The algorithm for a video panorama construction and its software implementation using CUDA technology

I.A. Kudinov[1], O.V. Pavlov[1], I.S. Kholopov[1,2], M.Yu. Khramov[1]

[1]*Ryazan State Instrument-making Enterprise, Seminarskaya str. 32, 390000, Ryazan, Russia*
[2]*Ryazan State Radio Engineering University, Gagarina str. 59/1, 390005, Ryazan, Russia*

Abstract

A video panorama constructing algorithm based on information from five different types pre-calibrated cameras with partially overlapping fields of view was developed and implemented using the CUDA C language. Distortion compensation, image stitching on the virtual unit sphere surface, and blending procedures are performed for the operator-controlled 1024×768 pixels region of interest with 50 fps.

Keywords: video panorama; camera calibration; distortion compensation; spherical panorama; region of interest; inclinometer; blending; CUDA technology

1. Introduction

The automatic generation of high-resolution video panoramas from information of cameras with partially overlapping fields of view (FoV) is one of the modern trends in the vision systems development. Generally, panorama navigation implies the presence of a user-controlled region of interest (RoI) with the predefined angular FoV dimensions and resolution [1]. In avionics, for example, the advantages of panorama systems in comparison with traditional electro-optical systems are [2, 3], at first, the possibility of simultaneous use of a panorama field by several independent operators, and, secondly, the absence of mechanical parts.

2. Problem statement

A panoramic image from N frames with overlapping (or pairwise overlapping) FoVs is formed by finding a correspondence between the pixel coordinates of each frame. This correspondence for the camera frames with the numbers i and j is determined by the 3×3 dimension homography matrix $\mathbf{H}_{ij}$ [4]:

$$\mathbf{x}_i = \mathbf{H}_{ij}\mathbf{x}_j, \tag{1}$$

where the matrix transformation (1) performs the recalculation of the j-th camera image homogeneous pixel coordinates into the i-th camera coordinate system, $\mathbf{x}_j = [u_j, v_j, 1]^T$ and $\mathbf{x}_j = [u_i, v_i, 1]^T$ are homogeneous pixel coordinates of i-th and j-th images, and (u, v) are coordinates of pixel which is located at the intersection of u-th row and v-th column.

There are several approaches to the panorama construction. In the absence of a priori information, the estimation of the homography matrix $\mathbf{H}_{ij}$ is based on m interest points (IP) detection, building descriptors of the neighborhood for each IP and their automatic matching. In this case homography matrix $\mathbf{H} = [[h_1, h_2, h_3]^T, [h_4, h_5, h_6]^T, [h_7, h_8, 1]^T]$ estimation is performed by pseudosolution (in terms of minimum mean square error) of overdetermined system of equations for $m \geq 4$ inliers:

$$\mathbf{A}_{(2m \times 9)}\mathbf{h}_{(9x1)} = \mathbf{0}_{(2m \times 1)},$$

$$\mathbf{A}_{(2 \times 9)} = \begin{bmatrix} -u_j & -v_j & -1 & 0 & 0 & 0 & u_i u_j & u_i v_j & u_i \\ 0 & 0 & 0 & -u_j & -v_j & -1 & v_i u_j & v_i v_j & v_i \end{bmatrix}. \tag{2}$$

The pseudosolution of (2) is the last column-vector of matrix $\mathbf{V}$ (which is the result of the matrix $\mathbf{A}_{(2m \times 9)}$ singular value decomposition) corresponding to the minimal singular value Σ_{min}:

$$\mathbf{A}_{(2m \times 9)} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \ \mathbf{h} = \mathbf{V}^{<\Sigma min>}.$$

The pay for the universality of this approach to panorama construction is a low efficiency on homogeneous surfaces (grass, arable land, forest, water surface, sky), and under low contrast conditions, and when the overlapping of cameras FoVs is small.

