PARALLEL PROCESSING - THE EXAMPLE OF AUTOMATIC RELATIVE ORIENTATION

Beate Müller and Michael Hahn
Institute for Photogrammetry, Stuttgart University
Keplerstraße 11, D-7000 Stuttgart 1
ISPRS Intercommission II / III

ABSTRACT

Procedures designed to work on parallel processing hardware have two interdependent aspects: An algorithmic aspect and a hardware aspect. This is presumably the reason why in photogrammetry only little work is done in the field of parallel processing. The point in favour of parallelization is the excellent cost–benefit relation which is achieved by "super computers", such as an SIMD computer of the MasPar Computer Corporation which is used in this investigation.
In this paper principles of parallelization are introduced. We present a procedure for automatic relative orientation of digital stereo images. The single modules of this procedure — image pyramids, feature extraction, correspondence, orientation parameters — are described and discussed with regard to their adaptability for parallelization on SIMD computers. The implementation on such a computer is compared with the one on a standard computer. Finally some theoretical investigations due to the performance of the parallelized procedure are reported.

1 INTRODUCTION

From the title of the paper two questions are provoked immediately:

(1) Why use automatic relative orientation ?

(2) Why use parallel automatic relative orientation ?

A simple answer could be: Unless the extremely time consuming parts of the relative orientation of digital imagery are carried out very quickly, they might better be done by the human operator. This is state–of–the–art today, at least in photogrammetry. The measurement of some homologous points is done manually and the orientation parameters are computed by simple well–known algorithms.

A progressive answer to achieve an acceptable computation time for the whole process of automated relative orientation is to develop algorithms which work on parallel processing hardware. Parallelization in this field can be carried out on very different levels and hardware platforms, ranging from standard computers up to special purpose dedicated hardware systems.

Parallel processing means that several parts of an algorithm are processed simultaneously and/or multiple data are involved. Unfortunately, it is not possible to use existing algorithms without examining them, to put them on parallel processing hardware and then to run them efficiently. E.g., with a recursive procedure it could happen that from a large number of processors only one is actually working. That means, before implementing an algorithm one has to think about which parts of the algorithm can be processed in parallel. More difficult is the a priori investigation of the efficiency of the parallel working part, and the strongest challenge is to reformulate the algorithm to come to a solution which is well suited for parallel processing. From efficient parallel algorithms it is expected that they run on the parallel processing hardware significantly faster than the corresponding sequential algorithms on an appropriate sequential computer.

The aim of this investigation is to design a procedure for automatic relative orientation of digital stereo images, which runs on a massively parallel computer. All processes, including the lower level image processes the establishment of feature correspondences and the reconstruction of the orientation are analysed to be adapted or reformulated for parallel processing.

In photogrammetric literature the task of automatic relative orientation has been addressed very sparsely. Some experiments were carried out by Li [Li88] in the environment of an analytical plotter. More sophisticated is the procedure described by Schenk et al. [Sche90]. They first establish correspondences of contours defined by zero–crossings, then extract corners from these contours to find corresponding points. The points are put into a bundle adjustment program to obtain the orientation parameters. These processes are embedded in a coarse–to–fine control strategy mainly to find approximate values automatically. In Computer Vision there exist a vast amount of algorithms revolving around the problem of relative orientation. Usually, the problem of stereo correspondence and the aspect of the reconstruction of the orientation and structure parameters are addressed separately. In this context a variety of different features, such as points, straight lines, orthogonal corners, conic arcs, etc. are considered. For a recent review on the reconstruction problems refer to Huang and Netravali [Hua90].

This paper is organized as follows: In section 2 we describe different models of computers and characterize them by their level of parallelism. Basic criteria to determine the efficiency of algorithms are explained, which enables us to discuss the complexity of simple procedures. The next section gives a short description of an SIMD computer. The main part of this paper is the presentation of our procedure for automatic relative orientation and the investigations with respect to an efficient parallelization of the different modules. The orientation is usually considered to be a helpful step for other tasks, for example such as surface reconstruction. Because of this we especially pay attention to modules which are relevant for other problems also, like the image pyramids or the finite element modelling. In the last section the implementation is discussed and theoretical investigations of the performance of the parallelized procedure are reported.

With this paper we continue investigations on vectorization of modules for feature extraction reported by Hahn and Schneider [Ha91]. In a forthcoming paper [Mü92] experimental results concerning the quality of the orientation will be discussed, and a detailed comparison of the parallel and the sequential algorithm of the automatic orientation procedure will be given with respect to the computational performance evaluated by comprehensive tests.

2 GENERAL REMARKS ON PARALLEL PROCESSING

2.1 Computer Models

Nowadays increasing requirements made on computer systems demand continuous developments and improvements of these systems. One development is the change from one–processor–systems to multiple–processor–systems. To distinguish the different computer systems according to their instruction and data streams we follow Flynn's classification [Fly66] (see figure 1).

