Parallel algorithms of the scanning mask method for primary images processing

Atanaska Bosakova-Ardenska, Naiden Vasilev

Abstract: *This paper discusses the possibility of parallel images processing. It proposes several parallel algorithms for primary processing of grey and colour images by method of scanning mask. An evaluation of proposed algorithms has been made here.*

Key words: *parallel algorithm, primary images processing, method of the scanning mask*

INTRODUCTION

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

Before the images have been used, they are usually 'cleaned'. During the primary images processing three essential types of defects are cleaned:

1) *High-frequently 'grain' noice: those are the pixels, whose gradation is sharply different from neighbour pixels' gradation.*

2) *Presence of spots: that is fields from a given object in which the gradation sharply differs from the common object one, because of the spots of light or shadows entered during the photos-taking.*

3) *Object spread: this type of defects occurs if a moving object is photoed by low speed of registration or good focus is missing on the optical systems of the registration cameras.*

The defects into images can be removed by treatment of the specter (appropriate for the case) - distribution of the pixels by frequency on the picture, or of its histogram - distribution of pixels by gradation.

Primary processing could be:

1) *Anisotrop: the processing is done on the base of input data*;

2) *Recursive: where on every next step is used the result from the previous step.* The method of the scanning mask is one of the methods for primary images processing [2]. It is founded on every element visitation of the matrix with a mask that includes the surround area of pixels of the treated element (pixel). The masks are square matrixes with an odd number of the raster (3X3, 5X5, 7X7, 9X9). The central element of the mask is identical with the treated element of the image. Every element of the mask contains a coefficient (fig.1). The image is represented by one or more matrixes dependent of if it is colour picture or not. Every element of the matrix of the picture represents a value specifying the gradation of the pixel. If the picture is black-and-white then the gradation defines the hue of the grey. The disposition of the pixels in the matrix is defined of the indexes in the matrix.

k_1	k_4	k_7
k_2	k_5	k_8
k_3	k_6	k_9

Fig.1 Scanning mask 3x3

The value of the gradation of the treated element situated under the central one of the mask is replaced with the value of the function $\mathbf{f(k_l,b_l)}$, where $\mathbf{k_l}$, **l**=1,2,…,t – are the coefficients of the mask and $\mathbf{b_l}$ are the values of $\mathbf{b_{ij}}$, **i**=0,1,2,…,**n-1, j**=0,1,2,…,**m-1** of the gradations of the pixels gotten from the matrix of the picture fallen under the mask. The values of **t** are usually 9, 25, 49 and 81. The mode of the function $\mathbf{f\ (k_l,b_l)}$ specifies the filter. One of the most popular filters is 1/9', where **t**=9 and all the coefficients have cost 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 mean cost becomes the new value of the gradation of the new element. Throughout this treatment the outlying rows and columnes aren't filtred because they couldn't be central elements.

AIMS

The problems of image processing are characterized with similar operations over large data massives. In some cases of images processing the requirements of the computer systems are too high (for example – when treating moving objects). This enforces the quest of algorithms allowing acceleration in processing. One effective method reaching high efficiency in images processing is the usage of parallel algorithms. The nature of the primary images processing is conductive to the discovering and the usage of parallel algorithms.

The aim of this working out is some parallel algorithms to be proposed in the field of primary images processing by anisotrop method of the scanning mask.

PARALLEL ALGORITHMS

Lets the picture be whit dimensions **n**x**m** pixels. We will build the parallel extracted solution [3] of this problem whit the images processing by the scanning mask method (3X3). In other words we will indicate on every step of the solution when is the maximal number of operations which could be executed at the same time. In this case the problem can be solved in 5 steps. The number of simultaneously executed operations by step is:

- first step: the operations summing up – 4 for every pixel - **4(n-2)(m-2)**;
- second step: operations summing up - 2 for every pixel - **2(n-2)(m-2)**;
- third step: operations summing up – 1 for every pixel - **(n-2)(m-2)**;
- fourth step: operations summing up – 1 for every pixel - **(n-2)(m-2)**;
- fifth step: operations dividing – 1 for every pixel - **(n-2)(m-2)**.

This is the solution with minimal number of parallel steps where the number of the processor elements must be **4(n-2)(m-2)**, which is a really high number of elements.

Equal loading-up of the processor elements could be reached when their number is **(n-2)(m-2)**. In this case the parallel algorithm will be executed for 9 steps and the treatment of all the pixels of the picture except outlying ones is executed simultaneously. In other words, the number of the processor elements in this case is equal to the elements (pixels) to which we calculate new values.

In both cases the requirement of estimated number of processor elements is hardly to be contented regarding the sizes (in pixels) of the images nowadays. We will propose parallel algorithms which requirements could be contented and what with we could achieve significant acceleration in the working out of the examined problem.