With priori information about the mutual position of the camera's coordinate systems obtained during the preliminary calibration, the homography matrix can be estimated by the formula:

$$\mathbf{H}_{ij} = \mathbf{K}_i[\mathbf{R}_{ij} - \mathbf{t}_{ij}\mathbf{n}^T/d]\mathbf{K}_j^{-1}, \tag{3}$$

where $\mathbf{K}_i$ and $\mathbf{K}_j$ are intrinsic matrices, $i, j = 1..N$, $\mathbf{R}_{ij}$ and $\mathbf{t}_{ij}$ are respectively a rotation matrix and translation vector for transition from j-th camera coordinate system to i-th camera coordinate system, d – the perpendicular length to the shooting

plane with the normal **n** in the reference (*i*-th) camera coordinate system. If the distance to the observed objects is large ($\|\mathbf{t}_{ij}\| << d$), then we have the following approximate equality from (3):

$$\mathbf{H}_{ij} \approx \mathbf{K}_i \mathbf{R}_{ij} \mathbf{K}_j^{-1}. \quad (4)$$

While combining information from several cameras, it is advisable to form the resultant panoramic image not in the plane in accordance with (1), but on a virtual uniform curvature surface: usually a unit sphere or a cylinder with a unit radius [5]. It allows us to work with normalized homogenous pixel and spatial coordinates. The geometric problem statement for combining $N = 3$ camera frames with intersecting FoVs on the unit sphere surface (the coordinate system of camera with number 0 is taken as reference) is shown at Fig. 1. It is expected that by analogy with (4) all nodal points of the camera lenses are located in the unit sphere center *O*.

In the Fig. 1 the symbol **M** denotes the point of the object spatial homogeneous coordinates, which image is projected onto the point on the unit sphere surface with spatial coordinates $\|\mathbf{M}_{sph}\| = 1$ in the coordinate system of the sphere $OX_{sph}Y_{sph}Z_{sph}$, and the symbols $\mathbf{x}_0$ and $\mathbf{x}_2$ are the homogeneous pixel coordinates of its image on the frames of cameras with numbers 0 and 2 respectively.

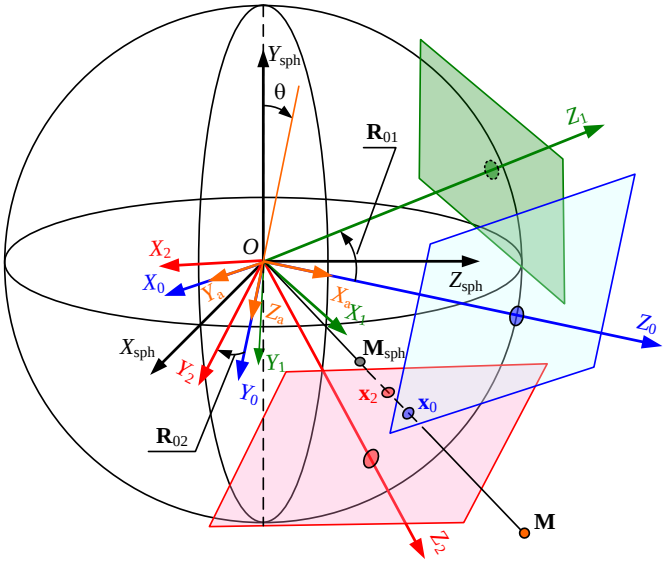

Fig. 1. Unit virtual sphere for panorama forming.

The correct spherical panoramas construction implies the availability of information about the angular position of the optical axes of the cameras in relation to the horizon plane. This problem is solved in our work by mounting of inclinometer (based on the triaxial MEMS accelerometer) on the reference camera case. An accelerometer coordinate system is designated as $OX_aY_aZ_a$ (Fig. 1). The roll φ and pitch θ angles are estimated by the formulas [6]:

$$\varphi = \operatorname{atan2}(a_y, a_z), \quad \theta = \operatorname{atan2}[-a_x, (a_y^2 + a_z^2)^{0,5}], \quad (5)$$

where $[a_x, a_y, a_z]^T$ – a vector of accelerometer measurements (taking into account its calibration [7]).

The choice of pixels in areas where the radius-vector $\mathbf{OM}_{sph}$ crosses several camera planes can be fulfilled by several criteria: for example, the maximum distance from the pixel to the intersection line of the camera planes or the minimum length of the normalized pixel coordinates vector. In order to minimize computations we choose a criterion of minimum angle between a vector to a pixel and the *i*-th camera principal axis, as such pixels, usually, have the minimal distortion correction error:

$$\min_i \left[\left(\mathbf{R}_{0i} \mathbf{R}_{\varphi\theta} \right)^{<3>} \cdot \mathbf{M}_{sph} \right], \quad (6)$$

where

$$\mathbf{R}_{\varphi\theta} = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} - \quad (7)$$

is a rotation matrix of reference camera coordinate system relative to the horizon plane, $\mathbf{R}_{0i}$ – estimated during calibration rotation matrix of *i*-th camera relative to the reference camera, $^{<k>}$ – *k*-th column of matrix, and symbol «·» is a dot product.