SISD (single instruction, single data) one–processor–system	**SIMD** (single instruction, multiple data) array processor
MISD (multiple instruction, single data)	**MIMD** (multiple instruction, multiple data) multicomputer

Figure 1: Flynn's scheme

Most of the SISD computer systems work with the von–Neumann–architecture. This means that they consist of only one processor decoding one instruction per cycle. In contrast to this the other three systems are provided with several processor elements (PEs). The processors of an MISD computer work simultaneously with different instructions on one data element. An SIMD system executes one sequential instruction stream which works in parallel on a large number of data elements. Because all processor elements are of the same speed (need the same time for an instruction) and are at the same stage of the instruction stream, no explicit synchronization is required. In this classification the class of MIMD computers is the most universal and the most powerful class. N processors execute different instructions on different data at the same time. This involves problems of memory organization, memory access and synchronization of the PEs. In both multiple data classes systems with shared memory (SM) and with local memory can be distinguished. They are called SM MIMD/SIMD and IN MIMD/SIMD, respectively. The PEs of an IN (interconnection network) system are connected by a network which can be arranged in various topologies (e.g. torus, array, binary tree, hypercube, etc., cf. [Akl89]).

According to the "granularity" of the processing unit it is possible to divide the different concepts of parallelism into four levels [Brä90] (see figure 1).

level of parallelism	**processing unit**	**example**
program	job, task	multitasking
procedure	process	MIMD program
expression	instructions	array computer (SIMD)
bit	inside instructions	von–Neumann architecture

Table 1: Levels of parallelism

On the highest, most coarse–grained level several jobs or independent programs are executed parallel. The parallelism on the procedural level is characterized by simultaneously execution of several independent processes of one program on different processors. Expression parallelism means that the instructions are executed sequentially, but parallel on a lot of processing units (data parallelism, massively parallel processing).

To obtain considerable increase in performance by parallelizing a sequential algorithm careful considerations about the most effective level of parallelism and the used hardware are needed. In practice usually one uses an available computer system and tries to fit the algorithm to work efficiently on it.

2.2 Complexity of Algorithms

For the determination of the efficiency of a parallel algorithm it is fundamental to compare the complexities of the sequential algorithm and the corresponding parallel algorithm for a certain problem. The complexity of an algorithm can be derived from the requirements on memory, computation time or used machines.

One criteria of efficiency is the time complexity $T(n)$, depending on the number of data n. In general not the real run time of an algorithm is interesting, but the order $O(n)$ of it. If the complexity for any amount of data differs only by a fixed factor, this factor can be neglected.
(E.g.: $O(2n+1) = O(n)$, $O(5n^4) = O(n^4)$)

time complexity	run time	example (sequential)
$O(1)$	constant	
$O(\log n)$	logarithmic	binary search
$O(n)$	linear	summation
$O(n * \log n)$		quick sort (divide–conquer)
$O(n^2)$	square	bubble sort
$O(n^k)$	polynomial	matrix multiplication (k=3)
$O(2^n)$	exponential	tree search (blind)
$O(n!)$		travelling salesman problem

Table 2: Time complexities

Figure 2 shows common time complexity classes [Wenz91]. Algorithms with the time complexity $T(n) = O(1)$ solve a problem in *constant* time, independent of the number of data. Algorithms with logarithmic characteristics $T(n) = O(\log n)$ are also good–natured. Because large n satisfies

$$O(\log n) < O(n) < O(n * \log n) < O(n^k) < O(2^n) < O(n!)$$

the run time increases according to this order. The acceleration of a parallel algorithm compared to a sequential one is called *speed–up* and is defined as the ratio

$$Sp(n) = \frac{T_1(n)}{T_N(n)} .$$

$T_1(n)$ is the required time to solve a problem with dimension n on a one–processor–computer system. $T_N(n)$ is the corresponding time for an N–processor–computer system. The speed–up is always between 1 and N.

$$1 \leq Sp \leq N$$

Different criteria for efficiency can compete with each other. Reduction of time has often to be compensated by more memory.

2.3 Description of the MasPar Computer

We have used the SIMD computer MasPar–1216C of the MasPar Computer Corporation to implement the algorithm for the automatic relative orientation. Figure 2 shows the global system architecture of the MasPar.

