Parallel algorithm "conveyer processing with a minimal memory usage" of recursive method of scanning mask for primary image processing

Naiden Vasilev
Atanaska Bosakova-Ardenska

Abstract: *This paper analyses a parallel algorithm "conveyer processing" of recursive method of scanning mask and proposes a modification of it - "conveyer processing with a minimal memory usage". This algorithm removes a doubling of output information and decreases the number of operations compared with the algorithm "conveyer processing".*

Key words: *parallel algorithms, recursive method of scanning mask*

INTRODUCTION

The images processing is an important field of the information processing in the contemporary computer systems. Some of the fields where the image processing is essential are robotics, criminology, text recognition, recognition of the targets in military actions, medicine, mineralogy, cartography and etc. [1].

The method of the scanning mask is one of the methods for primary images processing. It represents a convolution of image with mask, e.g. it is founded on every-element visitation of the image matrix with a mask which includes the surround area of pixels of the treated element (or pixel) and multiplication of pixels under a mask with the coefficients of the mask. The masks are square matrixes with an odd number of the raster (3x3, 5x5, 7x7, 9x9) whose elements are coefficients. The central element of the mask coincides with the treated element of the image. The image is represented by one or more matrixes in dependency of if it is colour picture or not. Every element of the image matrix is represented by a value specifying the gradation of the pixel. If the image is black-and-white then the gradation defines the levels of the gray. The disposition of the pixels in the matrix is defined by the indexes in the matrix.

The value of the gradation of the treated element situated under the central element of the mask is replaced with a value received from the function $\mathbf{f(k_l,b_l)}$ where $\mathbf{k_l}$, $\mathbf{l}$=1,2,…,t – are the coefficients of the mask and $\mathbf{b_l}$ are the values of $\mathbf{b_{ij}}$, $\mathbf{l = 1,2,…,m, j=1,2,…,n}$ of the gradations of the pixels fallen under the mask. The values of **t** are usually 9, 25, 49 and 81. The type of the function $\mathbf{f\ (k_l,b_l)}$ specifies the filter. One of the most popular filters is '1/9' ('mean', 'neighbourhood averaging') where **t**=9 and all the coefficients have value 1. The function is:

$$f(k_l b_l) = \frac{1}{9}\sum_{l=1}^{9} k_l b_l = \frac{1}{9}\sum_{l=1}^{9} b_l \qquad (1)$$

Calculated by this way value becomes the new value of gradation of the processed element. During this treatment on every next step moving the mask **the new values are used which are obtained from the preceding steps.** For this processing the outlying rows and columns couldn't be filtered because they couldn't become central elements. The number of unprocessed rows and columns is equal to $t^{1/2}$-1.

AIMS

The tasks of images processing are characterized by operations "of the same kind" executed over large data massives. In some cases the requirements to the computer systems for the image processing are too high (processing moving objects for example). This impels the quest of algorithms insuring an acceleration of the processing. One effective method for reaching a high capacity in image processing is the usage of parallel algorithms (PA).

This work is an extension of works [2] and [3]. The aim there is a parallel conveyer algorithm to be proposed for primary image processing by the recursive method of scanning mask with minimal memory usage.

ANALYSIS OF THE RECURSIVE METHOD OF A SCANNING MASK

Let the image is sized mxn. We will designate the matrix of the initial picture with B and its elements with b_{ij}, I = 1,2,…,m; j=1,2,…,n. m and n are the number of rows and columns respectively in the picture. The consecutive treatment of pixels by the recursive method of scanning mask is performed by moving the mask from left to right and from above downwards. Hence:

- the value of b_{22} is received from the old values of gradation of the pixels falling under the mask;
- when calculating the values b_{2j}, j=3,4,..,n-1 a new value takes part - b2,j-1;
- when calculating the values b_{i2}, i = 3,4,...,m-1 two new values take part - $b_{i-1,2}$ and $b_{i-1,3}$;
- when calculating the values $b_{i,n-1}$, i = 3,4,…,m-1 two new values take part - $b_{i-1,n-2}$ and $b_{i-1,n-1}$;
- when calculating the values of all other pixels four new values take part, as it is shown on fig. 1.

