Seam Line Optimization for Remote Sensing Image Mosaicking Based on Graph Shortest Path Finding

Andrew Sudorgin
Joint-Stock Company Research Institute of Precision Instruments (JSC RI PI)
Moscow, Russia
andrew.sudorgin@gmail.com

Abstract

Seam line determination is important step of remote sensing image mosaicking. Automatic seam line selection problem using graph shortest path finding is discussed in this paper. Approaches to graph construction from image similarity cost function, graph memory layout and parallel seam line optimization using shortest path on graph are presented.

Keywords: remote sensing data processing, image mosaic, seamless coverage, optimization

1 Introduction

Acquisition of holistic image of a certain territory in the form of a mosaic from various (in the sense of geometry and brightness-contrast) separate images is an important problem when processing satellite and aerial photos. Wide range of remote sensing equipment with unique spectral sensitivity, variation of angle of view, position of the Sun and atmosphere condition, non-simultaneous image acquisition cause unique deformation of the image of object of interest. Such a large number of negative factors means that it is practically impossible to compose the images, both from a geometric point of view, and from the point of view of radiometric conditions. One of the solutions of the described problem when seamless mosaic is created is optimal seam line detection within overlapping image regions.

To obtain a visually qualitative result that does not introduce distortions into the information, strict and, in some sense, contradictory restrictions on the behavior of the seam line are superimposed in the construction of photographic plans and mosaic of images: a seam line must not cross high-altitude objects, seam line must cross linear objects perpendicularly and should be inside low-informativeness areas [1]. It is not easy task to satisfy all those criteria completely, especially when editing by hand.

2 Related Work

There are some different approaches for seam line creation:

- Draw seam line along image borders (no optimization);
- Central image area maximization;
- Draw seam line inside areas with maximum similarity.

Draw seam line along image borders is used when there are no quality demands to resulting image mosaic, or there are strict time limitations. Since brightness and geometrical distortions of remote sensing images are minimal in their central areas we can draw seam line as close to image center as possible using Voronoi polygons [2]. This approach is helps to avoid of image margin brightness inequality and to include central areas with minimal geometry distortions to final mosaic. However, the first two approaches completely do not take into account the properties of the input images.

Optimal seam line detection methods are based on the search of the pixels with minimal differences in brightness, gradient and some other parameters which are calculated from the image overlap regions (intersections of image pairs in coordinate system of an output image). Seam line being determined is a base for united mosaic from several initial images. In this case the visual divergence is significantly eliminated. Many approaches bring optimal seam line determination problem to energy function minimization. In

In: V. Voevodin, A. Simonov (eds.): Proceedings of the GraphHPC-2017 Conference, Moscow State University, Russia, 02-03-2017, published at http://ceur-ws.org.

some articles normalized cross correlation (NCC) [3,4], texture analysis [6, 7], image differences [6], morphological analysis [8] are suggested as a energy function. In order to create energy cost function some authors also use vector roads graph [9] and digital surface model [10]. The following algorithms are used as optimization: snake model [6, 11], Dijkstra's algorithm [3,4,5], Floyd-Warshall algorithm, dynamic programming, graph cuts [12, 13].

Even though in practice, graph cuts algorithms have computational complexity close to linear [14, 15] and there are parallel approaches that allow processing graphs that are larger than the available RAM [16], they remain relatively slow, since a much larger amount of RAM is required for the solution. For example, one of the fast realizations of the graph cut algorithm requires memory several times more than the graph itself takes, since it is required to store and process not only edge weights, but also forward/backward edge capacity and other auxiliary values [17].

Due to the fact that the solution of optimization problems on graphs is associated with high computational complexity, a number of authors try to simplify the structure of the graph, leading to a reduction in the time of solving the problem at the cost of a slight degradation in quality [18].

The main contribution of this paper is a new approach to seam line optimization with low memory consumption and a high degree of parallelism.

3 Approach to Seam Line Optimization

To achieve a visual invisibility of the seam line the value of the pixels of overlapped images must coincide at points with the same coordinates in geographic coordinate system (or coordinate system of output image for non georeferenced mosaic). Seam line optimization is one of steps of remote sensing image mosaicking. Normally, it follows after geometry alignment and radiometric correction of input images. We expect that these two steps have already been correctly done, but not ideally due to some reasons and we need to eliminate the remaining visual inconsistencies.