This MasPar system has 16 384 (2^{14}) processor elements (PEs) arranged as a 128 × 128 PE–array. Peak performance is given as 30 000 MIPS (millions of instructions per second) and 1500

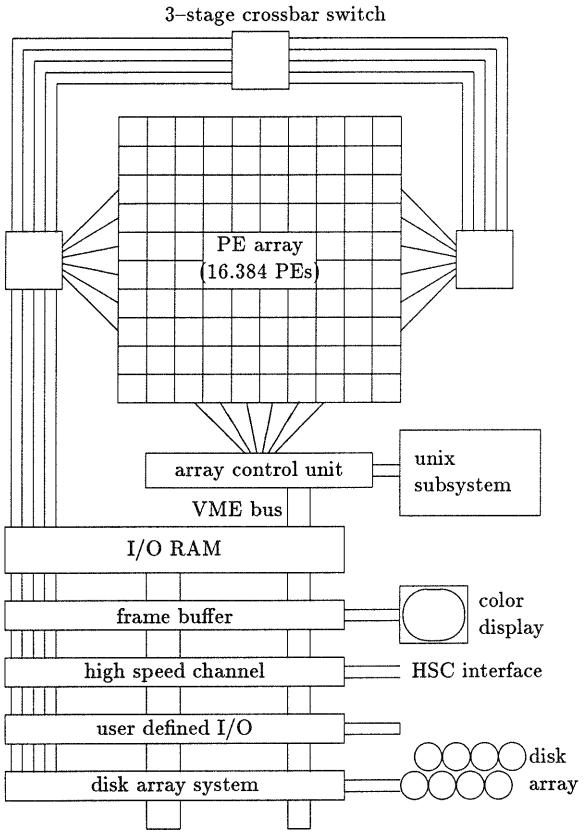

Figure 2: MasPar–1216C architecture

MFLOPS (millions of floating point operations per second) [Bla90]. System access is controlled by a front–end system (in figure 2 shown as unix subsystem) which performs all I/O services. Currently the front–end system is a DEC–station 5000, connected to the back–end by a VME–bus. The PE–array is controlled by the ACU (array control unit), which has a RISC–CPU. The ACU decodes all instructions and broadcasts them to the PEs.

Each PE consists of a 4–bit arithmetic logical unit, 32 32–bit registers and 16 kByte local memory. The PEs are connected by two different networks. First, a two–dimensional mesh structure connects each PE with its eight neighbour PEs. Second, a 3–stage crossbar switch, called router, allows parallel communication of random connection patterns, i.e. PEs can communicate with non–neighbour PEs directly. Of course the router is considerably slower than the eight–neighbourhood–connection. Most of the other components (frame buffer, high speed channel, disk arrays) are also typical for image processing systems.

Programming languages available for the MasPar are MPL (MasPar Application Language), a MasPar–specific extension of C, MPF (MasPar FORTRAN), a parallel FORTRAN dialect and a language called *Parallaxis*, based on Modula 2. Parallaxis is a development of the Institute for Parallel and Distributed High–Performance Computer Systems (IPVR) of the University of Stuttgart. For the investigations reported in this paper the programs have been written in MPL.

3 AUTOMATIC RELATIVE ORIENTATION

Let us first state more precisely what we mean by "automatic relative orientation". It is not necessary to remark, that this is more than only solving some equations known from analytical photogrammetry. We assume the interior orientation of the digital or digitized images to be known. Further, in the case of aerial images, which are of most interest from a practical point of view, the photographs are assumed to be taken under typical geometrical conditions in which one model is formed by two images. No assumptions are made regarding modelling the shape of the imaged 3D-surface, or concerning starting points for initialization of the correspondence process. Approximate values, wherever they are needed within the algorithms, have to be found automatically, e.g. by multiresolution techniques and a coarse–to–fine strategy. This standard technique is widely used in many image analysis and reconstruction tasks [Ack91]. The aim of the automatic orientation procedure is to estimate the orientation parameters. Structure information (e.g. the depth of corresponding points or the coordinates of these points in the model coordinate system) can be reconstructed within the orientation step, but this is not considered to be an essential part of the orientation. As in the conventional case the location of the corresponding features and the orientation can be used to compute structure by simple spatial resection in a further step.

In the following subsections the procedure is explained in detail. The structure of our approach is sketched in figure 3.

