Bull. Nov. Comp. Center, Comp. Science, 22 (2005), 79–89

PLVIP — a parallel image processing library based on the vertical processing principle*

E.V. Rusin

Abstract. The experimental library PLVIP for parallel image processing, elaborated and implemented in the Image Processing Laboratory of the ICM&MG SB RAS is described. The library is built on the vertical processing principle and is installed on the multiprocessor computers of the Siberian Supercomputer Center, MVS-1000/M and RM600-E30. The basic characteristics of this library (supported data formats, computational process organization, implemented subprograms) and an example of its application are considered.

1. Introduction

A bulk of remote sensing data (information flows up to 128 Mbps, a single image size about 1 Gb, a daily content of receive data up to 60 Gb) and, also, the need in its real-time interpretation (for example, the forest fires and the flood monitoring problems) require high-performance computers involved in processing. Today it is obvious that, by increasing the computer clock rate alone, one cannot reach necessary performance, and parallel processing is the only means to obtain results in the time required.

The existing image processing systems belong, as a rule, to one of the three classes: they are either multipurpose and highly optimized libraries for personal computers and workstations, such as Intel IPP [1], or open source library OpenCV [2]; or programs for single problems solution on multiprocessor computers; or more specialized systems based on custom hardware. Unlike some other fields of informatics that require a large amount of computations (linear algebra problems solution, high-performance data bases building, etc.), there do not exist basic image processing systems on multiprocessor general-purpose computers today.

The present work is dedicated to describing the experimental library for parallel image processing PLVIP, elaborated and implemented in the Image Processing Laboratory of the ICM&MG SB RAS. As algorithmic basis, the principle of vertical processing was chosen, which provides effective processing of arrays of short data, for example, halftone images with low color depth. The library is installed on the multiprocessor computers of the Siberian Supercomputer Center, MVS-1000/M and RM600-E30.

*Supported by Russian Academy of Sciences, Integration project No. 13.09 and Russian Foundation for Basic Research under Grant 05-07-90057.

In this paper, after a brief description of the vertical processing method, the basic principles of building the library PLVIP are considered: supported data formats, computational process organization, and implemented subprograms. Application of the software developed is illustrated by an example of solution of the classic digital cartography problem, an elevation map recovery from a given set of contours. In conclusion, the author's considerations about future work direction are stated.

2. The vertical processing

In the 1970s, for supporting solutions of problems permitting mass information processing, including image processing ones, specialized fine-grained SIMD computers were created, which consisted of a very large number (up to one hundred thousands) synchronously operating one-bit processor elements with own memories and an interconnection network connecting them [3]. The characteristic feature of such computers was a *vertical*, or word-parallel bit-serial, approach to data processing (vertical processing, VP) [4].

The VP principle is schematically shown in Figure 1, which represents an example of processing of the same data array by the conventional and the "vertical" techniques. Unlike the conventional processing, when the array elements are in turn placed in a processor, and a single element is processed in the processor as a whole, under the VP, the same-named, i.e., of the same seniority, bits of multiple elements are placed in the processor together.

A set of values of same-named bits of all pixels of an image is called a bit plane of the image. Bit planes are exactly processed units of the vertical

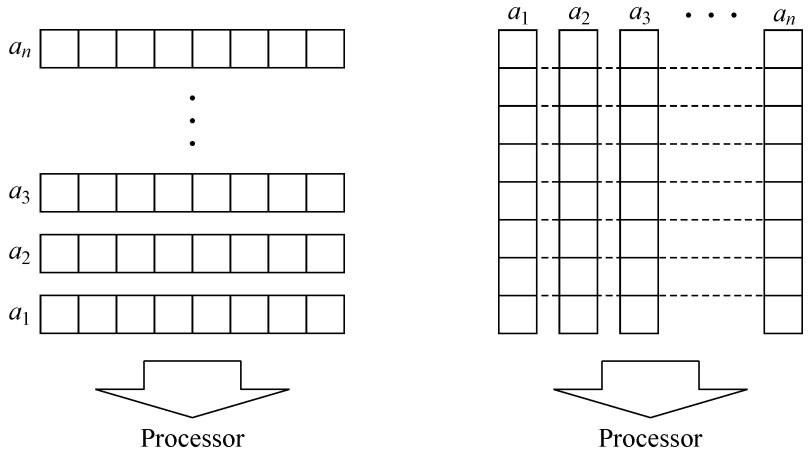

Figure 1. Conventional (on the left) and the vertical (on the right) approaches to data processing

