THE HYBRID CPU/GPU IMPLEMENTATION OF THE COMPUTATIONAL PROCEDURE FOR DIGITAL TERRAIN MODELS GENERATION FROM SATELLITE IMAGES

V.A. Fursov [1, 2], Ye.V. Goshin [1, 2], A.P. Kotov [1, 2],
[1] Image Processing Systems Institute of RAS – Branch of the FSRC "Crystallography and Photonics" RAS, Samara, Russia,
[2] Samara National Research University, Samara, Russia

Abstract

In this paper a procedure of building a digital terrain model (DTM) from the satellite images is researched. The procedure is based on the authors' previously developed algorithms of fast image matching for building disparity maps implemented on GPUs (Graphics Processing Units). In this paper we propose a computational procedure for constructing a DTM from the satellite stereo images. Experimental studies have shown that while this procedure constructs a DTM that may be less accurate than the one achieved with the use of the ENVI software, it offers a significantly shorter time of processing.

Keywords: digital image processing, stereo images, 3D-scene reconstruction, image matching, CUDA-technology, ENVI.

Citation: Fursov VA, Goshin YeV, Kotov AP. The hybrid CPU/GPU implementation of the computational procedure for digital terrain models generation from satellite images. Computer Optics 2016; 40(5): 721-728. DOI: 10.18287/2412-6179-2016-40-5-721-728.

Acknowledgements: The work was funded by the Russian Science Foundation (grant #14-31-00014).

Introduction

Building a digital terrain model (DTM) from satellite images is one of crucial tasks of the Earth remote sensing data (ERS) processing and analysis.

In particular, paper [1] provides the analysis and comparison of the digital elevation models (DEM) from high resolution QuickBird and Pleiades satellite stereo images. Paper [2] describes generation and evaluation of DEM from two panchromatic cameras of the Cartosat-1 satellite, which are capable of acquiring stereoscopic data along the orbital track.

Nowadays researchers conduct their experiments not only on satellite images, but also on synthetic one. Studies not only dedicated to real satellite images, but also to synthetic images. E.g., paper [3] draws a comparison between DEMs generated with the use of forward, reverse and other possible synthetic stereo pairs for different types of topographies.

In most of papers ground control points (GCP) and / or rational polynomial coefficients (RPC) are used for the DEM generation, so part of our study is dedicated to introduction of RPC coefficients [4, 5] into our procedure.

There is also a wide range of papers highlighting practical use of DEM, e.g. papers [6, 7].

The software components for the DTM construction are incorporated in most of commercial software systems of remote sensing data processing. ENVI, PHOTOMOD and Geomatica [8 – 10] are the best-known systems. Nevertheless, there is a problem in the efficiency of the DTM construction. As a rule, space images are of large dimensions, which cause some processing problems associated with both limited volumes of memory and computational capability. Therefore, users have to select some relatively small fragments in the initial images and build local terrain models.

However, there is often need to solve this problem in real time, for example, to monitor emergencies, analyze the target environment, or calculate routes, etc. Generally, GPU computing with the use of CUDA technology is applied to stereo reconstruction problem in cases when the number of points of images is small but there is a constant flow of images, e.g. video from unmanned aerial vehicles [11, 12]. Conversely, in this research we have a different issue which consists in processing of small number of large images. Nevertheless, our computational procedure still allows us to improve the speed of DTM construction with the use of CUDA technology implementation [13].

In this paper we describe a procedure of DTM construction from remote sensing data [14] and provide a detailed description of its main stages. The main attention is paid to the description of the distinctive features of these stages, in comparison with the known. We also analyze the degree of internal parallelism. Taking into account this analysis, we propose a hybrid general procedure for DTM construction from satellite images. In general, the procedure is realized in the hybrid computing systems consisting of both graphics and central processors. Our aim is to show that the implementation of the general procedure on hybrid CPU / GPU system provides substantially higher speed of ERS data processing compared with the software package ENVI, which comes at the cost of occasional loss of DTM reconstruction accuracy.

1. Main stages of the procedure

The general scheme of the main stages of the considered procedure for three-dimensional DTM reconstruction from stereo satellite images is shown in Fig. 1. The main stages of this procedure are the rectification of images, image matching (finding the corresponding points) and determining the 3D coordinates of the DTM.

The initial data for this procedure are high-resolution satellite images (HRSI), obtained from different perspectives, as well as metadata represented in the form of a set of RPC (Rational polynomial coefficients) [4] $\underset{(20\times1)}{a}, \underset{(20\times1)}{b}, \underset{(20\times1)}{c}, \underset{(20\times1)}{d}$. These coefficients represent a model of image registration by a satellite camera.

$$Y = (\mathbf{a}^T\mathbf{u})/(\mathbf{b}^T\mathbf{u}), X = (\mathbf{c}^T\mathbf{u})/(\mathbf{d}^T\mathbf{u}), \quad (1)$$

where

$$\mathbf{u} = [1\ L\ P\ H\ LP\ LH\ PH\ L^2\ P^2\ H^2\ PLH\ L^3\ LP^2\ LH^2\ ,\ L^2P\ P^3\ PH^2\ L^2H\ P^2H\ H^3]^T$$

X, Y are normalised coordinates of the images, and P, L, H are normalised coordinates of a point in 3D space.