3.1 Energy Cost Function

In this article to create optimal seam line we propose to build energy cost function comprised of three functions: the image similarity matrix W_s, the image informativeness matrix W_i and the forbidden zone matrix (linear extensive, high-altitude and other visually obvious foreground objects). The similarity matrix W_s is calculated as follows:

$$W_s(x, y) = (I_0(x, y) - I_1(x, y))^2 \quad (1)$$

where $I_0(x, y)$ and $I_1(x, y)$ denote the brightness on the first and second overlapping images in the (x, y) position (here and later we assume that (x, y) is a pixel coordinates (row/column) of input images in coordinate system of output mosaic and all geometry transformations, including nonlinear ones, already have been done at geometry align step).

To define informativeness matrix W_i Moravec [19] algorithm is used. According to this algorithm, we calculate the changes in the average brightness value of the image of a small fragment around the point of interest when the fragment is shifted by one pixel in four directions (horizontal, vertical and diagonal) and choose among them the minimal value, which is the measure of the informativeness of the image. Thus, fragments of the image with a small brightness variation are marked as low-informative, and with high variation, as highly informative. Total informativeness $W_i(x, y)$ is the sum of each overlapping image informativeness. By drawing a seam line in places with a low value of the Moravec's operator of interest, we will satisfy the requirement [1] of carrying out the seam line in places of image with low informativeness.

In order for the seam line to cross the line objects along a line close to the perpendicular and not intersect the high-altitude objects, we introduce the matrix of forbidden zones. To create forbidden zone matrix $W_r(x, y)$ we rasterize vector lines and polygons, which represent linear extensive and high-altitude objects, using specially defined penalty coefficients for crossing those objects. Due to the fact that the seam line receives an additional penalty when crossing objects marked in matrix $W_r(x, y)$, the optimization algorithm will try to bypass such objects or cross them along the line of the minimum possible length, that for thin elongated objects (roads, rivers) will be perpendicular.

Resultant energy function defined as a weighted sum of the particular matrices:

$$W(x, y) = \alpha W_s(x, y) + \beta W_i(x, y) + \gamma W_r(x, y), \quad (2)$$

where α, β, γ are the weighting coefficients, which defined each matrix contribution. In most cases, we can choose all weights to be the same, but if one wants to amplify one of the constraints, the corresponding coefficient can be increased. For example, if the visual similarity of images is more important to us, then we increase α, and if it is more important for us to draw the seam line in places with low informativeness, then we increase the β coefficient, if we want to completely prohibit the intersection of linear and high-altitude objects, we can set the coefficient γ close to infinity.

Thereby, seam line, which was delineated along the path with minimal energy cost sum, meet the seam line principal criteria: do not intersect high-altitude objects, linear objects should be intersected along the

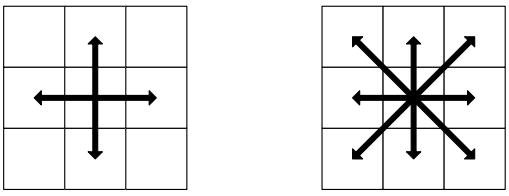
shortest distance (perpendicularly), and the other object should be intersected in the areas with maximum similarity and minimum informativeness.

3.2 Graph Data Structure Design

For optimal seam line determination we define energy function as weighted graph. Each node of the graph represents corresponding energy function matrix element. Let us connect all graph nodes to their neighbors, here we have two options: four- and eight- connected graph (Figure 1). We define the weight of the edge between energy matrix cells (x_1, y_1) and (x_2, y_2) as the sum of weights in the corresponded nodes multiplied by the distance between them:

$$\omega(x_1y_1 \to x_2y_2) = \\ [W(x_1, y_1) + W(x_2, y_2)]\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (3)$$

Thereby energy function matrix of $M \times N$ size is represented by flat graph consisted of MN nodes and $P_{conn}MN$ edges (where P_{conn} is a connection amount of energy matrix elements). Such a construction of the graph allows us not to store the graph explicitly, but to build it as necessary from the energy matrix on the fly.



a) Four-connectivity b) Eight-connectivity

Figure 1: Four- and eight-connectivity of the pixels used to create energy function graph in the overlapped image regions

Let us consider in more detail. The amount of memory required to store the energy function matrix $W(x, y)$ is:

$$S_w = S_eMN, \quad (4)$$

where S_e is the size of one matrix element. Majority of the Earth remote sensing photos are 16 bit per channel images, therefore it is reasonable to use similar approach to energy matrix storage data type ($S_e = 2$). The amount of memory required to store the graph explicitly in the form of compressed sparse row matrix (CSR) is:

$$S_{csr} = S_{Ptr}MN + S_{Ptr}P_{conn}MN + S_eP_{conn}MN, \quad (5)$$

where first term is a memory amount to store graph nodes, second term is memory amount to store graph edge pointers and third term is memory for edge weights, S_{Ptr} is node/edge pointer size (usually 4 or 8 byte), S_e is size of edge weight.