In the described scheme of calculation it is available "data flow dependency" [5], which is a restriction when on research for parallel algorithms. Initial point for this work out is Parallel Concise Cortege (PCC) [3,5] of the examined method. On fig.2 is shown PCC of the method for image sized 8x12. The numbers in the matrix' elements for this image show the parallel step with a lower number in which the element can be calculated. For example, elements b_{27}, b_{35} and b_{43} can be calculated simultaneously in the sixth parallel step but can not be calculated in a parallel step with a **lower** number, because their calculation depends on the results been received in the fifth parallel step. This is valid for all the elements of the image. On fig.3 is shown another structure of the same PCC: it is shown the indexes of elements which are calculated on every parallel step.

$b_{i-1,j-1}$	$b_{i-1,j}$	$b_{i-1,j+1}$
$b_{i,j-1}$	$b_{i,j}$	$b_{i,j+1}$
$b_{i+1,j-1}$	$b_{i+1,j}$	$b_{i+1,j+1}$

Fig.1 The new values taking part in the calculation

If the elements of the image are treated by PCC then after the elements of the first row have been processed, in the second row there are untreated two elements. After the elements of the second row have been processed - in the third one there are again two untreated elements and etc. (fig.2). Hence, the number of the steps of PCC of the recursive method of scanning mask is:

$$s = (n - 2) + 2(m - 3) = n + 2m - 8 \quad (2)$$

For PCC on fig.3 and fig.4 the number of step is: s = 12 + 2x8 – 8 = 20.

	1	2	3	4	5	6	7	8	9	10	11	12
1												
2		1	2	3	4	5	6	7	8	9	10	
3		3	4	5	6	7	8	9	10	11	12	
4		5	6	7	8	9	10	11	12	13	14	
5		7	8	9	10	11	12	13	14	15	16	
6		9	10	11	12	13	14	15	16	17	18	
7		11	12	13	14	15	16	17	18	19	20	
8												

Fig. 2 PCC for image sized 8x12

Step 1	2,2				
Step 2	2,3				
Step 3	3,2	2,4			
Step 4	3,3	2,5			
Step 5	4,2	3,4	2,6		
Step 6	4,3	3,5	2,7		
Step 7	5,2	4,4	3,6	2,8	
Step 8	5,3	4,5	3,7	2,9	
Step 9	6,2	5,4	4,6	3,8	2,10
Step 10	6,3	5,5	4,7	3,9	2,11
Step 11	7,2	6,4	5,6	4,8	3,10
Step 12	7,3	6,5	5,7	4,9	3,11
Step 13		7,4	6,6	5,8	4,10
Step 14		7,5	6,7	5,9	4,11
Step 15			7,6	6,8	5,10
Step 16			7,7	6,9	5,11
Step 17				7,8	6,10
Step 18				7,9	6,11
Step 19					7,10
Step 20					7,11

Fig. 3 PCC with the indeces of elements on every parallel step

We will pay attention of that some of partial sums, when finding the value of a pixel, could be used for finding the values of other pixels also. These sums could be used for reducing of the number of operation in PA.

ANALYSIS OF THE PARALLEL ALGORITHM "CONVEYER PROCESSING" [3]

The proposed algorithm [3] distinguishes with simplicity and clarity. By this algorithm every processor processes consequently the elements of *p* columns from left to right by rows. (When *p*=2, the number of parallel steps is minimal and is equal to the number of the steps in PCC.)