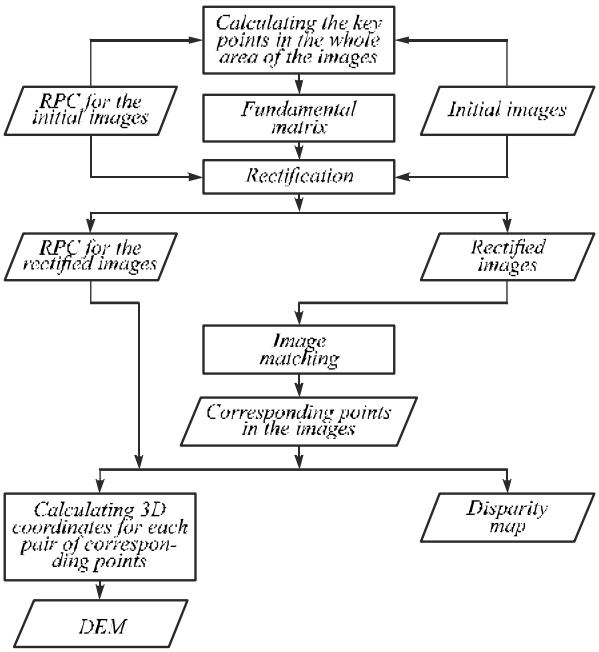

Fig. 1. The main stages of the computational procedure

Normalised coordinates P, L, H of a 3D point and the coordinates of images X, Y are defined by the following equations:

$$P = \frac{\varphi - LAT\_OFF}{LAT\_SCALE}, L = \frac{\lambda - LONG\_OFF}{LONG\_SCALE},$$
$$X = \frac{x - SAMP\_OFF}{SAMP\_SCALE}, Y = \frac{y - LINE\_OFF}{LINE\_SCALE}, \quad (2)$$
$$H = \frac{h - H\_OFF}{H\_SCALE},$$

where *LAT_OFF*, *LAT_SCALE*, *LONG_OFF*, *LONG_SCALE*, *H_OFF* and *H_SCALE* are the normalized parameters of the ground point coordinates, while *SAMP_OFF*, *SAMP_SCALE*, *LINE_OFF* and *LINE_SCALE* are the normalised parameters of the image point coordinates.

Rectification of stereo images is a transformation in which the corresponding points in the images are arranged in the same rows. The aim of the rectification stage is to simplify stereo images processing, in particular, the search of the corresponding points. It is also more convenient to build a disparity (horizontal parallax) map, as in this case there is a disparity in one coordinate only.

The main problem in the construction of the DTM procedure is the image matching, in particular, determining the corresponding points on different views. To apply methods for image matching, the images are typically rectified (the rows of the images are brought to the same orientation).

To construct the DTM from stereo satellite images, three well-known classes of image matching methods are applied: local, global and semi-global [5]. To match the images, for each point (x_0, y_0) in the first image a corresponding point $(x_0 + \Delta x, y_0 + \Delta y)$ in the second image is searched. In the case of rectified images, one-dimensional search can be used instead of the two-dimensional search. In this case, the problem is reduced to calculating the disparity between the images.

As a result of the matching, a disparity map can be formed, which is a visualization of the obtained shifts: the more the corresponding point of the initial image is shifted, the brighter each pixel of the disparity map is.

To process the rectified images, we have introduced RPC conversion into the procedure. Since RPC are specified for the initial images, it is necessary to calculate new coefficients for the rectified images according to the projective transformations applied to both images.

Calculation of three-dimensional points in the global coordinate system from the obtained corresponding points is performed using RPC for the rectified images. To do this, a nonlinear least-squares method is normally used [15].

Next, we give a more detailed, but a rather brief description of mathematical models and algorithms to be implemented at these stages of procedure. The stage of image matching is accompanied by a description of parallel implementation of the proposed algorithm. The final section provides examples of the procedure implementation and the performance characteristics achieved.

2. Rectification and RPC converting

The initial data for this stage are a pair of satellite images recorded at different angles sharing some area.

There are several approaches to the rectification: using the known shooting parameters (exact model), using the known fractional-rational image function (RFM, rational functional model) specifying the correspondence between the image coordinates and three-dimensional point in space [16], and using known or found corresponding points between the images (projective, polynomial model).

Key points are formed using the coefficients of the rational function (RPC), which are part of the metadata. After building the set of key points, the fundamental matrix is calculated.

The corresponding points on two projections are connected by a 3×3 fundamental matrix F [17], in particular, for the points with the coordinates set by 3×1 vectors $\mathbf{m}_L, \mathbf{m}_R$: $\mathbf{m}_L = [x_L, y_L, 1]^T$, $\mathbf{m}_R = [x_R, y_R, 1]^R$ the following condition is met:

$$\mathbf{m}_R^T \mathbf{F} \mathbf{m}_L = 0, \quad (3)$$

where

$$\mathbf{F} = \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix}. \quad (4)$$

Equation (2) defines the epipolar constraints on permissible coordinates of corresponding points in stereo

images. It is obvious that it is necessary to know the exact fundamental matrix to take these constraints into account.