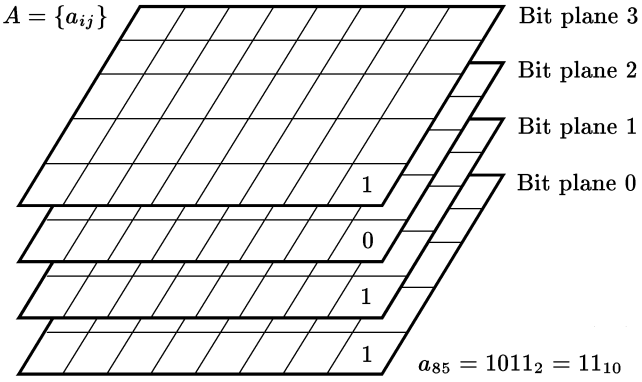

Figure 2. 16-color image A consists of four bit planes. The value of the pixel $(8, 5)$ of A is composed of the values of the corresponding bits of A planes

image processing (VIP); the value of a pixel is a composition of values of the corresponding bits of image planes (Figure 2).

The algorithmic basis of the VIP is formed of the operations with bit planes:

- bitwise logical operations between bit planes;
- shifts of a bit plane to horizontal (X-shifts) and vertical (Y-shifts) directions;
- calculation of a bit plane "mass", i.e. the number of unit bits in it;
- check of presence of at least one unit (or zero) bit in a bit plane and of equality of two bit planes.

The data model used in the VP has not found its application out of specially designed processors; in particular, the questions of implementation of VIP in general-purpose computers are unexplored now. Meanwhile, there exist the following reasons to study these questions:

1. Performance of bit-serial computations grows with a decrease in the digit capacity of data under processing; this allows making more efficient the processing of arrays of short data, for example, halftone images with a low color depth.
2. For the VP, there are no limitations on the digit capacity of data processed; this allows one to easily carry out calculations with an arbitrary accuracy.
3. The algorithmic basis of the VIP can be effectively implemented on general-purpose computers by bitwise logic operations between machine words and by shifts of ones; the present-day 64-bit microprocessors allow one to process 64 pixels of an image in parallel.
4. The intrinsic parallelism of the VP allows using multiprocessor computers of a conventional architecture for the VIP implementation.

The considerations given became the reasons for creating an experimental library for parallel image processing PLVIP (a Parallel Library for the Vertical Image Processing) built on the basis of the VP principle on the widespread conventional architecture MIMD-computers.

3. The PLVIP library

The library PLVIP is implemented as a set of C-subprograms. The programming language C and the parallel programming interface MPI were chosen as implementation tools; that provides both the possibility of low-level manipulation with bit planes and the portability of the source code of the library to most of the present-day multiprocessor architectures.

3.1. Data formats. The key problem in introduction the idea of the VP to conventional computers is in the development of a bit plane format. Impossibility to realize the full two-dimensional topology on a linear address space of conventional computer resulted in compromise, which is similar to the way the VGA video adapter maps video memory onto the main memory in some graphics modes: a bit plane of an $M \times N$ image is represented by a set of N-bit strings formed of sequentially kept machine words. The number of words in a string is equal to M/M_0, where M_0 is a machine word length (suppose, M is divisible by M_0). To support the neighborhood relation by the machine words shifts, the correspondence between the string bits and the image pixels is defined as follows: the i-th (from leftmost) bit of the j-th (from the lower address) word of the k-th string of the bit plane corresponds to a pixel with the coordinates $(jM_0 + i, k)$ (Figure 3).

On this basis, the formats of halftone images are built, integer-valued and fixed-point-valued, as sets of bit planes. The color depth of fixed-point-valued images is an even number, and the binary point divides the bit representation of a pixel value in half. The color depths of halftone images are defined before the library compilation.