The number of processors is k=(n-2)/p. We consider that ***k*** is an integer. Every processor contains the elements of **p** processed neighbouring columns and the elements of the two columns – one column before the p columns and one after. Every processor begins to work just after the previous processor has calculated the new values of elements of its first row. On every odd parallel step when p=2 every processor executes 9 operations and on every even step – 7 operations. In the common case the number of executed operations is 9 for every first processed column and for every next column are 7. The processors exchange the new values of elements necessary for the right executing of process. When p=2, on every parallel step the new values of elements are transferred to the neighbour processor on the right side, and on every even step – to the processor on the left side. In the common case when p = 3,4… etc. the new value of every processed element of the first column of processor P_i (i = 2,3,… k) is transferred to processor P_{i-1}, but the new value of every processed element of the last column of processor P_i (i = 1, 2,3,…k-1) is transferred to P_{i+1}.

In the considered algorithm we have a doubling of information. The number of doubled columns is **d=2k-2**. When p=2 then **d=n-4**. In other words only 4 columns of the image would not be doubled. If **the image consists of 1024 columns then 1020 of them would be doubled** (e.g. they would be present in two processors)!

Besides this the processors exchange only new values of image elements. It occurs a question is it possible some of the found partial sums to be used for finding other new

values of image elements. A reducing of number of operations is expected in parallel algorithm.

We will propose a parallel algorithm of recursive method of scanning mask where there is not a doubling of output information and where the processors exchange partial sums for reducing the number of operations, but not new values of pixels.

PARALLEL ALGORITHM "CONVEYER PROCESSING WITH A MINIMAL MEMORY USAGE"

The proposed algorithm is as shown on fig. 5 for p=2 and image sized 6x8. The number of processor is k=3. Every processor calculates the new values of elements of 2 columns (fig. 4). Processor P_1 calculates the new value of columns 2 and 3, P_2 processor – columns 4 and 5 and P_3 – columns 6 and 7. On fig.5 the elements are indicated with their indices. Besides this the operation dividing is not shown. Processor P_1 stores the elements of columns 1, 2 and 3, P_2 processor – 4 and 5, P_3 – 6, 7, and 8 column.

In the common case the distribution of output data to local processor memories is:

- in processor P_i i=1,2,3,…,k are saved the values of elements of i-number p-columns of the image. The counting of the columns starts from 2.
- in processor P_1 is saved the first column of the image.
- in processor P_k is saved the last column of the image;

	P1	P2	P3
Step 1	2,2		
Step 2	2,3		
Step 3	3,2	2,4	
Step 4	3,3	2,5	
Step 5	4,2	3,4	2,6
Step 6	4,3	3,5	2,7
Step 7	5,2	4,4	3,6
Step 8	5,3	4,5	3,7
Step 9		5,4	4,6
Step 10		5,5	4,7
Step 11			5,6
Step 12			5,7

Fig.4 A distribution of elements of image columns to processors when p=2

If an element is designated in gray background (fig.5) this means that the element participates with its new value.

The indication rectangle including three elements is for a partial sum. A rectangle starting an arrow is the place where this partial sum is calculated. A rectangle where an arrow ends is the place where this sum is used.

The indication '^' over an element in a rectangle shows that this partial sum should be re-calculated with the new value of indicated element.

The indication rectangle starting an arrow with a point shows that the designated partial sum should be found earlier, in the parallel step specified two steps earlier. This is necessary because this partial sum is needed in an earlier stage of calculating process.

The exchange interactions among processors are shown on fig. 5 where it could be seen that when p=2, on every odd step of PA every working processors transfers a partial sum to its neighbour on the left side (except the first one), and on an even step the sum is transferred to the right neighbouring processor (except the last one).

On the left side, on every step of PA is shown the number of operations for every processor. This number of operations is a sum of two digits - the first one indicates the number of operation for finding the new value of an element, the second one indicates the

number of operations for finding of the partial sum which should be transferred to another processor.