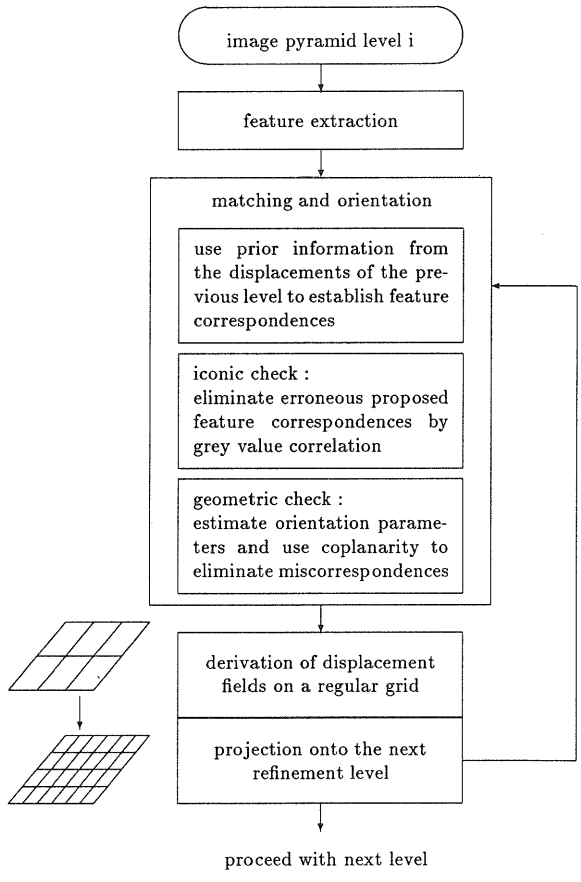

Figure 3: Structure of the automatic relative orientation procedure

3.1 Image Pyramids

Image pyramids are a multiresolution representation of an image. The original image (I_0) is convolved with a Gaussian kernel (G_σ) to obtain a smoothed image (I_0^σ) and then resampled by picking out each second pixel of a row and each second row to construct

(I_1) [Ack91] [Jä91]. The Gaussian pyramid can be generated by applying the following recursive formula :

$$\begin{aligned} I_i * G_\sigma &= I_i^\sigma \\ pick\ I_i^\sigma &\rightarrow I_{i+1} \end{aligned}$$

For smoothing with the Gaussian kernel we use a σ of 1 pixel radius. The second step is the resampling of I_i^σ in accordance with the coarsening of the spatial resolution.

3.2 Feature Extraction

On each level of the Gaussian image pyramid features are extracted with the point operator proposed by Förstner [Fö91] [Fö87]. The multiresolution representation of the symbolic image description forms a feature pyramid.

First, the point operator selects windows of significant (rough) texture. This selection is carried out by extracting local maxima of interest values in the corresponding images in combination with a threshold operation, which defines the number of significant windows (e.g. 1% of the total number of pixels). The point position of the feature (corner, circle, other isotropic texture) within this window is estimated and used in the following matching and orientation step.

3.3 Matching and Orientation

The matching and orientation process comprises the following three aspects :

- Use prior information from the previous level to establish feature correspondences.
- Eliminate erroneous feature correspondences by grey value correlation.
- Estimate the orientation and use the coplanarity condition to eliminate miscorrespondences.

3.3.1 Prior Information from the Previous Level (Approximate Values)

The predicted regular displacement fields obtained from the previous level (see section 3.4) are giving the approximate positions of the selected points in the corresponding stereo image partner. The approximate displacement vectors of the positions of the selected points are calculated by bilinear interpolation within the corresponding facets. The candidates of corresponding points are established by taking up all points found within a search window around the approximate position. The correspondence process which can result in multiple correspondences is sketched in figure 4.

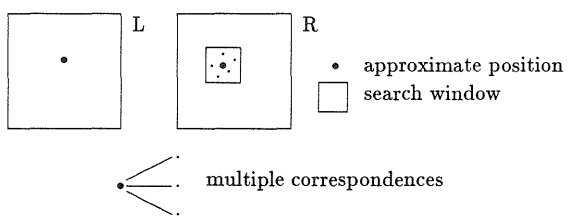

Figure 4: Establishing candidates of corresponding points

3.3.2 Iconic Check (Correlation)

All this candidates of corresponding points are verified by computing normalized cross correlation with the corresponding windows of these points. If the intensity information of the candidates is contradictory, the candidates are suspect and therefore eliminated.

3.3.3 Orientation Estimation and Geometric Check (Coplanarity Constraint)

It is well known that the image coordinates (x, y) and (x', y') of a point correspondence satisfy the coplanarity constraint, which can be formulated according to

$$[x'\ y'\ 1] \cdot E \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0\ . \qquad (1)$$

The so-called E–matrix [Hua89] can be composed by a skew-symmetric matrix T and a rotation matrix R.

$$E = T \cdot R \qquad (2)$$

$$E = \begin{bmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \\ e_7 & e_8 & e_9 \end{bmatrix}$$

$$T = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} \qquad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

The elements t_1, t_2, t_3 are the basis or the translation between the images, defined in the local model coordinate system.
Equation (1) is linear and homogeneous in the e_i terms, that means, the nine parameters e_i can be estimated up to an unknown scale parameter. Consequently, with at least eight point correspondences the elements of the E–matrix can be estimated by a linear least squares algorithm [Ste73] [Lon81]. We prefer this approximate but linear (direct) solution to the relative orientation. Of course the iterative algorithms which work with five or more points also can be used. But because our intension is to compute the orientation with a really large number of point correspondences P ($P \gg 5$ or 8), the differences between the linear and non linear solutions are small if a geometric configuration is given as it is in typical image flights with basis–to–height ratios in the range of 1 : 1 to 1 : 3.

