEF), etc. For instance, GEF method consists of calculating firstly Ix and Iy then the gradient can be determined approximately for every point in the input image by:

$$\begin{aligned} \nabla I(x,y) &= (I_x(x,y), I_y(x,y)) \\ |\nabla I(x,y)| &= \sqrt{I_x(x,y)^2 + I_y(x,y)^2} \end{aligned}$$

Non maxima in the gradient magnitude image is then suppressed and thresholded by hysterisis. In the following sections, a detailed description of the parallel version of the above described edge detector is presented.

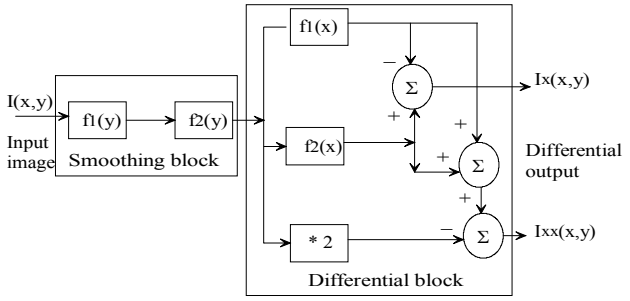

Fig.1. Processing model of ISEF filter4 [7,9]

3 Strategy for Parallel Algorithm Design

Despite the hardware technology improvements of parallel computers, there is a substantial misalignment between the potential user's expectations and the actual usability of software for such systems [10-15]. This is due to many machine dependents aspects that have a significant impact on the final performance obtained. Consequently, the design of parallel systems requires care and accuracy if the results obtained from the system are to be useful. These requirements mean that an accurate model of the system to be implemented must be derived and this model carefully mapped onto the target architecture [16]. Furthermore, the concurrency inherent in the processing model should be used to the full in any implementation of the parallel system. Therefore, the design method should include techniques that exploit and control the parallel nature of the system. These techniques concern mainly:

- Partitioning of the software into appropriate processes
- The design of the interprocess communication structure
- The internal design of each individual process.
- Task allocation, and task scheduling

In the following sections we look at the above-mentioned constraints in more details, aiming to translate the conceptual model of the problem into an algorithm that is formulated in parallel form with the objective of achieving a better performance in its execution.

3.1 Inherent Parallelism in ISEF Filter

In order to achieve the maximum speed up, a system must be designed to be able to exploit the characteristics of the problem to be solved in the most efficient way. The processing model of the optimal exponential filter (ISEF) which is an IIR filter shows a potential parallelism. As described in section 2, the whole processing task is split into two functional blocks performing respectively a 1-D smoothing operation and a 1-D differentiation into a perpendicular direction (fig.1). This temporal characteristic suggests the use of a two sets of processes working in a pipeline fashion. The first set of processes performs the smoothing operation and then communicates the results to the second pipe which carries out differentiation. The advantage of such configuration is the overlapping of successive image frames, hence, increasing the overall throughput of the algorithm.

As shown in fig.1, in each stage of the pipe, the functions f_1 and f_2 in their recursive form present other aspects of parallelism (spatial parallelism and systolism). By spatial parallelism here we mean that each image row and column can be independently and simultaneously processed during the operation of smoothing and differentiation. In other words, each image row (column) might be allocated to a concurrent process. The simultaneous data allocation is performed to meet the required throughput of the concurrent algorithm, and hence, achieving a considerable processing time reduction for the functions f_1 and f_2. In addition to this, all pixels in each image row (column) are characterized by uniform and local computation that is performed in a pipeline fashion, in the sense that the processing at a specific pixel is considered if and only if the one preceding is achieved. The progress in time of such processing imitate the systolic model of processing where a vector of processes, each of which associated to a pixel, is fed by the same set of data. The use of systolism reduces considerably the computational complexity and scheduling problems. Furthermore, by exploiting systolism at each row (column), an overlapping is enabled between rows (columns) of different image frames. This fine-grained partitioning reduces significantly the processing time of a sequence of images.