To determine the parameters of the fundamental matrix, a system of linear equations is solved by least squares method with at least eight given corresponding points. The corresponding points in the two images will be in the same rows if the fundamental matrix has the following form:

$$\mathbf{F}' = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}. \quad (5)$$

To achieve this, a particular projective transformation is applied to both images [18]. For the first and the second images, these transformation matrices are denoted as $\mathbf{H}_L$ and $\mathbf{H}_R$, respectively, and satisfy the following equation:

$$(\mathbf{H}_R\mathbf{m}_R)^T \mathbf{F}'\left(\mathbf{H}_L\mathbf{m}_L\right) = 0. \quad (6)$$

As the result of the rectification, the corresponding points will be in the same rows.

As previously mentioned, the initial RPC cannot be applied to the obtained rectified images. Therefore, we need to calculate additional coefficients $H_{11}H_{12}…H_{33}$ based on the projective transformation matrix **H**. We can rewrite the equations (2) with these coefficients as following:

$$\begin{aligned} & x + \frac{H_{11}x + H_{12}y + H_{13}}{H_{31}x + H_{32}y + H_{33}} = \\ & = X * SAMP\_SCALE + SAMP\_OFF, \\ & y + \frac{H_{21}x + H_{22}y + H_{23}}{H_{31}x + H_{32}y + H_{33}} = \\ & = Y * LINE\_SCALE + LINE\_OFF, \end{aligned} \quad (7)$$

where matrix **H** is equal to $\mathbf{H}_R$ for point $\mathbf{m}_L$ and to $\mathbf{H}_L$ for point $\mathbf{m}_R$.

3. Stereo matching

We use the local method of image matching consisting in finding the shifts by comparing the distribution functions of brightness on fragments of the left and right stereo images. For each pixel of the left stereo image we search for the corresponding pixel in the right image within a local window.

The ENVI software package implements the local method, in which the criterion for the similarity of pixels is a normalized cross-correlation between the brightness values of the pixels in the left and right images.

Another modification of the local method taking into account epipolar constraints via penalty coefficients is implemented in paper [19]. In this study, we do not use the penalty coefficients, because the local method is implemented to the rectified images. Therefore, the search area is focused on the epipolar lines, at small intervals vertically.

Here is a detailed description of the implemented local method. Let us denote the coordinates of the points in the first image as (x_0, y_0), and the coordinates of the corresponding points in the second image as $(x_0 + \Delta x, y_0 + \Delta y)$, where Δx and Δy are relative shifts of the coordinates x_0 and y_0, respectively. Let $I_L(x, y)$ and $I_R(x, y)$ be the intensity distribution function of the counts in these images. Matching algorithm consists in detection for each point (x_0, y_0) in the first image a corresponding point $(x_0 + \Delta x, y_0 + \Delta y)$ in the second image by maximizing the normalized cross correlation coefficient $E(x_0, y_0, \Delta x, \Delta y)$:

$$E\left(x_0, y_0, \Delta x, \Delta y\right) = \frac{\sum\limits_{x,y \in D(x_0,y_0)} \left(I_L(x,y) - \overline{I_L}\right)\left(I_R(x+\Delta x, y+\Delta y) - \overline{I_R}\right)}{\sqrt{\sum\limits_{x,y \in D(x_0,y_0)} \left(I_L(x,y) - \overline{I_L}\right)^2 \sum\limits_{x,y \in D(x_0,y_0)} \left(I_R(x+\Delta x, y+\Delta y) - \overline{I_R}\right)^2}},$$
$$\overline{I_L} = \frac{1}{N} \sum_{x,y \in D(x_0,y_0)} I_L(x,y), \quad \overline{I_R} = \frac{1}{N} \sum_{x,y \in D(x_0,y_0)} I_R(x+\Delta x, y+\Delta y), \quad (8)$$

where $D(x_0, y_0)$ is an area around point (x_0, y_0), N is a number of pixels in the area $D(x_0, y_0)$.

Parallel implementation of the described algorithm is shown in Fig. 2. The interaction between CPU and GPU is presented in the form of the interaction between the three blocks. The first and third blocks include routines that are only executed on CPU. The results of their implementation are used in the second block to run CUDA kernels on GPU.

In the first block a pyramid of images is generated which is used for further image matching. For better visualization, Fig. 2 shows a three-level pyramid of images (Block 1) for a pair of rectified images. The pyramid is formed as a set of images obtained by decreasing the resolution twice in both coordinates. Thus, an image with a 2^N times smaller resolution than the original one is formed at the N^{th} level of the pyramid.

After the pair of images for the third level of the pyramid has been formed, the routines in the second block begin. These routines process the image of the third level of the pyramid with zero initial shifts.

When the routines in CUDA kernel have been completed, relative shifts for the left image are formed as an array. After copying the array from GPU memory to RAM, they are saved as an image. This image is a disparity map, which is scaled for all levels of the pyramid in Block 3 (see Fig. 3).

At the next run of CUDA kernel the rectified images from the second level of the pyramid and the initial shifts from the previous run are used. The values of the initial shifts coordinates are doubled (Block 3). The number of CUDA kernel runs depends on the number of pyramid levels.