If the E–matrix is calculated, all the candidates of corresponding points can be checked according to equation (1). That means, the coplanarity constraint is used to eliminate erroneous candidates. Together with the iconic check two relatively independent checks are carried out. One is working on the intensities directly, the other is exploiting a geometric condition (coplanarity).

3.4 Deriving Displacement Fields on a Regular Grid

All the point correspondences P_i, which are confirmed within the matching process, are used to derive a displacement vector field. The individual displacements $(p_x, p_y)_{P_i}$ are simply the difference between the estimated point location of the corresponding points. Because for matching on the next refinement level it is more convenient to have the displacement information on a regular grid, all displacements are used to reconstruct a displacement vector model by finite element (FE) modelling. The grid spacing for the displacement lattice is a multiple of the pixel size (e.g. 10–20 × pixel size). The displacements $(\hat{p_x}, \hat{p_y})_{ij}$ are estimated at the nodes ij, assuming a bilinear interpolation model within

a facet and smoothness of the displacement fields $\hat{p_x}(i,j)$ and $\hat{p_y}(i,j)$. Smoothness results from regularization of $\hat{p_x}$ and $\hat{p_y}$ with a Laplacian kernel.

The projection of the estimated regular displacement vector field onto the next level of the pyramid gives the predicted prior information used to start the matching on the next refinement level again (section 3.3.1).

Adjusting the regular displacement fields with the FE–model by least squares results in solving a normal equation system with a large but sparse normal equation matrix. Though algorithms to solve linear equations in parallel are known (e.g. cf. [Hos83]), they are not designed to exploit the sparseness of the matrix efficiently. Therefore we developed an algorithm based on the following idea: Apply the FE–approach not on all points simultaneously, but use it just like a local operator. This FE–operator with a window size of 3×3 or 5×5 nodes is utilized to estimate the displacements at the central node. Because local operators usually fit to data parallel approaches, the algorithm proposed is expected to be well-suited for this task. Global support of this local operation is reached by the prior information of the previous level (multilevel control) and the regularization.

3.5 Combining the Parts of the Algorithm

For both images of the stereo pair image pyramids are generated. Applying the Förstner operator points of interest are extracted at each level of the pyramids.

The matching process is started at a coarse level of resolution. Correspondence of points is established using iconic and geometric checks to eliminate erroneous point correspondences. The remaining point pairs are used to determine the regular displacement fields of the current pyramid level i. These estimated displacements $(\hat{p}_x, \hat{p}_y)$ are projected onto the next finer level $(i-1)$. By applying bilinear interpolation, from these predicted displacement fields the individual, approximate displacement vector of each interest point is derived. By establishing candidates of point correspondences this hierarchical matching process is continued until the finest level is reached.

The estimated E–matrix can be decomposed uniquely into rotation and translation as shown e.g. in [Tsa84]. In this way, the more familiar five parameters (instead of eight e_i's) of the relative orientation are obtained.

3.6 Example

The example we use to demonstrate the automatic orientation procedure is taken from [Ha88]. The digital image pair is a part of digitized aerial stereo imagery and each digital image has about 1200×900 pixels. Figure 5 shows the upper two levels of the image pyramids together with selected points. Most of them are of "general isotropic texture" a few are classified to be "corners" or "circular features". The matching process yields 15 correspondences on the lowest resolution level (selected are ~ 30 points in each image). The regular displacement vector field reconstructed by the FE–approach is shown in figure 6. This vector field is used as prior information to guide the search for establishing candidates of correspondences on the next refinement level and so on.

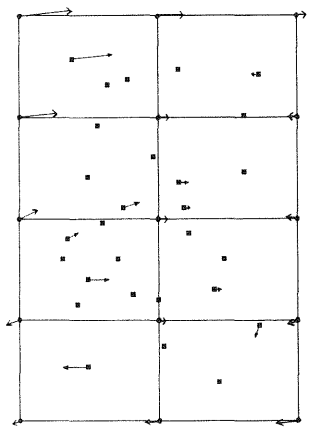

Figure 6: Displacement vectors and the derived regular displacement vector field

4 ANALYSIS OF PARALLEL ALGORITHMS

In this chapter we approach the parallelization of our procedure for relative orientation by discussing the complexity of the algorithms theoretically. The most important aspects which have to be taken into consideration at the implementation of the procedure are shortly described. In general any filter algorithm which works on a lot of independent data is suitable for parallelization on SIMD systems. The parts of our orientation algorithm where these conditions are given are the generation of image pyramids and the feature extraction. The matching process and the FE–procedure have to be considered separately.

The investigations described in this chapter are based on an ideal theoretical computer model which is provided with any number of PEs and any size of memory. The executed instructions are assumed to be of the same speed.

Let $T_1(n)$ be the time complexity of an algorithm for a one–processor computer and n be the number of data. $T_N(n)$ is the corresponding time complexity of an algorithm for an N–processor computer. Sp describes the speed–up as introduced in section 2.2.