3.2 Parallel Algorithm Structure

Due to the potential parallelism inherent in the processing, the algorithm requirements in terms of communication, control and scheduling are very high. Therefore, the parallel algorithm structure

should be carefully designed to satisfy these vital constraints. To outline the logical structure of such concurrent algorithm and for sake of readability, an Occam like pseudo- code is used to express concurrency, communication and scheduling. It should be noted here that other parallel programming languages could be used to describe the proposed parallel algorithm.

Edge detection using ISEF filter provides reliable and accurate edge maps and in revenge, requires an enormous computational demand. To exploit inherent parallelism described in the previous section, the proposed algorithm consists of three categories of concurrent processes that are: image allocator process, smoothing processes and differentiation processes. The smoothing and differentiation operations are carried out using arrays of processes as shown in fig.2. The algorithm structure is designed so that the first and second derivatives in the horizontal direction (Ix, Ixx) and those in the vertical direction (Iy, Iyy) are generated simultaneously. Hence, enabling a concurrent computation of the derivatives (Ix, Ixx, Iy, Iyy) for a sequence of images in a systolic way. The following Occam process describes the overall architecture of the concurent algorithm.

```
PAR
   Image_Allocator ( )
   Smoothing_Stage( )
   Differentiation_Stage( )
```

3.2.1 Image Allocator Process

The image allocator process divides the image space into separate rows and columns and then allocates every couple (i^{th} row, i^{th} column)$_{i=1,n}$ to the first process on the i^{th} row of the array of processes as shown in fig.3. Each couple of image row and column consists of pairs of pixels, which enables the simultaneous computation of all derivatives (Ix, Ixx, Iy, Iyy). The functional description of the processing at this stage can be summarized by the following code:

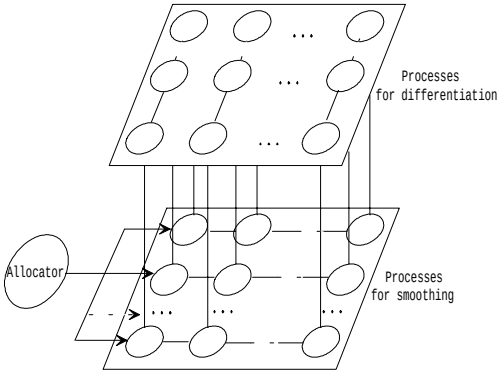

Fig.2. The overall structure of the parallel algorithm for edge detection

```
PAR  i = 0 FOR N
   SEQ
      Get_couple(row_i, column_i)
      Ch_allocator[i] ! couple
```

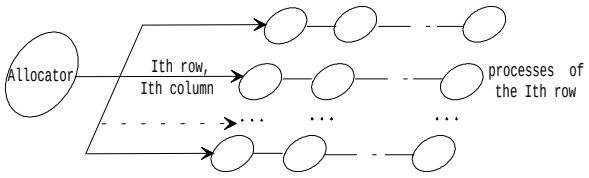

Fig.3. Image allocator process

3.2.2 Processes for Smoothing

Processes for the smoothing operation are concerned with the computation of the one-dimensional (1-D) smoothing function. The way processes are logically interconnected shows the independence in processing separate rows and columns. Only pairs of image pixels or resulting pixels are communicated between neighboring processes in a systolic way. For each process, the 1-D smoothing operation is achieved using two concurrent sub-processes (producer and consumer) as shown in fig.4. The process describing the overall processing at this stage is as follows:

```
PAR  i=0 FOR N
   PAR  j=0 FOR N
      SEQ
         Initialisations
         PAR
            Producer(i,j)
            Consumer(i,j)
```