When we store and use the graph in the described implicit way, it is possible to reduce memory usage by $\frac{S_{Ptr}+S_{Ptr}P_{conn}+S_eP_{conn}}{S_e}$ times. When 8 byte pointers and eight- connected graph are used, it is possible to reduce the RAM usage by a factor of 44 in comparison with the CSR graph storage type.

With this approach to storing the graph and when combined with the block method of storing the energy matrix $W(x, y)$ in memory, the processor cache is more efficiently used, since when a graph is traversed, most nodes connected by edges are in the adjacent memory cells.

Moreover, the block storage of the energy matrix $W(x, y)$ allows us to calculate and load matrix blocks as we use at the optimization algorithm, which further reduces the algorithm's requirement for the amount of memory and allows us to process images larger than the available RAM.

3.3 Hierarchical Processing

Typical satellite image size is about 50000×100000 pixels. When seam line between two images is created, we deal with the graph of 10^9 nodes and 10^{10} edges.

To avoid processing the entire array of data, we apply a hierarchical approach to image processing.

In the first step, we construct optimal seam line at the overview level of the original images (Figure 2). To do this, we calculate the coordinates of the points of intersection of the image frames in the coordinate system of the output image (V_b and V_e), read input images from the overview levels of detail, build a coarse energy matrix and run the optimization process between the graph nodes corresponding to the points of intersection of the image frames. As a result, we obtain a coarse approximation of the seam line (blue polyline with black vertices). In the case where the image frames are not convex polygons and the intersection result is a multipolygon, each part of the multipolygon is processed independently.

In the second step, we will refine the seam line between the vertices of the seam line obtained on an overview scale. To do this, we construct the fine energy matrix using fragments of input images from the detailed level and perform the optimization process again, taking as the initial and final nodes of the graph, those that correspond to the vertices of the coarse seam line. Note that to carry out the optimization, it is not necessary to build the entire energy matrix $W(x, y)$, but only the part that is near the coarse approxima-

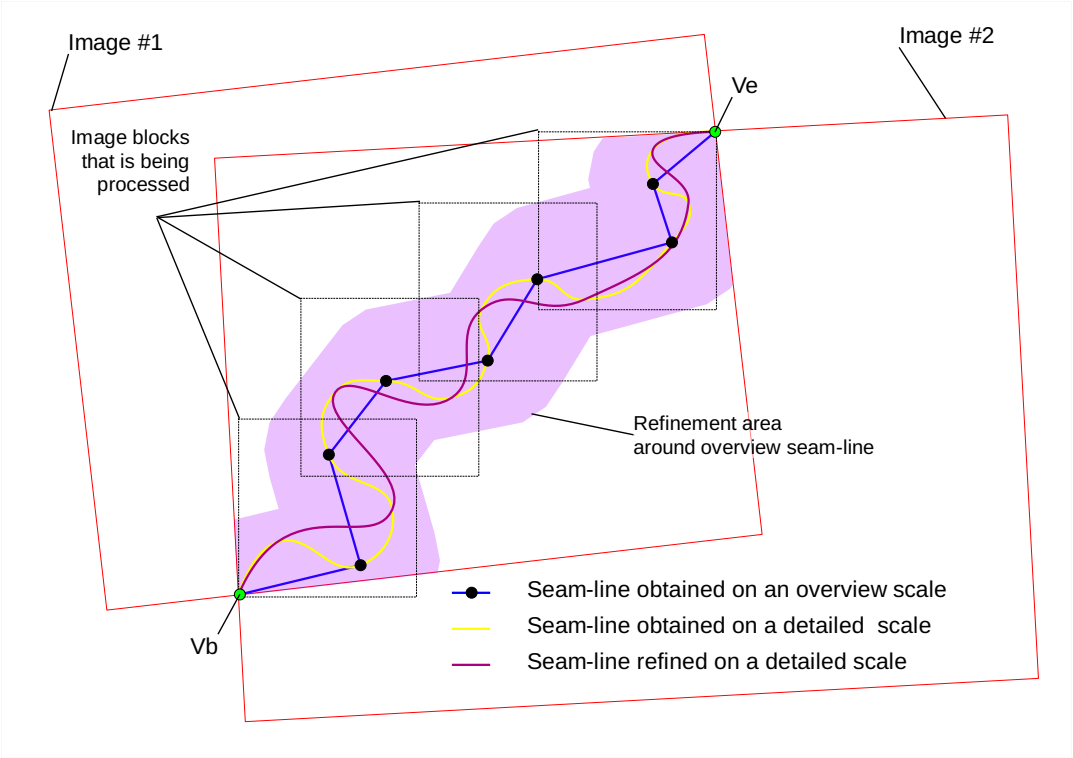


Figure 2: Optimal seam line determination for two overlapped images

tion of the seam line. Moreover, each image fragment can be processed in parallel.