Let's accept that we dispose with an original object represented by a massif **A** and whit a treated picture in massif **B**, so during the processing we will not destroy the original image.

1. Solution: 1 row in a processor;

Every processor $\mathbf{p_i}$, **i**=1,2,…,**n**-2 will process elements of row **i**. The zero number and the (**n**-1) row and the zero number and the (**m**-1) column do not change. Then the parallel algorithm would look like:

```
For i = 1 to n – 2 do in parallel
    Begin
        B[i,0]:=A[i,0]; B[i,m-1]:=A[i,m-1];
```

```
            For j = 1 to m – 2 do
                  Begin
                    B[i,j]:= 0;
                    For y = -1 to 1 do
                        For x = -1 to 1 do
                          B[i,j]:= B[i,j]+ A[i+x][j+y];
                    B[i,j]:= B[i,j] /9;
                  End;
        End;
     For j = 0 to m – 1 do
        Parbegin
            B[0,j]:=A[0,j]; B[n-1,j]:=A[n-1,j];
        Parend;
```

It is used pseudo-language similar to Pascal. The expression '**do in parallel**' is used to mark the dividing into parallel ways. This means that the cycles for the different values of **i** can be executed at the same time. The operators between the keywords **parbegin** and **parend** could also been accomplished simultaneously (in parallel way). In this case **n-2** processor elements (or processors) are needed. For the work of every processor p_i data of three rows of the image are needed: (**i-1**), **i** and (**i+1)**. The executing of the two operators in the last cycle could be assigned to the processors p_1 and p_{n-2}.

During the execution of the shown algorithm for parallel architecture with shared memory, delays could appear during reading and writing in the memory. In the parallel architecture with shared memory every processor must contain 3 rows of the picture values, i.e. the values of the rows from the 2-nd to (**n**-3)-th will be written in the memories of three processors, the 1-st one and (**n**-2)-nd – in two processors, and the zero number and (**n**-1) – in one processor. Every processor completes **8(m-2)** operations summing and **(m-2)** operations dividing and **2** operations of transfer. The processors p_1 and p_{n-2} are completing also **m** operations of transfer each.

When a colour image should be treated it would be three massives instead of one (for the three basic colours RGB). In this case **3(n-2)** processors can process one line each or **(n-2)** processors can treat three pixels each (for every of the colours).

2 Solution: one row in a processor using the calculated partial sums.

The processing described above can be accelerated if the three partial sums are stored **S1** (b1+b2+b3), **S2** (b4+b5+b6) and **Snext** (b7+b8+b9) (fig.1) during the finding of the gradation of chosen element. On fig. 2 are marked the partial sums of the element <1,2>. The partial sums **S2** and **Snext** are used in the sum for finding the value of the next element in the row (<1,3>), but for this element they should be considered as the sums **S1** and **S2** appropriately. Therefore the value of the sum of the element can be found by 4 summing-up (instead of 8). In other words the idea of acceleration is based on the fact that when the mask is creeping to the right on the row, some part of the sum for the next element has been already calculated.