Global memory on GPU is allocated only once for all operations. The total amount of the allocated memory is equal to the number of pixels of the left image for the N-level pyramid × (2 × 16 bit + 64 bit). This is because the initial image pixel depth is 16 bits, and the matrix that holds the relative horizontal and vertical shifts comprises two float values in each element of the matrix. Memory deallocation on the GPU is performed after all the calculations have been completed. To run CUDA kernel on the N^{th} level of the pyramid, the following parameters are used: mesh size, block size, number of threads, image size, the size of the search area and the size of the processing window.

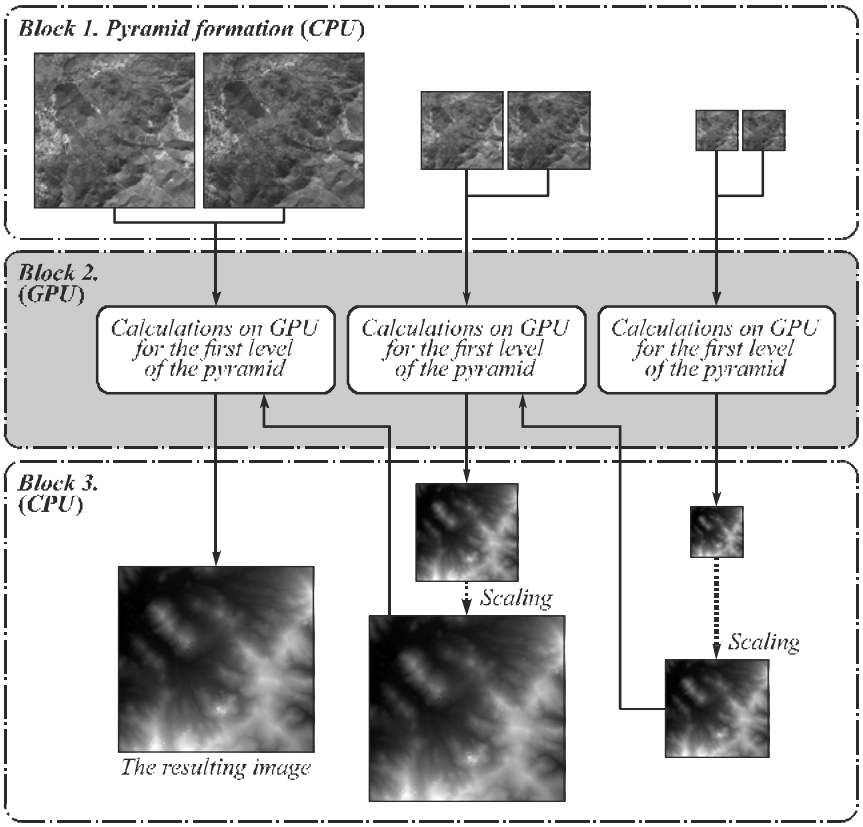

Fig. 2. CPU/GPU interaction for a three-level pyramid

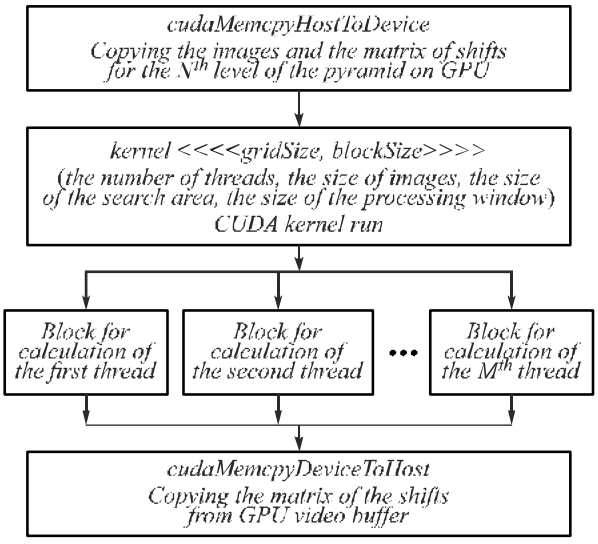

Fig. 3. Block of calculations on GPU

Before running CUDA kernel the required data are copied from the RAM memory to global memory of a video card. Memory for the results storage is also allocated on the graphics card. Each thread calculates the Euclidean norm for the two selected descriptors. Descriptor of a given point is a feature vector of the image fragment with its centre in the desired corresponding point. To find the corresponding point, it is necessary to calculate the Euclidean norm for all points descriptors selected in the search area as possibly corresponding.

The number of the created threads equals to the size of the image (in pixels).

Fig. 4 shows the enlarged scheme of the calculations, carried out by one thread. In Fig. 3 this computation block is identified as Block for calculation of the thread. Each thread performs calculations independently of the other threads. This embodiment allows us to avoid the redundant data array formation on GPU. Consecutive comparing of the Euclidean norms on CPU is also omitted.

After determining the maximum normalized cross correlation coefficient, the relative shift is determined on CPU.