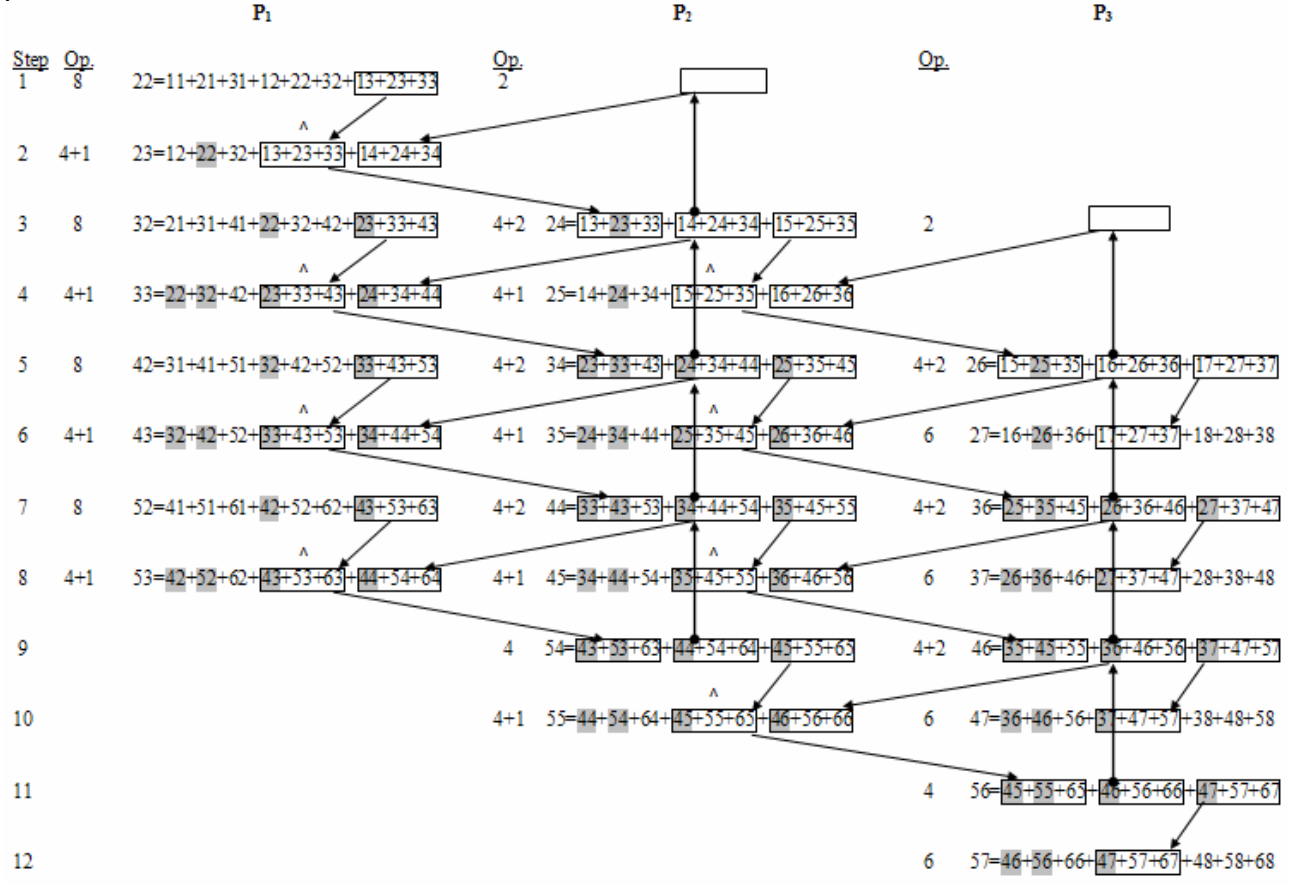

Fig.5 Parallel algorithm "Conveyer processing with a minimal memory usage" (for image sized 6X8, p=2, mask sized 3X3)

The number of operations of processor P_1 is 52=48+4. (Except the operation dividing.) In the common case it is 8(m-2)/2+5(m-2)/2=13(m-2)/2.