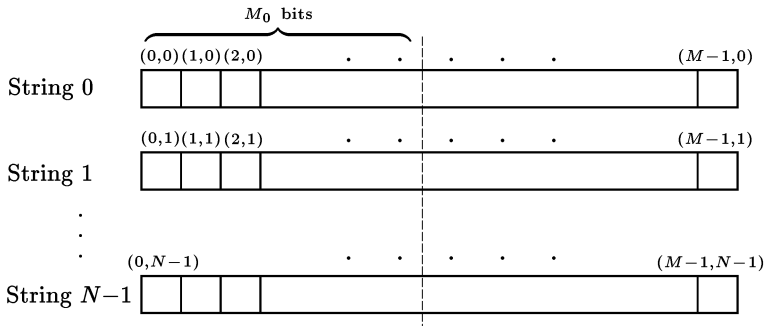

Figure 3. The bit plane representation

3.2. Computational process organization. As basis principle of calculations parallelization, the domain decomposition principle was chosen: bit planes (and, therefore, halftone images consisting of them) are cut into contiguous horizontal strips, whose number is equal to the one of operating processors; the strips are distributed among processors (Figure 4).

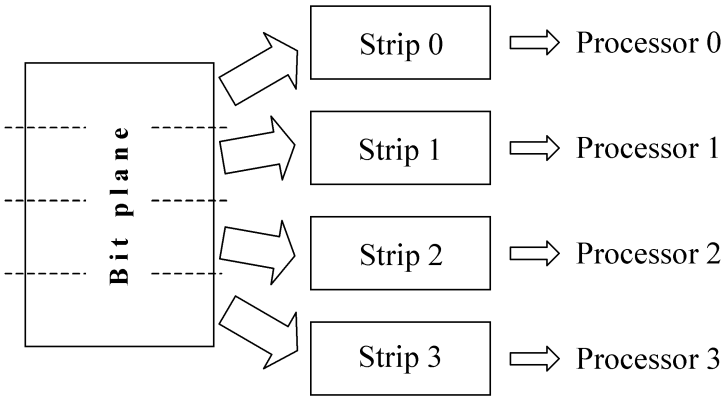

Figure 4

This principle provides effective parallelization of the logical operations between bit planes and the X-shifts of a plane, but Y-shifts, the mass computation, and the comparison of planes in their general form require the data transfer between processors under such a way of parallelization. Unlike bit planes comparison and mass calculations, when the intensity of interprocessor communications can be essentially weakened in practical cases by using in a single processor P the results of these operations in the strip corresponding to P, an intensive use of the bit planes Y-shifts by the algorithm makes one unsuitable for such a parallelization. Therefore, to provide alternative ways of parallelization, most operations are implemented in the PLVIP library both for cut into strips images and for the whole images in one processor.

As calculations topology, the "star" was chosen: all input/output operations and transformations of image formats are carried out by the one — "root" — processor; the other processors receive data from the root, processing them, and, if necessary, return the results to the root. The choice of such a topology is stipulated by the feature of the file system of the MVS-1000/M computer.

3.3. Subprograms of the library. A set of the library subprograms includes ones for:

- The library initialization/deinitialization.
- Images input/output in BMP and PCX formats.

- Distributing an image between processors and assembling an image on the root processor from strips.
- Basic operations for binary images (or bit planes): copying, bitwise logical operations, shifts, comparisons, and mass calculations.
- Basic operations for halftone images: copying, transformation from the vertical format to the traditional and backwards, setting/getting a pixel value, etc.
- Arithmetic calculations for halftone images: pixelwise summation, subtraction, multiplication, division; calculations of the sum of image pixels and of the inner product of two images.
- Halftone images processing: building histogram, profiles, level set of pixels values; thinning by the Zhang Suen method, and Euclidean Distance Transform (EDT) calculations.

Table 1 shows the speed-up values of execution of some subprograms of the library on 2, 4, 8, 16, and 24 processors as related to one-processor execution (the size of test images was 640×640 pixels, the calculations were carried out on MVS-1000/M).