The producer process receives from his right neighbor as shown in figure 4 the following sequence: a block of data already processed; its own

image data; and the data that should be allocated to its left neighbors. After processing its own data, a copy of the results are sent to a FIFO operating buffer to be used by the consumer process (see fig.4), and another copy of those results is forwarded to the left producer process. Hence, the size of data traversing processes along each row decreases gradually as going to the left end. The pseudo-code describing the producer process is given below:

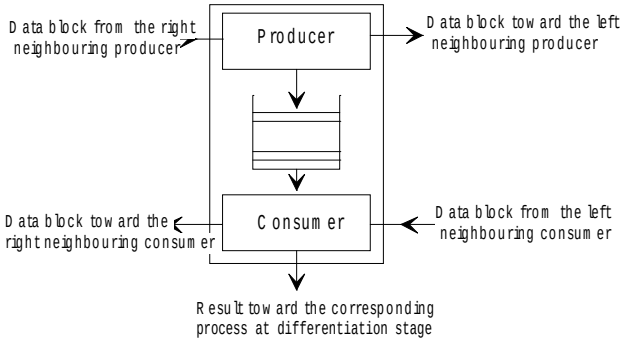

Fig.4. A typical concurrent process for smoothing

```
WHILE TRUE
  SEQ
    Ch_producer[i][j] ? Data_block
    Process the corresponding pair of data
    PAR
      Save the result in a FIFO operating buffer
      Ch_producer[i][j+1] ! Result_block
```

The consumer process performs a similar processing to the one described above. After receiving both of the partial results from the producer to which it is connected and those results that are communicated by its left neighboring consumer process, a local processing is performed and results are sent to both of the processes performing differentiation and the right neighboring consumer (see fig.4). The following pseudo-code summarizes the processing at this level:

```
WHILE TRUE
    SEQ
      Ch_consumer[i][j+1] ? Data_pair
      Process the  pair of data
      PAR
        Ch_consumer[i][j] ! Result_pair
        Ch_stage_1-2[i][j] ! Result_pair
```

The use of a FIFO operating buffer as a communication tool between the producer and consumer sub-processes is advantageous in the sense that the time separating the end of a production and the beginning of its consummation is spent processing data of different image sequence. Hence enabling overlapping the 1-D smoothing function of a sequence of images.

3.2.3 Processes for differentiation

In a pipeline fashion, the set of processes at this stage performs the 1-D differentiation operation into a perpendicular direction as shown in fig.2. For each process, four concurrent sub-processes apply a set of elementary processing operations (see ISEF model of processing) to each pair of data received from the smoothing stage in order to compute (Ix, Iy, Ixx and Iyy). The pseudo-code describing the processes for differentiation is given below:

```
PAR  j=0 FOR N
    PAR  i=0 FOR N
        SEQ
            Initialisations
            PAR
                Receiver(j,i)
                Producer_1(j,i)
                Producer_2(j,i)
                Consumer(j,i)
```

The receiver process stores pairs of data resulting from the smoothing stage in a FIFO operating buffer for later processing. Once a pair of data is consumed, it is then removed from the buffer. The pseudo-code describing the receiver process is given below :

```
WHILE TRUE
  SEQ
    Ch_stage_1-2[i][j] ? Data_pair
    Store data in a FIFO operating buffer
    WHILE  buffer is full
        SKIP
```

The producer sub-processes P_1 and P_2 performs the same elementary processing operations but in opposite direction using pairs of data stored in the receiver buffer and results communicated by neighbouring processes. The intermediate results of these sub-processes are saved in their own buffers and also communicated to neighbouring processes. The pseudo-code describing the producer sub-process P1 is given below:

```
WHILE TRUE
  SEQ
  Ch_producer_1[i-1][j] ? Data_pair
  WHILE  (Producer_1_buffer is full)   OR
```

```
        (Receiver_buffer is empty)
      SKIP
    Process the received pair of data
    PAR
      Store the results in buffer
      Ch_producer_1[i][j] ! Result_pair
```