To realize the sliding RoI function with size $W \times H$ pixels with angular dimensions in the horizontal and vertical directions $\Delta\varphi_w$ and $\Delta\varphi_h$ respectively we introduce a quaternion $\mathbf{q}_{\text{vis}}$ [8] that is defined by the current line of sight azimuth α and the elevation angle β:

$$\mathbf{q}_{\text{vis}} = [\cos(\alpha/2)\cos(\beta/2),\ \cos(\alpha/2)\sin(\beta/2),\ \sin(\alpha/2)\cos(\beta/2),\ \sin(\alpha/2)\sin(\beta/2)]^{\text{T}}. \quad (8)$$

To each pixel of the RoI with coordinates (u, v) corresponds the point $\mathbf{M}_{uv} = [x_{uv}, y_{uv}, z_{uv}]^{\text{T}}/\|[x_{uv}, y_{uv}, z_{uv}]^{\text{T}}\|$ on the unit sphere (Fig.2), determined by the radius-vector with the corresponding quaternion

$$\mathbf{q}_{uv} = \mathbf{q}_{\text{vis}}\mathbf{q}_{uv0}, \quad (9)$$

where from geometric constructions of Fig. 2 the initial position ($\alpha = \beta = 0$, $z_{uv} = 1$) of the RoI pixels (u, v) corresponds to quaternion

$$\mathbf{q}_{uv0} = [\cos(\alpha_u/2)\cos(\beta_v/2),\ \cos(\alpha_u/2)\sin(\beta_v/2),\ \sin(\alpha_u/2)\cos(\beta_v/2),\ \sin(\alpha_u/2)\sin(\beta_v/2)]^{\text{T}}, \quad (10)$$

$$\alpha_u = \text{arctg}(x_{uv}), \quad \beta_v = \arcsin[y_{uv}/(x_{uv}^2 + y_{uv}^2 + 1)^{0,5}],$$

$$x_{uv} = (2u/W - 1)\text{tg}(\Delta\varphi_w/2), \quad y_{uv} = -(2v/H - 1)\text{tg}(\Delta\varphi_h/2).$$

Quaternion $\mathbf{q}_{uv}$ allows us to determine coordinates of the point $\mathbf{M}_{uv}$ in the unit sphere coordinate system [8]:

$$\mathbf{M}_{uv} = [2(q_x q_z + q_w q_y),\ 2(q_y q_z - q_w q_x),\ q_w^2 + q_z^2 - (q_x^2 + q_y^2)]^{\text{T}}, \quad (11)$$

where q_w and $[q_x, q_y, q_z]^{\text{T}}$ are scalar and vector parts of quaternion $\mathbf{q}_{uv}$.

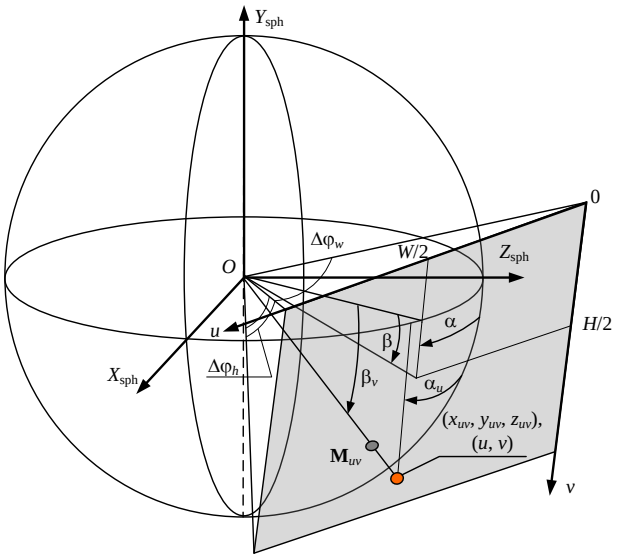

Fig. 2. The relationship between the angular and spatial coordinates of the point $\mathbf{M}_{uv}$ on the virtual unit sphere surface.