The number of operations of processor P_2 is 44=32+12. In the common case it is 6(m-2)/2+5(m-2)/2=11(m-2)/2. This formula is valid also when the number of intermediate processors is more than one.

The number of operations of processor P_3 is 48=40+8. In the common case it is 12(m-2)/2.

It is obvious that the processors of parallel architecture are not equally loaded. Processor P_1 has 13/11 loading compared with the intermediate processors' loading. An equalization of loading could be reached if processor P_1 calculates previously **all partial sums of the neighbour triples in the first column**. In this case the loading of processor P_1 is equal to the loading of intermediate processors. The processor P_3 can achieve the same loading if it calculates previously **all partial sums of the neighbour couples in the last column** of the image.

If the loading of all processors is equal and if we don't admit the moving of the calculation of partial sums designated with points on fig.5 two steps earlier, then the parameters of the proposed PA are the same like on PA "conveyer processing" [3].

The acceleration reached by the proposed algorithm in comparison with the serial one is approximately equal to the ratio of the number of serial steps to the number of parallel steps.

Let's remove the restriction n-2 to be divisible on k (without reminder). Let the quotient is *p* and the remainder is *d*. A good distribution is **d** processors to process with

p+1 columns and **k-d** processors to process with **p** columns. In other words there will be a difference in the loading of the processors. During the exchange interaction between processors with a different number of columns, these ones which have a column less should complete one empty operation on every row except the first row which is needed to receive the necessary information. The task of exchange interaction will be object of further exploration works.

CONCLUSION

A parallel algorithm „conveyer processing with a minimal memory usage" is proposed using the recursive method of scanning mask. When comparing with PA "conveyer processing", in this case **we don't have a doubling of the output information** e.g. it is stored once time and **the quantity of exchanged information among processors of parallel architecture is the same**. Besides this when the loading of processors is equal then the number of operation in every processor of the proposed PA is reduced in comparison with this on PA "conveyer processing". The number of operation for every first processed column is 6 and 8 respectively and for every next column – 5 and 6 respectively. All above is shown in table 1.

PA	Doubled information	Number of operations on a parallel step	
		For first column	For every next column
„Conveyer processing"	2(n-2)/p - 2	8	6
„Conveyer processing with a minimal memory usage"	0	6	5

Table 1 Comparative parameters for algorithms "Conveyer processing" and "Conveyer processing with a minimal memory usage"

The future directions for development are related with a search of other parallel algorithms of the considered problem and with a search of parallel algorithms for other similar tasks as well

REFERENCES

[1] Gonzalez, "Digital image processing" second edition, 1987г.

[2] A. Bosakova-Ardenska, N. Vasilev, Parallel algorithms of the scanning mask method for primary images processing, CompSysTech'04, Rousse, Bulgaria

[3] Vasilev N., Bosakova-Ardenska A., One Parallel Algorithm of the Recursive Method of Scanning Mask for Primary Images Processing, Computer Scince'04, Sofia, Bulgaria

[4] Э.В.Евреинов, Ю.Г.Косарев, "Однородные универсальные вычислительные системы высокой производительности", Издательство "Наука" Новосибирск, 1966г.

[5] Seyed H. Roosta, Parallel Processing and Parallel Algorithms: theory and computation, 2000г.

[6] Vasilev N., Main Principles for Searching and Creating Parallel Algorithms, Information Technologies and Control, 2004, 2.

ABOUT THE AUTHORS

Assoc.Prof. Naiden Vaslilev, PhD, Department of Computer Systems, University of Sofia, branch Plovdiv, Phone: +359 32 659 705, E-mail: mnvasilev@yahoo.com.

PhD student Atanaska Bosakova-Ardenska, Department of Computer Systems, University of Sofia, branch Plovdiv, Phone: +359 32 659 704, E-mail: abosakova@yahoo.com.