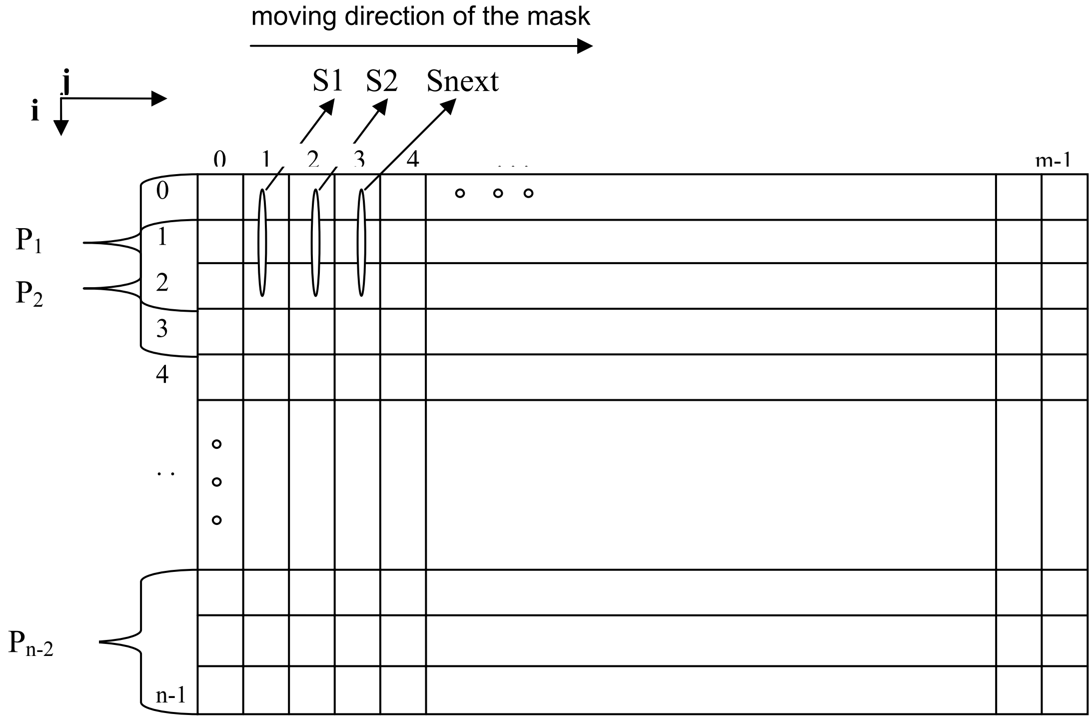

Fig.2 Formation of partial sums when moving on a row

As it stands in the previous solution every processor $\mathbf{p_i}$, **i**=1, 2,…, **n**-2 will process with the elements of one row – **i**. The zero and the (**n**-1) number row, and the zero and the (**m**-1) number column do not change. For the work of every processor $\mathbf{p_i}$ are needed the data of three rows of the picture: (**i-1**), **i** and (**i+1)** number of a row. The processors $\mathbf{p_1}$ and $\mathbf{p_{n-2}}$ can be entrusted with the executing of the two operators in the last cycle.

Then the parallel algorithm will be:

```
For i = 1 to n-2 do in parallel
   Begin
       B[i,0]:=A[i,0]; B[i,m-1]:=A[i,m-1];
       S1[i]:= 0;
       S2[i]:= 0;
       Snext[i]:= 0;
         For x = -1 to 1 do
             Begin
               S1[i]:= S1[i] + A[i+x][0];
               S2[i]:= S2[i] + A[i+x][1];
               Snext[i]:= Snext[i] + A[i+x][2];
               B[i,1] := (S1[i] + S2[i] + Snext[i]) / 9;
             End;
       For j= 2 to m-2 do
         Begin
             S1[i] := S2[i];
             S2[i] := Snext[i];
             For x = -1 to 1 do
               Snext[i] := A[i+x][j+1];
             B[i,j] := (S1[i] + S2[i] + Snext[i]) / 9;
```

```
        End;
    End;
For j = 0 to m – 1 do
    Parbegin
        B[0,j]:=A[0,j]; B[n-1,j]:=A[n-1,j];
    Parend;
```

During the simultaneous work every processor $\mathbf{p_i}$ complete **8** summing-up for the finding the value of the element <i,1>, **4(m-3)** summing-up for the finding the elements <i,j>, j=2,3,…,m-2, **(m-2)** dividings and 2 transfers. When **m** has higher values the number of the summing-up will be almost twice smaller than its number if partial values are not used.

If the object is represented by 100 pixels on the **j** direction of the picture, then the count of the summing-up will be: 8 + 97.4 = 396 (for 1 processor element) when the partial sums are used and 98.8 = 784 if they are not used.

The acceleration in this case is 1 - 396/784 = 1 - 0,505 = 0,495, i.e. 49,5% in comparison with the case when the partial sums are not used.

The proposed 2 algorithms can be generalized for **k**, **k**=2,3,…,**(n-2)/2** consecutive rows treated by one processor. In this case the number of the operations in every processor element will increase **k** times. The count of the processor elements is **(n-2)/k**, i.e. when increasing **k** their number is lowering. (Let (n-2)/k is integer. This assumption does not decrease the consumpsion of the reasons.) Thus the high requirements of count of processor elements could be decreased.

We will consider that in this solution, one row in processor of parallel architecture with shared memory, the written input information in the system is **3(n-2)** rows. In other words the image for processing is represented by almost three times more information. This is so because of the doubling of rows in the processors (see above). In the solution **k** rows in a processor, the written input information is **(1+2/k)(n-2)**. It could be seen that if **k** is higher, the quantity of the doubled information is less.

*3.Solution: **k** neighbour rows in a processor with exchange of partial sums.*

Every processor $\mathbf{p_q}$, **q**=1,2,…,(**n**-2)**k** treat the elements of the rows (**q**-1)**k**+1, (**q**-1)**k**+2,...,**qk**. Let (n-2)/**k** is integer. In other words every processor $\mathbf{p_q}$ processes the elements of **k** number of rows, thus the information in the processor elements is not doubled. The algorithm is as follows:

1. **j**=1. Every processor $\mathbf{p_q}$ calculates consequently the values of the gradation of the elements $\mathbf{b_{i1}}$, **i=(q-1)k+2, (q-1)k+3,...,qk-1**, completing **8** summing-up according to the indicated in formule 2 order for finding of the new value $\mathbf{b_{i1}}$ of every element:

$$b_{ij} = \frac{1}{9}(S_{q1} + S_{q2} + S_{q3}) \qquad (2)$$

where S_{q1}=b1+b4+b7, S_{q2}=b2+b5+b8, S_{q3}=b3+b6+b9 (fig.1).