On the each i-th camera principal plane the homogenous pixel coordinates $\mathbf{x}_{uvi}$ corresponding to the point $\mathbf{M}_{uv}$ are determined (taking into account the orientation relative to the horizon plane) through the projection matrix $\mathbf{P}_i$ [4]:

$$\mathbf{x}_{uvi} = \mathbf{P}_i\mathbf{M}_{uv}, \quad (12)$$

where $\mathbf{P}_i = \mathbf{K}_i[\mathbf{R}_{0i}\mathbf{R}_{\varphi\theta} \mid \mathbf{0}]$, $\mathbf{0} = [0, 0, 0]^{\text{T}}$, and the selection of the pixel transferred from the i-th camera plane to the RoI is performed by the criterion (6).

For the cameras with wide-angle lenses it is necessary to do the distortion compensation: for the normalized coordinates with distortion

$$\mathbf{x}_{\text{н}i} = \mathbf{K}_i^{-1}\mathbf{x}_{uvi} \quad (13)$$

coordinates without distortion are estimated according to the Brown-Conrady model [9] (in order to reduce the amount of calculations in our work only the first two coefficients of radial distortion are used). Corrected pixel coordinates without distortion are calculated by multiplying on the intrinsic matrices $\mathbf{K}_i$:

$$\mathbf{x}_{corri} = \mathbf{K}_i \mathbf{x}_{ni} \quad (14)$$

and since it is fractional so the brightness value is interpolated (in our work we used a bilinear interpolation [10]).

Since the scenes of each camera are different, then in the automatic exposure time mode the average brightness of the composing panorama frames is different too. Therefore, after panorama filling (or the RoI filling) it is additionally necessary to perform a smoothing procedure for brightness differences – blending. We used a simplified approach to blending, similar to the work [11].

3. The algorithm for a video panorama construction and its software implementation

The algorithm for forming the RoI image contains the following steps.

1. Initialization: calculating quaternions $\mathbf{q}_{uv0}$ by formula (10).

Main operation cycle:

2. Estimation of current angular position of reference camera by (5) and its rotation matrix $\mathbf{R}_{\varphi\theta}$ by (7).
3. Quaternion $\mathbf{q}_{vis}$ calculation for the current line of sight angular position by (8) and quaternions $\mathbf{q}_{uv}$ by (9).
4. Filling the RoI (with distortion compensation) according to (11)-(14) by criterion (6).
5. Blending procedure performing (optional).

As the processing for each RoI pixel is homogeneous this allows us to parallelize computations. In the layout based on a PC for implementing the algorithm steps 1, 4, and 5 we used the resources of the NVIDIA GPU: using CUDA technology and CUDA C language the whole amount of computations was distributed to 3072 parallel blocks (64 horizontally and 48 vertically) with 256 threads in each block (16 horizontal and vertical threads). In this case, according to [12], the pixel indices are represented as:

$$u = \text{blockIdx.y} * \text{blockDim.y} + \text{threadIdx.y}; \quad v = \text{blockIdx.x} * \text{blockDim.x} + \text{threadIdx.x}; \quad (15)$$

In (15) the following standard CUDA notation is used: blockIdx is a block identifier (number), blockDim – a block dimension, threadIdx – an identifier (number) of a parallel thread, and attributes "x" and "y" indicate the axis of the coordinate system in a two-dimensional Euclidean space.

As the copying from CPU memory to GPU memory and back is rather slow [12], so by the implementing of the algorithm the number of such operations is minimized: by the initialization GPU memory arrays are recorded with the data on the camera parameters (intrinsic matrices and distortion coefficients) and initial quaternions $\mathbf{q}_{uv0}$ (10) that correspond to RoI pixels. In the main operation cycle frames from each camera, the line of sight angular coordinates and the angular coordinates of reference camera, estimated from MEMS signals, are copied into the GPU memory, then the steps 3-5 of the algorithm are performed and the result (array of brightness values in the RoI) is copied back to the CPU memory for displaying on the PC monitor.

4. Results and Discussion

The layout of the video panorama system is implemented on a PC with Intel Core-i5 processor and NVIDIA GeForce GTX 560 Ti GPU (with 384 cores) and consists of five digital GigE interface video cameras, mounted on a special rigid polyamide bracket (Fig. 3): two Basler acA2000 cameras with 2048×1088 frame resolution and 5 mm megapixel Cowa lenses (from below) and three IDS 5240 RE cameras (from above) with 1280×1024 frame resolution and 5 mm megapixel Computar lenses (two outer cameras) and 8 mm Navitar lens (central reference camera), an evaluation board with a InvenSense MPU 9250 MEMS sensor mounted on the reference camera, two motorized rotation positioners Standa 8MR190-2 for moving the bracket with cameras in horizontal and vertical planes, and USB joystick Defender Cobra M5 for the RoI navigation.