The consumer sub-process provides the resulting derivatives Ix, Iy, Ixx and Iyy. The process (i, j) at differentiation stage computes the derivatives (Iy, Iyy) for pixel (i, j) and the derivatives (Ix, Ixx) for pixel (j, i) of the same image. This process is summarized by the following pseudo-code:

```
WHILE TRUE
 SEQ
   WHILE (Producer_1_buffer is empty)OR
         (Producer_2_buffer is empty)
     SKIP
  Compute the resulting derivatives
  (Ix,Iy,Ixx,Iyy)
```

4 Conclusion

In this paper, a parallel algorithm for edge detection based on Infinite Impulse Response filters is presented. In particular, a parallel version of the Infinite size Symmetric Exponential Filter (ISEF) that is an optimal, computationally efficient, smoothing filter is developed. The proposed algorithm uses efficiently all types of potential parallelism inherent in the low level processing such as spatial and temporal parallelism and systolism. The proposed algorithm can be easily used to support any IIR filters. Although this paper deals with 2-D edge detection, the proposed algorithm could be adapted to 3-D edge detection such as Castan's and Deriche 's 3D-filters. A performance evaluation study of the proposed algorithm is currently under investigation.

References:

[1] R. Haralick & L. Watson, A facet model for image data*, CGIP*, Vol. 15, 1981, pp.113-129.

[2] M. Huckel, An operator which locates edges in digitized pictures, *JACM*, Vol. 18, 1971, pp.113-125.

[3] J. Prewit, Object enhancement and extraction, *In: B. Lipkin and A. Rosenfeld, eds., Picture processing and psychopictories,* New York*,1970,* pp.75-149.

[4] T. Poggio, V. Torre & C. Koch, Computational vision and regularization theory. *Nature*, 1985, pp.314-319.

[5] V. Torre & T. Poggio, On edge detection*, IEEE Trans. on PAMI*, 8 (2), 1986, pp.147-163.

[6] R. Deriche, Fast Algorithms for low-level vision, *IEEE Transactions on PAMI*, 12 (1), 1990, pp.78-87.

[7] S. Castan, J. Zhao & J. Shen, New edge detection methods based on exponential filter, *Proc. of 10th Int. Conf. on Pattern Recognition, Atlantic City, NJ (USA)*, Vol.1, 1990, pp.709-711.

[8] S. Castan, J. Zhao & J. Shen, Optimal filter for edge detection methods and results, *Proc. of First European Conference on Computer Vision, Antibes (France)*, 1990, pp.13-17.

[9] J. Shen & S. Castan, Towards the unification of band-limited derivative operators for edge detection, *Signal Processing*, Vol. 31, 1993, pp.103-119.

[10] L. Luque et al, Transputer based system software, *IEEE Workshop on Parallelism & Distributed Processing, Spain*, 1994, pp.536-543.

[11] X.T. Zhang & D.M. Zhao, A Parallel algorithm for detecting dominant points on multiple digital curves, *Pattern Recognition*, Vol. 30, No. 2, 1997, pp.239-244, Feb. 1997.

[12] P.D. Mackenzie & O.F. Stout. Ultra-fast expected time parallel algorithms*, J. Algorithms*, Vol. 26, 1998, pp.1-33.

[13] L. Prechelt, Efficient parallel execution of irregular recursive programs, *IEEE Trans. On Parallel and Distributed Systems,* Vol.13, No.1, 2002, pp.167-178.

[14] N.S. Sundar et al., Hybrid algorithms for complete exchange in 2D meshes, *IEEE Trans. On Parallel and Distributed Systems,* Vol.12, No.12, 2001, pp.1201-1218.

[15] Y.Ben-Asher and K. Haber, Parallel solutions of simple indexed recurrence equations, *IEEE Trans. On Parallel and Distributed Systems,* Vol.12, No.1, 2001, pp.22-37.

[16] A.M. Tyrrell, Parallelisation*. IEEE Workshop on Parallelism & Distributed Processing, Spain*, pp 30-31, 1994.