4.1 Theoretical Investigations

The algorithm of generating image pyramids consists of filtering. Intuitively parallel filtering and resampling for each level of the pyramid is expected to be more efficient than sequential filtering and resampling. The time complexities for the sequential algorithms increase linearly with the number of pixels whereas the parallel resampling and filter algorithms are independent of the number of the data:

$$\begin{aligned} T_1^{pick}(n) &= O(n) \\ T_1^{G_\sigma}(n) &= (O(n) + O(\tfrac{n}{4}) + O(\tfrac{n}{16}) + \ldots) \cdot fs \\ &= O(p \cdot fs \cdot n) = O(n) \end{aligned}$$

$$\begin{aligned} T_N^{pick}(n) &= O(1) \\ T_N^{G_\sigma}(n) &= (O(1) + O(1) + O(1) + \ldots) \cdot fs \\ &= O(p \cdot fs \cdot 1) = O(1) \end{aligned}$$

where n is the number of pixels at level 0, p the number of pyramid levels and fs the time used for one filter operation. In both cases $T(n)$ depends on the filter size and the pyramid size. As long as the number of data n involved is very large compared with p and fs, the effect of these parameters can be neglected.

The algorithm for feature extraction consists of pixel oriented local operations like the determination of gradients and their squares, the calculation of normal equation matrices (which is a convolution), and the derivation of a measure of the isotropy of the texture. Non–maxima–suppression, thresholding operations and the estimation of the location of the optimal point position are also operations in which each pixel of the image is addressed

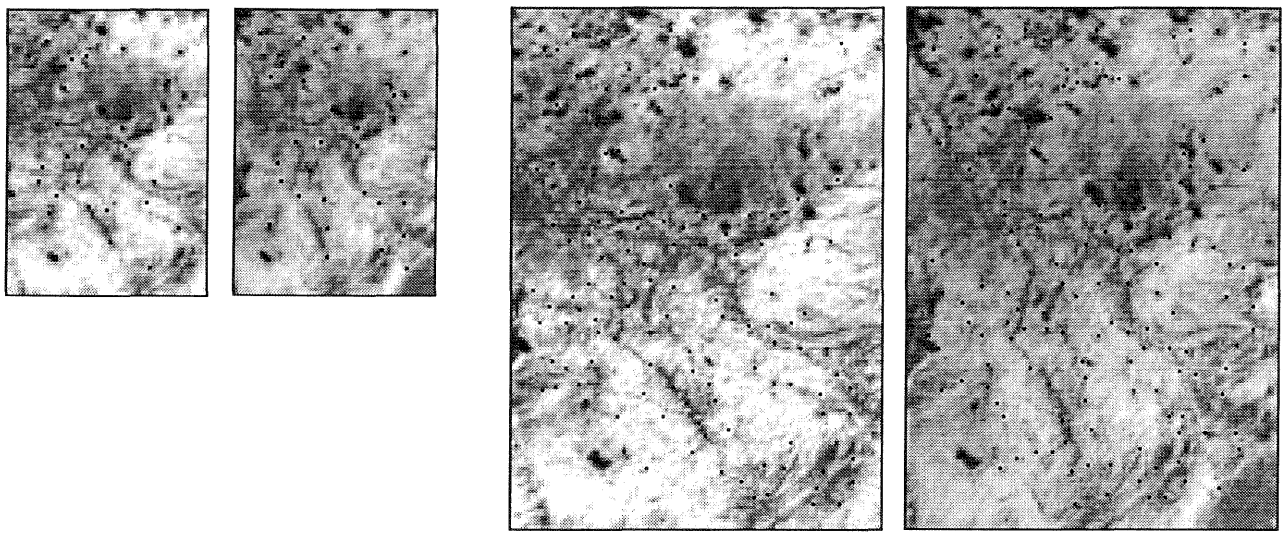

Figure 5: Two levels of the image pyramids. The selected points are marked in the images by black dots.

with the same sequence of instruction. Obviously, a data parallel approach is well–suited for this task. The time complexities are easily found to be :

$$\begin{aligned} T_1^{feature}(n) &= O(n) \\ T_N^{feature}(n) &= O(1)\ . \end{aligned}$$

That means, a parallel algorithm for generating image pyramids and feature extraction which is independent on the image size can be developed. In contrast to that, the computation time of the corresponding sequential algorithm increases linear with the number of pixels. For both algorithms together the speed–up is given by

$$\begin{aligned} Sp &= \frac{T_1(n)}{T_N(n)} = \frac{T_1^{pick}(n) + T_1^{G_\sigma}(n) + T_1^{feature}(n)}{T_N^{pick}(n) + T_N^{G_\sigma}(n) + T_N^{feature}(n)} \\ &= \frac{O_1(n)}{O_N(1)} = O(n) \end{aligned}$$

The algorithms above work pixel parallel. In contrast to that the matching part works parallel with the number of corresponding points m.