Fig. 3. General view of the video panorama forming layout.

The layout allows to create a panorama with a 3600×2400 pixels resolution and 180°×120° FoV or display RoI with user-defined resolution and angular FoV dimensions.

Preliminary camera calibration was performed using OpenCV libraries [13]: on 40 test "chessboard" pattern frames, containing 12×11 cells with a side of 3 cm, the intrinsic matrices and distortion coefficients of their lenses were estimated; then on 40 test "chessboard" pattern frames, containing 9×6 cells with a side of 3 cm, the rotation matrices of cameras relative to the reference camera were estimated.

The influence of the angular orientation estimation error of the reference camera relative to the horizon plane on the geometry of the panorama is shown in Fig. 4. As can be seen from the figure, the pitch estimation error leads to an incorrect horizon display (the position of the spherical panorama equator is shown by the orange line).

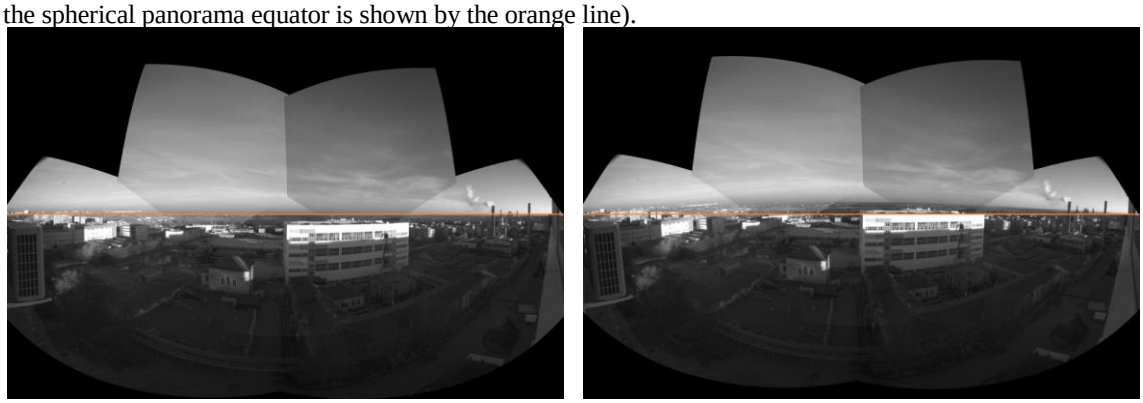

Fig. 4. The influence of the pitch estimation error on the spherical panorama geometry (without blending): left – the pitch of the reference camera coincides with the true, $\theta = 15°$, right – the pitch of the reference camera doesn't coincide with the true, $\theta = 12°$.

The results of the 1024×768 pixels RoI forming with a 40°×30° FoV and the line of sight angular coordinates $\alpha = 15°$ and $\beta = –5°$ with blending and without it are shown in Fig. 5 and 6 respectively. The times required to create one frame on the CPU and GPU resources are summarized in Table 1 (the time for copying the data from the CPU memory to the GPU memory and back is 3.5 ms), and the times for different RoI sizes are summarized in Table 2.

Table 1. Time required for the 1024×768 pixels RoI forming, ms.

	CPU	GPU with CUDA	Gain
Without blending	114	6.5	17.5
With blending	272	17.2	15.8

Table 2. Time required for the RoI forming on GPU with CUDA for different resolution, ms.

RoI resolution, pixels	800×600	1024×768	1280×1024
Without blending	5.1	6.5	9.7
With blending	12.9	17.2	23.4
CPU ↔ GPU copy time	2.9	3.5	3.7

In Fig. 7 and 8 are shown the results of intermediate computations for blending implementation: the boundaries of the camera frames in the RoI (Fig. 7) and the 100-pixel width binary mask, over which the brightness is smoothed (Fig. 8).