4. 3D model calculation

After the image matching, for each pair of corresponding points we can calculate a point in 3D space. To do this, we can use the method described in paper [15]. To calculate the desired three-dimensional coordinates, alongside with the coordinates of the corresponding points, we use meta-information about the rational function coefficients (RPC). The initial data for the consid-

ered approach consist of terrain images obtained from different angles, and metadata in the form of an RPC set [4] $\underset{(20\times1)}{\mathbf{a}}$, $\underset{(20\times1)}{\mathbf{b}}$, $\underset{(20\times1)}{\mathbf{c}}$, $\underset{(20\times1)}{\mathbf{d}}$, which represent a model of image formation by a satellite camera. On the basis of these data, we obtain the following system of nonlinear equations for the left and right images (indicees *L* and *R* stand for the left and right images, respectively):

$$\begin{aligned} Y_L &= g_L(\varphi,\lambda,h) = \frac{N_{YL}(P,L,H)}{D_{YL}(P,L,H)} = \frac{\mathbf{a}_L^T\mathbf{u}}{\mathbf{b}_L^T\mathbf{u}}, \\ X_L &= f_L(\varphi,\lambda,h) = \frac{N_{XL}(P,L,H)}{D_{XL}(P,L,H)} = \frac{\mathbf{c}_L^T\mathbf{u}}{\mathbf{d}_L^T\mathbf{u}}, \\ Y_R &= g_R(\varphi,\lambda,h) = \frac{N_{YR}(P,L,H)}{D_{YR}(P,L,H)} = \frac{\mathbf{a}_R^T\mathbf{u}}{\mathbf{b}_R^T\mathbf{u}}, \\ X_R &= f_R(\varphi,\lambda,h) = \frac{N_{XR}(P,L,H)}{D_{XR}(P,L,H)} = \frac{\mathbf{c}_R^T\mathbf{u}}{\mathbf{d}_R^T\mathbf{u}}, \end{aligned} \tag{9}$$

where

$$\begin{aligned} N_{YL}(P,L,H) &= a_0 + a_1L + a_2P + a_3H + a_4LP \\ &+ a_5LH + a_6PH + a_7L^2 + a_8P^2 + a_9H^2 \\ &+ a_{10}PLH + a_{11}L^3 + a_{12}LP^2 + a_{13}LH^2 \\ &+ a_{14}L^2P + a_{15}P^3 + a_{16}PH^2 \\ &+ a_{17}L^2H + a_{18}P^2H + a_{19}H^3, \end{aligned} \tag{10}$$

$$\begin{aligned} D_{YL}(P,L,H) &= b_0 + b_1L + b_2P + b_3H + b_4LP \\ &+ b_5LH + b_6PH + b_7L^2 + b_8P^2 + b_9H^2 \\ &+ b_{10}PLH + b_{11}L^3 + b_{12}LP^2 + b_{13}LH^2 \\ &+ b_{14}L^2P + b_{15}P^3 + b_{16}PH^2 \\ &+ b_{17}L^2H + b_{18}P^2H + b_{19}H^3, \end{aligned} \tag{11}$$

$$\begin{aligned} N_{XL}(P,L,H) &= c_0 + c_1L + c_2P + c_3H + c_4LP \\ &+ c_5LH + c_6PH + c_7L^2 + c_8P^2 + c_9H^2 \\ &+ c_{10}PLH + c_{11}L^3 + c_{12}LP^2 + c_{13}LH^2 \\ &+ c_{14}L^2P + c_{15}P^3 + c_{16}PH^2 \\ &+ c_{17}L^2H + c_{18}P^2H + c_{19}H^3, \end{aligned} \tag{12}$$

$$\begin{aligned} D_{XL}(P,L,H) &= d_0 + d_1L + d_2P + d_3H + d_4LP \\ &+ d_5LH + d_6PH + d_7L^2 + d_8P^2 + d_9H^2 \\ &+ d_{10}PLH + d_{11}L^3 + d_{12}LP^2 + d_{13}LH^2 \\ &+ d_{14}L^2P + d_{15}P^3 + d_{16}PH^2 \\ &+ d_{17}L^2H + d_{18}P^2H + d_{19}H^3. \end{aligned} \tag{13}$$

Similar expressions for $N_{YR}(P,L,H)$, $D_{YR}(P,L,H)$, $N_{XR}(P,L,H)$, $D_{XR}(P,L,H)$ differ from the above written ones by the values of their coefficients.

The algorithm for the solution of the nonlinear system of equations (6) enables us to find the 3D point coordinates P, L, H in the global coordinate system. This algorithm is performed in two steps.

Step 1. The initial values of ground coordinates are calculated. The distortions caused by the optical projection can be represented by the ratios of first-order terms in the RPC. Excluding the RPC of high order, we can write the functions transforming the object coordinates into pixel coordinates as follows:

$$\begin{aligned} Y_L &= \frac{a_0 + a_1L^0 + a_2P^0 + a_3H^0}{b_0 + b_1L^0 + b_2P^0 + b_3H^0}, \\ X_L &= \frac{c_0 + c_1L^0 + c_2P^0 + c_3H^0}{d_0 + d_1L^0 + d_2P^0 + d_3H^0}, \end{aligned} \tag{14}$$

where $(a_k, b_k, c_k, d_k)k = 0,\ldots,3$ are first order RPC. Similar expressions can be written for X_R and Y_R.