To investigate the time complexities of the matching algorithm described in section 3.3 let us first consider the three parts:

(1) Grey value correlation and thresholding operations.

(2) Estimation of the E–matrix, which is identical with a solution of a linear equation system.

(3) Finite element modelling.

It is easy to recognize, that the first part has the same characteristica as the pixel oriented operations mentioned above. The correlation and threshold operations depend on the number of point correspondences m and on the time fs needed for the particular operations. Assuming m to be the largest factor we obtain

$$\begin{aligned} T_1^{(1)}(m) &= T((fs_c + fs_{nm}) \cdot m) = O(m) \\ T_N^{(1)}(m) &= T((fs_c + fs_{nm}) \cdot 1) = O(1) \end{aligned}$$

Part two is the solution of a linear adjustment problem. The number of unknowns h is 8, and the number of equations depends on the amount of point correspondences m. The observation equations of the problem are

$$A_{(m\times h)} \cdot x_{(h\times 1)} - l_{(m\times 1)} = v_{(m\times 1)}\ .$$

The single steps to come to the solution are the calculation of A and l, A^TA, $(A^TA)^{-1}$, A^Tl, v and thresholding. The time complexity of the sequential computation is influenced by the matrix multiplication, whereas in the parallel algorithm most time is needed to calculate the design matrix A. Thus we find

$$\begin{aligned} T_1^{(2)}(m) &= T(h^2 \cdot m) = O(m) \\ T_N^{(2)}(m) &= T(h \cdot m) = O(m)\ . \end{aligned}$$

This suggests that no time will be saved by parallelizing this part.

For the third part which is the finite element approach it is most difficult to analyse the time complexities. The time needed for the calculation of the displacement vectors arises linearily with m in the sequential solution and can be solved in parallel in a constant time.

$$\begin{aligned} T_1^{(3.1)}(m) &= O(m) \\ T_N^{(3.1)}(m) &= O(1) \end{aligned}$$

The sequential estimation of the displacement vector field on the regular grid is of complexity $O(g^2 \cdot m)$, where g marks the number of the grid points. In the parallel solution we work with a local operator described in section 3.4. From this follows, that the time complexity depends only on the time used for the local operator. In detail there is a dependency from the number of correspondences and grid points within the operator window, which we address $O(Lop)$.

$$\begin{aligned} T_1^{(3.2)}(m) &= O(g^2 \cdot m) \\ T_N^{(3.2)}(Lop) &= O(Lop) \end{aligned}$$

For the projection of the estimated regular displacement fields the number of the grid points g is essential for the sequential solution; in parallel we have a constant time.

$$\begin{aligned} T_1^{(3.3)}(g) &= O(g) \\ T_N^{(3.3)}(g) &= O(1) \end{aligned}$$

The time dependency of the bilinear interpolation step applied to calculate the approximate displacements for all selected interest points (total number is i) is:

$$\begin{aligned} T_1^{(3.4)}(i) &= O(i) \\ T_N^{(3.4)}(i) &= O(1) \end{aligned}$$

The last step is searching candidates for correspondence in the search window. If the search window is of size n_s, the time complexity is

$$T_1^{(3.5)}(i) = T(n_s \cdot i) = O(i)$$
$$T_N^{(3.5)}(i) = T(n_s \cdot 1) = O(1)$$

Finally the total speed–up of the algorithm comprising matching and FE–modelling is obtained:

$$Sp = \frac{3 \cdot O(m) + O(g^2 \cdot m) + O(g) + 2 \cdot O(i)}{O(m) + O(Lop) + 5 \cdot O(1)}$$
$$= \frac{O(m) + O(g^2 \cdot m) + O(i)}{O(m) + O(Lop)}$$

With this theoretical considerations the time complexities of the automatic relative orientation are examined. Of course, it should be recalled, that working with a theoretical machine model (characterized at the beginning of this section) simplifications and approximations are necessary. Nevertheless, just this simplifications help to analyse the parallel algorithm before doing the implementation.

4.2 Implementation

For the implementation of the algorithms the following general aspects have to be taken into account:

- Make allowance for exploiting the given topology of the processor elements (see 2.3) if possible.
- Maximize load of the PEs.
- Organize the program to ensure minimal communication between the PEs, because communication instructions are slow.

The image pyramid and feature extraction algorithms are programmed relating the PEs to the pixels of the images. Because of simultaneously execution of an instruction the capacity of all PEs is fully used. The implementation of the matching procedure just under development. Details about the implementation, about real measured running times and a comparison between the run times of implemented sequential and parallel algorithms will be published later [Mü92]. There also a quality assessment of the orientation will be given.

5 CONCLUDING REMARKS

A method for automatic relative orientation of a digital stereo image pair has been presented. Some parts of the procedure such as the generation of image pyramids or the extraction of significant points can also be used for other tasks (e.g. surface reconstruction). The other two important parts of the procedure are the matching to find the displacement vectors and the finite element process to estimate the displacement fields on a predefined regular grid.