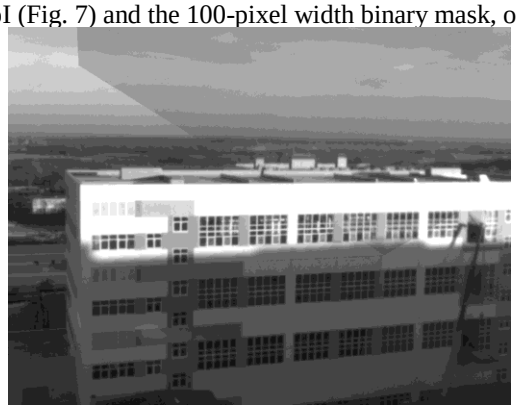

Fig. 5. RoI without blending.

Fig.6. RoI with blending.

Increasing of the RoI forming time with blending more than 2 times is explained by the need to perform auxiliary procedures: searching the intersection lines of camera frames in the RoI, determining areas for brightness fusion and smoothing images to estimate the low-frequency component of brightness in the each camera frame. To reduce the amount of calculation, we apply a smoothing procedure with an 8×8 window and an accumulator. Its computational complexity is $O(n^2)$.

For $\|\mathbf{t}_{0i}\| < 15$ cm and distance to objects $d > 70$ m the hypothesis about camera lenses nodal points superimposition provides the error of stitching on the camera frame boundaries not more than 5 pixels.

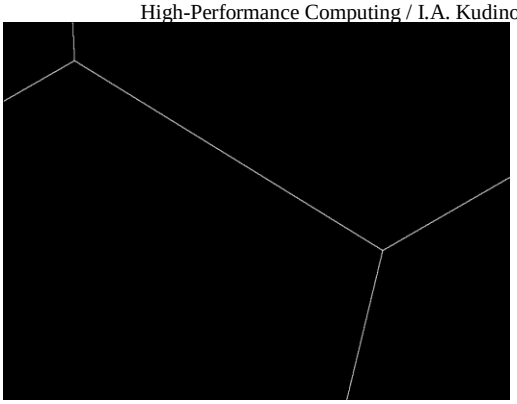

Fig. 7. The boundaries of the camera frames in the RoI (binary image).

Fig. 8. Blending binary mask obtained from Fig. 7.

The results of the experiment showed that the use of GPU resources makes it possible to reduce the RoI forming time up to 16 times in comparison with the CPU (when only one CPU core is used).

5. Conclusion

The algorithm for a video panorama construction, designed and implemented with the CUDA technology and parallelization of computations on a GPU, for a one megapixel resolution region of interest provides panorama navigation with 50 fps on the whole 8.2 megapixels panoramic video frame with 180°×120° field of view.

References

[1] Banta B, Donaldson G. Apparatus and method for panoramic video hosting. Patent US US9516225, 2016.
[2] AN/AAQ-37 Distributed Aperture System (DAS) for the F-35. URL: http://www.northropgrumman.com/Capabilities/ ANAAQ37F35/Pages/ default.aspx (01.10.2016).
[3] IronVision™. URL: http://elbitsystems.com/media/IronVision.pdf (10.09.2016).
[4] Hartley R, Zisserman A. Multiple view geometry in computer vision. 2nd edition. Cambridge: Cambridge University Press, 2003; 656 p.
[5] Szeliski R. Image alignment and stitching: a tutorial. Foundations and trends in computer graphics and vision 2006; 2(1): 1–104.
[6] Tilt sensing using a three-axis accelerometer. URL: http://www.freescale.com/files/sensors/doc/app_note/AN3461.pdf (11.04.2016).
[7] Wang L, Wang F. Intelligent calibration method of low cost MEMS inertial measurement unit for an FPGA-based navigation system . International J. of Intelligent Engineering and Systems 2011; 4(2): 32–41.
[8] Kuipers JB. Quaternions and rotation sequences. New Jersey: Princeton University, 1998; 400 p.
[9] Brown DC. Close-range camera calibration. Photogrammetric Engineering 1971; 37(8): 855–866.
[10] Parker JA, Kenyon RV, Troxel DE. Comparison of interpolating methods for image resampling. IEEE Trans. on Medical Imaging 1983; 2(1): 31–39.
[11] Xiong Y, Pulli K. Fast image stitching and editing for panorama painting on mobile phones. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 13-18 June 2010: 47–52.
[12] Sanders J, Kandrot E. CUDA by example. New York: Addison-Wesley, 2010; 290 p.
[13] Camera Calibration and 3D reconstruction. URL: http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html (14.03.2015).