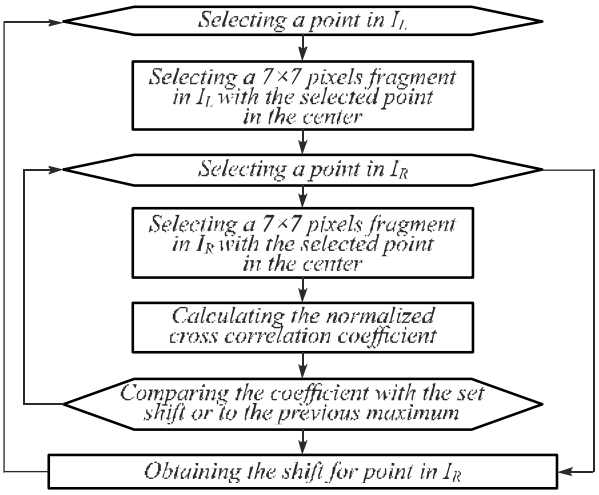

Fig. 4. Scheme of the computation on one thread at CUDA kernel run

These equations can be transformed into the following equation groups:

$$\begin{aligned} &(a_0 - b_0X) + (a_1 - b_1X)L^0 + (a_2 - b_2X)P^0 + \\ &\quad + (a_3 - b_3X)H^0 = 0, \\ &(c_0 - d_0Y) + (c_1 - d_1Y)L^0 + (c_2 - d_2Y)P^0 + \\ &\quad + (c_3 - d_3Y)H^0 = 0. \end{aligned} \tag{15}$$

Needless to say, we only need to know the first order RPC of two or more images to calculate the initial object coordinates (P^0, L^0, H^0) either from all the above mentioned equations or from only three of them, two of which involve Y and one X.

Step 2. The final object coordinates are calculated. By performing a Taylor expansion, the observation equations can be written as

$$\begin{aligned} Y_L &= \frac{N_{YL}(P^0,L^0,H^0)}{D_{YL}(P^0,L^0,H^0)} + \frac{\partial g}{\partial P}\Delta P + \frac{\partial g}{\partial L}\Delta L + \frac{\partial g}{\partial H}\Delta H, \\ X_L &= \frac{N_{XL}(P^0,L^0,H^0)}{D_{XL}(P^0,L^0,H^0)} + \frac{\partial f}{\partial P}\Delta P + \frac{\partial f}{\partial L}\Delta L + \frac{\partial f}{\partial H}\Delta H, \end{aligned} \tag{16}$$

and the final object coordinates can be obtained with the use of least square adjustment.

Our implementation of this algorithm uses data decomposition for parallelism executing a single instruction on multiple data (SIMD). We take the algorithm of system solution (10) described in the previous section as the single instruction. It is justified by the fact that the calculations on each thread are performed independently.

The computation time of a serial C++ algorithm mainly depends on such characteristics of a processor as its clock rate and instructions. Basically, to execute such code in IDE Microsoft Visual Studio, a developer only needs to write the code, and command line parameters are generated automatically by integral means of Visual Studio. In case of parallel implementation of CUDA algorithm we need to take into consideration the characteristics shown in the Table 1. The following features serve as restrictions for the implementation and CUDA kernel launch.

Table 1. CUDA Driver Version and GeForce GTX 750 Ti specifications

CUDA Driver Version / Runtime Version	7.5 / 7.5
CUDA Capability Major/Minor version number	5.0
Streaming Multiprocessors (SM) count	5
Total amount of global memory	2048 MB
Total amount of constant memory	65536 bytes
Total number of registers available per block	65536 bytes
Maximum number of threads per multiprocessor	2048

To find the solution of the system (10) we used the following non-atomic operations: vector multiplication, dot product and matrix inversion. In CPU and GPU implementation these operations are written in C++ without the use of linear algebra libraries with a view to higher efficiency.

The speeding-up s of our CUDA algorithm in comparison with the serial one can be estimated by the formula

$$s = (t_{HtoD} + t_{\text{kernel}} + t_{DtoH})/t_{ENVI}. \quad (17)$$

This formula takes into account communication time between CPU and GPU. Time of input data transfer from CPU memory into GPU memory is denoted as t_{HtoD}, time of CUDA kernel output transfer from GPU memory into CPU memory is denoted as t_{DtoH}, CUDA kernel computation time as t_{kernel} and ENVI software application computation time as t_{ENVI}.

CUDA kernel computation time depends on the parameters defined at launch. The parameters were chosen so that the graphic card occupancy was maximum. Occupancy is a ratio of the executed threads to maximum number of threads per streaming multiprocessor (SM).

Using NVIDIA Visual Profiler we calculated that 236 registers are required to execute a single thread. Consequently, one SM cannot execute more than 277 threads. Since the block size must be divisible by 32, one block cannot contain more than 256 threads. Thus, CUDA occupancy is equal to $256/2048 = 0{,}125$. Since the graphic card has only five SM, the peak graphic card occupancy, in terms of this GPU implementation, is achieved with 5×256 threads. Execution of that many threads provides linear speeding-up. Experimental results are shown in Table 2.

Table 2. 3D coordinates calculation for 12000 × 12000 pairs of corresponding points

ENVI computation time (milliseconds)	800×10^3
CUDA kernel computation time (milliseconds)	1489
GPU memory copy time (milliseconds)	1471
Total GPU computation time (milliseconds)	2960
Speeding-up	270

5. Experimental results

The stereo pairs obtained from satellites IRS-P5 with a spatial resolution of 2.5 meters were chosen as the initial data. Stereopair IRS-P5 (Cartosat-1) was obtained on January 30, 2008. The initial images are shown in Fig. 5.