Table 1

Subprogram	Number of processors				
	2	4	8	16	24
Pixelwise multiplication	1.6	4.4	10.5	18.0	24.4
Inner product	2.0	3.8	7.4	14.4	19.9
Building histogram	2.2	3.7	10.0	18.5	21.7
Thinning by Zhang Suen	1.7	3.6	6.1	8.8	10.3
Building level set of pixels values	1.7	3.3	7.6	10.0	20.0
EDT calculations	2.0	4.7	7.5	15.5	20.0

The exceeding, in some cases, operating processors number N by the speed-up value can be explained by the two factors:

1. When N grows, the volume of data a single processor manipulates with decreases, and, beginning with certain $N = N^*$, the data is entirely placed in the processor's cache the access to which being much faster than the one to the main memory.
2. In a number of algorithms, optimization was applied, which allows one to decrease the computations volume on a single processor, using some global properties of the strips being processed (the absence of any unit (or zero) bits in a bit plane strip, an equality of two bit plane strips, etc.). When N increases and, respectively, the size of a strip processed by a single processor decreases, the effectiveness of such an optimization grows.

4. Application of the library PLVIP: solution of the problem of elevation map recovery from a set of contours

4.1. The problem statement. The library PLVIP was used for solving the problem of elevation map recovery from a set of contours. The problem is considered in the following statement [5]:

1. Let the rectangular region B of the raster plane with square elements (pixels) be given; B represent the result of discretization of some region of the Earth's surface.
2. A set of B pixels in which the Earth's surface height value equals H is called a *contour of the level* H.
3. Let the set $C_1, C_2, \ldots, C_S$ of contours of the level $L_1, L_2, \ldots, L_S$, respectively, be given on B and $L_i < L_j$ when $i < j$.
4. A set of contours reflects the Earth's surface continuity, that is, there are no contours C_i and $C_j, j > i + 1$, that are not separated from one another by the contours $C_{i+1}, C_{i+2}, \ldots, C_{j-1}$.

Based on 1–4, it is necessary to approximate the value of the Earth's elevation in pixels of the set $U = B \setminus \bigcup_{i=1}^{S} C_i$.

It is shown in [6] that the fourth condition can be weakened and one can allow even the intersection of contours of different levels. However, the respective changes in the algorithm do not influence its parallelization, so all the reasonings below will be referred to the simpler statement 1–4.

4.2. Algorithm of solution. The considered algorithm of the problem solution is described in detail in [6]. Without going into details, let us note that the algorithm consists of the following steps:

1. Each contour C_i is divided into the non-intersecting subsets of pixels C_{ij}, $j = 1, \ldots, J_i$ (segments) such that two different segments of one contour C_i are separated from one another by the set $D_i = \bigcup_{k \neq i} C_k$.
2. For each segment C_{ij}, a region of dependence $\mathrm{dep}(C_{ij})$ is built, which consists of pixels not separated from C_{ij} by D_i (Figure 5); in $\mathrm{dep}(C_{ij})$, a field Dist_{ij} of Euclidean distances to C_{ij} and a field of the segment level Hight_{ij}, whose value is equal to L_i in every pixel, are built.
3. Four global fields are assembled from the built local ones: $\mathbf{Dist}_0$ and $\mathbf{Dist}_1$, which are compositions of the fields of distances to the segments of even- and odd-numbered contours, respectively, and $\mathbf{Hight}_0$ and $\mathbf{Hight}_1$, which are the similar compositions of the fields of segments levels (the composition operation is correct because the regions of dependence of either two segments of non-neighboring contours or two different segments of one contour do not intersect).

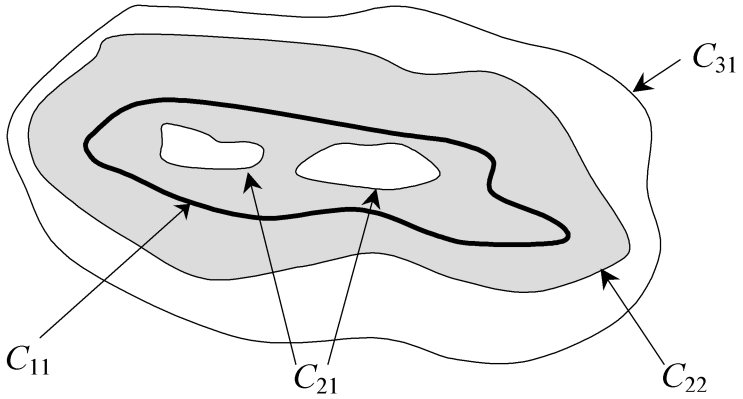

Figure 5. The segments of the contours C_1, C_2, and C_3; the grey region is $\mathrm{dep}(C_{11})$