To discuss this procedure with respect to parallelization an introduction into some fundamental principles of parallelization is given. By a theoretical analysis of the individual processes the expected speed–up of the parallel procedure with respect to a sequential algorithm is worked out. The experimental prove concerning the performance of the whole procedure as well as to the quality is just in work and will be reported in [Mü92].

Finally we want to note, that by a simple modification of the procedure an approach for surface reconstruction can be obtained. If the orientation on each level is found, the corresponding points can directly be used to estimate structure (depth in the local model coordinate system). The finite element process applied to the structure gives a surface model. By backprojection to the image plane the prior information for starting the matching process on the next refinement level is obtained. At this point we are in the loop of the described orientation procedure again.

References

[Ack91] Ackermann, F., Hahn, M.: Image Pyramids for Digital Photogrammetry.
In: Ebner,H. et al.: Digital Photogrammetric Systems, Wichman Verlag, Karlsruhe, 1991

[Akl89] Akl, S.: The Design and Analysis of Parallel Algorithms.
Prentice Hall, New York, 1989

[Bla90] Blank, T.: The MasPar MP–1 Architecture.
Proceedings of the IEEE Compcon Spring 1990, February 1990

[Brä90] Bräunl, T.: Massiv parallele Programmierung mit dem Parallaxis – Modell.
Informatik Fachberichte 246, Springer Verlag, 1990

[Fly66] Flynn, M.J.: Very high–speed computing systems.
Proceedings of the IEEE, vol. 54, p. 1901–1910, 1966

[Fö87] Förstner, W., Gülch, E.: A fast Operator for Detection and Location of Distinct Points, Corners and Centers of Circular Features.
In: Proc. ISPRS Intercommision Workshop on "Fast Processing of Photogrammetric Data", Interlaken, June, 1987

[Fö91] Förstner, W.: Statistische Verfahren für die automatische Bildanalyse und ihre Bewertung bei der Objekterkennung und – vermessung.
Habilitationsschrift, DGK, Reihe C, no. 370, München, 1991

[Ha88] Hahn, M., Förstner, W.: The Applicability of a Feature Based and a Least Squares Matching Algorithm for DEM – Acquisition. Int. Arch. Ph. RS. (27), B9, III pp. 150, 1988

[Ha91] Hahn, M., Schneider, F.: Feature Based Surface Reconstruction — A Hierarchical Approach Developed for MOMS-02 Imagery.
IGARRS, 1991

[Hos83] Hossfeld, F.: Parallele Algorithmen.
Informatik Fachberichte 64, Springer Verlag, Berlin, Heidelberg, 1983

[Hua89] Huang, T.S., Faugeras, O.D.: Some Properties of the E Matrix in Two – View Motion Estimation.
IEEE Trans. on PAMI, vol. 11, no. 12, p. 1310–1312, December, 1989

[Hua90] Huang, T.S, Netravali, A.N.: Motion and Structure from Feature Correspondences: A Review.
In: Huang: Tutorial on Computer Vision and Dynamic Scene Analysis, ISPRS Comm. V Symposium, Zürich 1990

[Jä91] Jähne, B.: Digitale Bildverarbeitung.
2. Auflage, Springer Verlag, Berlin, Heidelberg, 1991

[Li88] Li, M.: High Precision Relative Orientation Using the Feature Based Matching Techniques.
ISPRS 27 B3, p. 456–465, 1988

[Lon81] Longuet – Higgins, H. C.: A Computer Algorithm for Reconstructing a Scene from Two Projections.
Nature 293, 1981

[Mü92] Müller, B., Hahn, M.: Quality and Performance Analysis of Automatic Parallel Relative Orientation. In preparation, Stuttgart, 1992

[Sche90] Schenk, T., Toth, Ch., Li, J.–Ch.: Zur automatischen Orientierung von digitalen Bildpaaren. ZPF, 58. Jahrgang, 6/90, p. 182–189, November, 1990

[Ste73] Stephanovic, P.: Relative Orientation – A New Approach. The ITC Journal 3, 1973

[Tsa84] Tsai, R. Y., Huang, T. S.: Uniqueness and Estimation of Three – Dimensional Motion Parameters of Rigid Objects with Curved Surfaces. IEEE Trans. on PAMI, vol. 6, no. 1, p. 13–27, January, 1989

[Wen89] Weng, J., Huang, T. S., Ahuja, N.: Motion and Structure from Two Perspective Views : Algorithms, Error Analysis, and Error Estimation. IEEE Trans. on PAMI, vol. 11, no. 5, p. 451–475, May, 1989

[Wenz91] Wenzel, L.: Parallele Programmierkonzepte. Franzis Verlag, München, 1991