The partial sums S_{q1} and S_{q3} are needed for processors $\mathbf{p_{q-1}}$ and $\mathbf{p_{q+1}}$ for the finding of gradation of the elements $\mathbf{b_{(q-1)k,1}}$ and $\mathbf{b_{qk+1,1}}$.

Partial sums $S_{q+1,1}$ and $S_{q-1,3}$ are needed for processor $\mathbf{p_q}$ for finding of the gradation of elements $b_{(q-1)k+1,1}$ and $b_{qk,1}$.

2. Every processor $\mathbf{p_q}$ exept $\mathbf{p_1}$ delivers its partial sum S_{q1} to $\mathbf{p_{q-1}}$ and receives the partial sum $S_{q+1,1}$, after that the value of $\mathbf{b_{qk,1}}$ is counted.

3. Every processor $\mathbf{p_q}$ exept $\mathbf{p_{(n-2)k}}$ delivers its partial sum S_{q3} to $\mathbf{p_{q+1}}$ and receives partial sum $S_{q-1,3}$, after that the value of $\mathbf{b_{(q-1)k+1,1}}$ is counted

4. **j**=**j**+1. If **j** is smaller than **m-1** it could pass towards p.1. If **j** is equal to **m**-1 – then end.

The case **k**=3 is shown on fig.3.

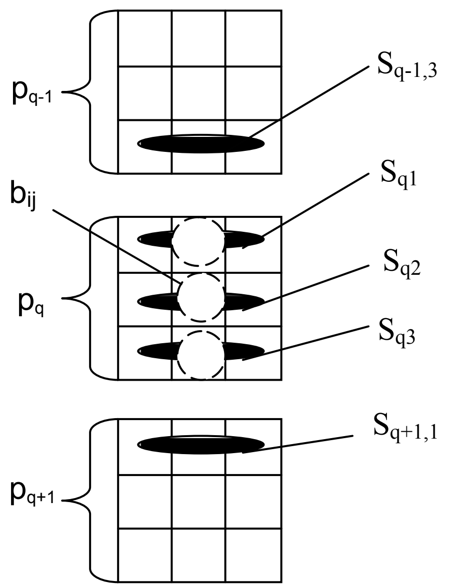

Fig.3. Solution: 3 rows in processor with exchange of partial sums

We will pay attention to that the changing of **k** does not influence on the number of sent and received partial sums of every processor. They are always 2 for processors $\mathbf{p_1}$ and $\mathbf{p_{(n-2)/k}}$ - one sent sum and one received. For the other processors they are always 4: two sent sums and two received.

When **k** increases, common count of the exchanged partial sums will decrease. In other words the traffic in the communication network decreases. That can be important factor when choosing suitable algorithm for the solution of the problem [4].

Increasing the number of elements calculated on 1 step of the mask-moving from 1 parallel branch hasn't influenced over the number of the exchanged partial sums for this branch but significantly lower the loading up of the communication network at all.

For example: if the image has 1000 pixels then during the distribution of three elements for each processor, we will need 998/3≈333 processor elements which will exchange 331*2+2*1=664 partial sums for 1 step of the algorithm. By solution 10 rows in processor will be needed: 998/10≈100 processors; that sent and received 98*2+2*1=198 partial sums for finding the new values for 10 elements. In this case the number of the exchanged sums decrease ≈3,35 times. In other words **k** is back proportionately on the common count of exchanged sums.

In this algorithm with 12 summing-up are found the new values of three elements. In parallel architecture with shared memory it doesn't double the input information, but it takes place a transfer of partial sums among the processors (two sent and two received by a processor). In the previous algorithm after the first element has been found (8 summing-up), also 12 summing are needed for the finding the values of three more elements. Part of the input information however should be doubled but transfer of information among processors is missing.

CONCLUSION

The proposed methods for acceleration of calculations in primary anisotrop image processing by a scanning mask can be used in the treatment of large images. For this aim there are needed computer systems with parallel architectures. In this way significant acceleration could be achieved when resolving this problem. Also this acceleration can be calculated previously.

LITERATURE

[1] Т.Павлидис, "Алгоритмы машинной графики и обработки изображений", Москва, 1986

[2] P.Pavlova, Lection course of "Computer vision", Plovdiv 2002

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

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

ABOUT THE AUTHORS

Ms. Atanaska Bosakova-Ardenska, department "Computer systems", Technical University Sofia, branch Plovdiv, tel.:032 659 704, E-mail: abosakova@yahoo.com.

Assoc. Prof Ph.D. Naiden Vasilev, department "Computer systems", Technical University Sofia, branch Plovdiv, tel.: 032 659 705, E-mail: mnvasilev@yahoo.com.