4. The field of the height approximating values **H** is obtained as a result of an array arithmetic operation applied to the global fields built:

$$\mathbf{H} = \frac{\mathbf{Dist}_0\mathbf{Hight}_1 + \mathbf{Dist}_1\mathbf{Hight}_0}{\mathbf{Dist}_1 + \mathbf{Dist}_0}.$$

An example of the algorithm application is shown in Figure 6: the left picture is a set of 15 contours on the raster map of the size of 500×500 pixels (the black pixels against white background), the right picture is the resulted elevation map.

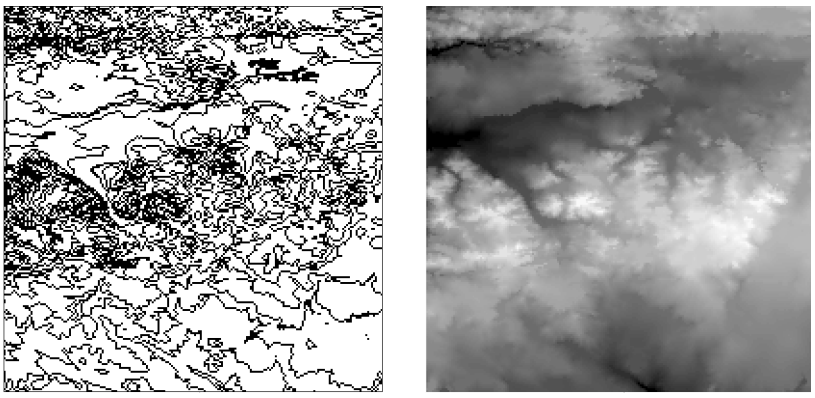

Figure 6. An example of the recovery algorithm application

4.3. Parallelization of the algorithm with the library PLVIP. The most time-consuming parts of the algorithm are the extraction of segments from contours and calculating the EDT for them (the final arithmetic operation takes a smaller part of the algorithm execution time and can be easily

parallelized by the domain decomposition). In this case, only the EDT calculation is well parallelized by the domain decomposition [7]. The operation of segment extraction and its region of dependence building consists of an iterative sequence of shifts of a contour image; moreover the criterion of termination of segment extraction is a predicate concerning some image as a whole, and this image is changed from iteration to iteration.

Consider three approaches to the algorithm parallelization implemented with the PLVIP library.

Algorithm A_1 — the basic parallelization way of the PLVIP library. Each image involved in calculations is cut into horizontal strips, which are then distributed among processors. Here an image Y-shift results in data transfer across the boundary between the neighboring strips, and the check of the condition of termination of segment extraction requires synchronization of the work of all operating processors. Next, for the parallel calculation of the distances field of a segment, each processor needs the whole image of the segment obtained, so it is necessary to assemble it on every processor. An obvious disadvantage of this approach is an intensive interprocessor interaction.

Algorithm A_2 — the parallelization by contours. Each processor processes (i.e., extracts segments, builds the regions of their dependence, and calculates distances and levels fields) its own subset of contours, a contour is processed as a whole by one processor. After that, the local fields of distances and levels are gathered on the root processor, where then the global fields $\mathbf{Dist}_{0(1)}$ and $\mathbf{Hight}_{0(1)}$ are composed. Next, the global fields, which are fixed-point-valued images, are cut into strips and distributed among processors for the final arithmetic operation carrying out. The advantage of this approach is an insignificant dependence of parallel processes, as synchronization and interprocessor interaction are required only when the data obtained are assembled. The disadvantage is the processors number limitation by the contours number and a possible non-uniform processors load due to the workload of a single processor depends on the number and complexity of contours it processes.