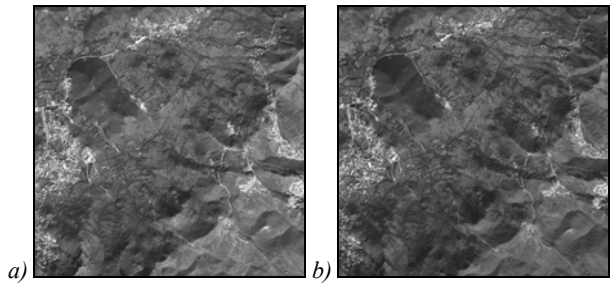

Fig. 5. Initial images: a) left, b) right

Based on the ENVI matching algorithm, the three-dimensional model was built. On the resulting model (Fig. 6*a*)) the mountains and the terrain are seen clearly. Fig. 6*b*) shows a disparity map generated using the proposed hybrid CPU/GPU procedure (Fig. 1).

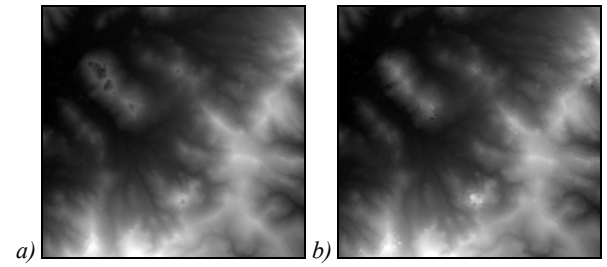

Fig. 6. Disparity map: a) ENVI, b) proposed parallel algorithm

DEMs shown in Fig. 7*a*, *b* were generated for the above mentioned regions of the disparity map. In order to compare these DEMs, the proximity measure offered in paper [20] was calculated. In particular, we calculated elevation difference on the preset significance level 95 % (so-called linear error, LE95) which was equal to 8.64 meters.

Fig. 7 shows a three-dimensional terrain model obtained after image matching.

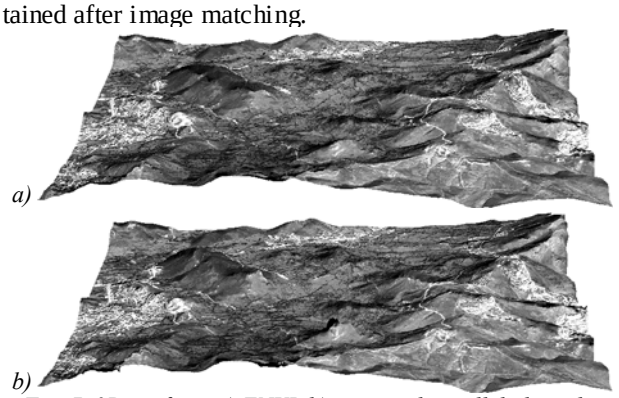

Fig. 7. 3D-surface: a) ENVI, b) proposed parallel algorithm

Table 3 shows the results of comparative studies of the implementation times of the matching algorithm ENVI-5.0 software package and the proposed parallel algorithm on the GPU at different numbers of pyramid levels (the data were obtained using GeForce GTX 750 Ti, and Intel Core i7-6700K, 16GB DDR4, OS Windows 10). Rectified images with the same size of 12000×12000 pixels were used in the experiments.

As it can be seen from the table, the time of CPU/GPU method implementation for the resolution of 219×219 pixels is not given. This is because the pyramid used within the proposed approach is one level less. The implementation time for the CPU/GPU is given in milliseconds. The proposed procedure calculates the corresponding points several orders of magnitude faster than ENVI-5.0 software. This effect is caused by several reasons. Firstly, ENVI is a package comprising a lot of modules for a wide range of applications. Our development is aimed at solving one particular problem of this spectrum. Secondly, ENVI is written in a programming language IDL. Our parallel implementation uses CUDA technology and is written in C++ language. Thus, our development is aimed at the specific task and uses modern technology.

Table 3. Obtained experimental results

Pyramid level	1	2	3	4	5	6	7
Image resolution in pixels	219×219	438×438	875×875	1750×1750	3500×3500	6000×6000	12000×12000
Implementation time of the ENVI method (in milliseconds)	1.1×10^3	1.87×10^3	7.44×10^3	28.24×10^3	112.54×10^3	441.27×10^3	1806.5×10^3
Implementation time of the CPU/GPU method (in milliseconds)	-	6.3	20.2	70.4	252.9	925.3	3390.1

Conclusion

Experimental results demonstrate that the developed general procedure provides the 3D DTM quality comparable to that achieved with the use of the ENVI-5.0 software. However, implementation time of the proposed procedure of the same pair of images with dimensions of 12000×12000 is about 300 times less. Unfortunately, there are currently no representative databases of 3D model test sets with corresponding stereo images. Therefore, further research will be aimed at the development of methods of test images and 3D scene models formation, and the procedure verification in order to obtain objective statistical evaluation of DTM quality.

References

[1] Qayyum A, Malik AS, Muhammad Saad MNB. Comparison of digital elevation models based on high resolution satellite stereo imagery. International Conference on Space Science and Communication (IconSpace) 2015: 203-208.

[2] Pandey P, Venkataraman G. Generation and evaluation of Cartosat-1 DEM for Chhota Shigri Glacier, Himalaya. International Journal of Geomatics and Geosciences 2012; 2(3): 704.

[3] Giribabu D, Rao SS, Murthy YK. Improving Cartosat-1 DEM accuracy using synthetic stereo pair and triplet. ISPRS journal of photogrammetry and remote sensing 2013; 77: 31-43. DOI: 10.1016/j.isprsjprs.2012.12.005.

[4] Fraser CS, Hanley HB. Bias-compensated RPCs for sensor orientation of high-resolution satellite imagery. Photogrammetric Engineering & Remote Sensing 2005; 71(8): 909-915. DOI: 10.14358/PERS.71.8.909.

[5] Grodecki J, Dial G. Block adjustment of high-resolution satellite images described by rational polynomials. Photogrammetric Engineering & Remote Sensing 2003; 69(1): 59-68. DOI: 10.14358/PERS.69.1.59.

[6] Paradella WR, Cheng P. Using Geoeye-1 stereo data in mining application: automatic DEM generation. Geoinformatics 2013; 16: 10-12.

[7] Zhou Y, Parsons B, Elliott JR, Barisin I, Walker RT. Assessing the ability of Pleiades stereo imagery to determine height changes in earthquakes: A case study for the El Mayor-Cucapah epicentral area. Journal of Geophysical Research: Solid Earth 2015; 120(12): 8793-8808. DOI: 10.1002/2015JB012358.

[8] User guide ENVI. Source: ⟨https://www.exelisvis.com/portals/0/pdfs/envi/DEM_Extraction_Module.pdf⟩.

[9] User guide PHOTOMOD. Source: ⟨http://www2.racurs.ru/download/docs/rus/DEM.pdf⟩.

[10] User guide Geomatica. Source: ⟨http://www.pcigeomatics.com/pdf/geomatica/tutorials/Live_DEM_Editing.pdf⟩.

[11] Gomez-Balderas J-E, Houzet D. A 3D reconstruction from real-time stereoscopic images using GPU. Conference on Design and Architectures for Signal and Image Processing (DASIP 2013) 2013: 253-258.

[12] Pollefeys M, Nistér D, Frahm JM, Akbarzadeh A, Mordohai P, Clipp B, Engels C, Gallup D, Kim S-J, Merrell P, Salmi C, Sinha S, Talton B, Wang L, Yang Q, Stewénius H, Yang R, Welch G, Towles H. Detailed real-time urban 3d reconstruction from video. International Journal of Computer Vision 2008; 78(2-3): 143-167. DOI: 10.1007/s11263-007-0086-4.

[13] Kotov AP, Fursov VA, Goshin EV. Technology for fast 3D-scene reconstruction from stereo images. Computer Optics 2015; 39(4): 600-605. DOI: 10.18287/0134-2452-2015-39-4-600-605.

[14] Baltsavias EP, Stallmann D. SPOT stereo matching for DTM generation. Proc SPIE 1993; 1944: 152-163. DOI: 10.1117/12.155800.

[15] Kadomcev BB. Dynamics and the Information. Izbrannye trudy: in 6 volumes [In Russian]. Moscow: "Fizmatlit" Publisher; 2003: 2; 508-515.

[16] Rational Functional Model. Source: ⟨http://geotiff.maptools.org/STDI-0002_v2.1.pdf⟩.

[17] Forsyth DA, Ponce J. Computer vision: A modern approach. Prentice Hall Professional Technical Reference; 2002. ISBN: 0-130-85198-1.

[18] Hartley RI. Theory and practice of projective rectification. International Journal of Computer Vision 1999; 35(2): 115-127. DOI: 10.1023/A:1008115206617.

[19] Fursov VA, Goshin EV. Information technology for digital terrain model reconstruction from stereo images. Computer Optics 2014; 38(2): 335-342.

[20] Jacobsen K. DEM generation from high resolution satellite imagery. Photogrammetrie-Fernerkundung-Geoinformation 2013; 5: 483-493.

Authors' information

Vladimir Alekseyevich Fursov is Doctor of Engineering Science, Professor, head of Supercomputers and General Informatics sub-department of Samara University, leading researcher. Research interests are development of the theory of estimation on small number of observations, development of methods of image processing and training to pattern recognition, development of high-performance parallel methods both algorithms of image processing and pattern recognition oriented on application of multiprocessor computing systems. E-mail: *fursov@ssau.ru* .

Yegor Vyacheslavovich Goshin, Candidate of Engineering Sciences. Research interests are image processing, recognition algorithms, parallel computations and stereovision. E-mail: *goshine@yandex.ru* .

Anton Petrovich Kotov, Master of Applied Mathematics and Computer Science. Currently studies at Samara University. Research interests are image processing, recognition algorithms, 3D-scene reconstruction, and parallel computations. E-mail: *antonykotov@gmail.com* .

Code of State Categories Scientific and Technical Information (in Russian – GRNTI)): 28.23.15, 50.41.25, 89.57.35.
Received September 17, 2016. The final version – October 31, 2016.