Algorithm A_3 — the parallelization by contours with workload equalization. The extraction of segments from each contour is carried out by one processor. Processors that have no contours or which have already processed their contours receive extracted segments on request and calculate the EDT for them. If there are no requests for the segment, it is processed by the processor have extracted it. The requests processing is carried out by the root processor, which does not take part in the calculation itself. This approach seems to be the most flexible of the three listed ones; there is no strong processes dependence as in A_1, and more uniform processors load is reached than in A_2.

Table 2

Number of CPUs	T_1	S_1	T_2	S_2	T_3	S_3
2	1136	2.27	1270	2.03	1943	1.33
3	911	2.83	965	2.67	1004	2.57
4	882	2.92	860	3.00	787	3.27
5	909	2.83	690	3.73	584	4.41
7	785	3.28	507	5.08	436	5.91
9	804	3.20	424	6.08	364	7.08
11	780	3.30	429	6.00	311	8.28
13	866	2.97	323	7.98	262	9.83
15	833	3.09	246	10.47	236	10.92
20	935	2.76	—	—	220	11.71
24	—	—	—	—	211	12.21

The computational experiments were carried out on MVS-1000/M for a set of 15 contours on the raster map of the size of 1600×1400 pixels (the total segments number is 29951). The results of the experiments are shown in Table 2. For every algorithm A_i, its execution time T_i (in seconds) and speed-up S_i as related to one-processor execution (2574 s) are given for different numbers of operating processors. As is seen, the algorithm A_3 ranks below other approaches in the execution speed only for a small (2–3) number of processors.

The detailed analysis of critical sections of all the three algorithms is done in [8]. Here we just note that the results of experiments confirm the right of the chosen strategy of the library PLVIP building — providing alternative ways of parallelization of calculations.

5. Conclusion

Despite the serious limitations placed by the vertical image representation (uniformity of a computational algorithm, the growth of computational costs with an increase in color depth, etc.), the creation of the library PLVIP is an important step toward the understanding of the principles of building image processing systems on conventional multiprocessor computers. Based on experience gained, the work on creation of a universal high-performance image processing library on multiprocessor computers of the Siberian Supercomputer Center starts in the Image Processing Laboratory of the ICMMG SB RAS. As important principles of its building, the support of both vertical and conventional image formats and, also, the variety of parallelization methods ("image on single processor", "cutting into strips", "cutting into strips with overlapping", etc.) are assumed. In addition to the basic algorithms of image processing, the original ones elaborated in the laboratory and oriented to aerospace images will be also included in the library.

References

[1] Intel IPP. — http://intel.com/software/products/ipp/.

[2] OpenCV. — http://sourceforge.net/projects/opencvlibrary/.

[3] Potter J.L., Meilander W.C. Array processor supercomputers // Proc. IEEE. — 1989. — Vol. 77, No. 12. — P. 1896–1914.

[4] Shooman W. Parallel computing with vertical data // AFIPS Conf. Proc. — 1960. — Vol. 18. — P. 111–115.

[5] Rusin E.V. The Parallel Algorithm of Approximation of Earth Elevation Matrix by Given Contours // Proc. Int. Conf. "Mathematical Methods in Geophysics". — Novosibirsk: ICM&MG SB RAS, 2003. — Vol. 2. — P. 612–617 (In Russian).

[6] Kim P.A., Pyatkin V.P., Rusin E.V. Three massively parallel algorithms for solving computational geometry problems by using Euclidean distance transform // Pattern Recognition and Image Analysis. — 2004. — Vol. 14, No. 2. — P. 267–275.

[7] Rusin E.V. About one mass algorithm of calculation of the field of distances to a set on a discrete raster plane // Proc. Young Scientists Conf. — Novosibirsk: ICM&MG SB RAS, 2002. — P. 131–137 (In Russian).

[8] Rusin E.V. About parallelization of an algorithm of Earth elevation map recovery from given contours // Proc. Int. Conf. on Comput. Math. ICCM-2004. — Novosibirsk: ICM&MG SB RAS, 2004. — Vol. 1. — P. 197–202 (In Russian).