But the new seam lime will have fractures near the vertices of the coarse seam line, because end point vertices of seam lime, are rigidly fixed. To eliminate unwanted fractures, we will carry out the third stage of the seam line refinement on a detailed scale. To do this, we will take as a initial and final nodes of the graph, those that correspond to the midpoints of the segments, obtained in the second step, and we will perform optimization again. Thus, we obtain a seam line, unnoticeable both at the detailed and at the overview scale (Figure 2).

3.4 Optimization

For optimization we use the shortest path Dijkstra's algorithm [20] with the Goldberg modification [21]. The best implementation of Dijkstra's shortest path algorithm with Fibonacci heap has a running time $O(m + nlog(n))$, where n is graph node count, m is graph edge count. Goldberg's shortest path algorithm with multi-level bucket has an average running time that is linear, and a worst-case time of $O(m+nlog(C))$ where C is a ratio of the biggest graph edge weight to the smallest nonzero graph edge weight. Since the graph constructed from the energy matrix has a restriction on the maximum weight of the edge, this gives us a significant advantage in the running time.

Data: $W(x, y)$, V_b, V_e
Result: S - optimal seam line vertices
$D.fill(\text{inf})$; $P.fill(\text{inf})$;
$D[V_b] \leftarrow 0$;
$Q_{MLB}.insert(0, V_b)$
while Q_{MLB} *not empty* **do**
  $c \leftarrow q.extractMin()$
  for $p = 0; p < P_{conn}; ++p$ **do**
    $n \leftarrow c + A_p$
    // Compute edge weight according to (3)
    $\omega \leftarrow (W(c_x, c_y) + W(n_x, n_y))length(A_p)$
    $d \leftarrow D(c_x, c_y) + \omega$
    if $d < D(n_x, n_y)$ **then**
      if $P(n_x, n_y)$ *is not set* **then**
        $q.insert(d, n)$
      else
        $q.decreaseKey(d, n)$
      end
      $D(n_x, n_y) \leftarrow d$
      $P(n_x, n_y) \leftarrow p$
    end
  end
end
$S \leftarrow turnBack(P, V_e)$

Algorithm 1: Pseudo code of the optimization algorithm taking into account structure of the graph

Implementation details taking into account the proposed structure of the graph are given in the Algorithm 1. The input data for the algorithm are energy function matrix $W(x, y)$ and V_b, V_e – start and end points. Note that the coordinates of the corresponding mosaic pixels are used as pointers to the nodes of the graph. In addition to the input data, the algorithm uses several important internal structures. $D(x, y)$ is distance the matrix from the start node to each other. The matrix $D(x, y)$ must have a data type that allows storing the sums of the elements of the matrix $W(x, y)$ without overflowing. If we assume that the size of the element of matrix $W(x, y)$ is equal to two bytes, then the size of the elements of matrix $D(x, y)$ must be equal to four or eight bytes. A is a possible arc list, for four-connectivity case $A_4 = [\leftarrow, \rightarrow, \uparrow, \downarrow]$, and for eight-connectivity case $A_8 = [\leftarrow, \rightarrow, \uparrow, \downarrow, \nearrow, \searrow, \nwarrow, \swarrow]$, according to Figure 1. $P(x, y)$ is node's parent matrix. Since the graph has an ordered structure and each node has four or eight neighbors, we can use the arc number in array A as a pointer to the adjacent node. Thus, no more than one byte is needed to store the element of the matrix P. Q_{MLB} is a multi-level bucket data structure. According to the calculations in the paper [21], the structure of Q_{MLB} requires no more than $O(log(C))$ words of memory. Thus, the total amount of memory necessary for seam line optimization consists of memory sizes for the matrices $W(x, y)$, $D(x, y)$ and $P(x, y)$, the size of the structure Q_{MLB}, can be neglected since $log(C)$ is much smaller than the size of the graph in our case.

3.5 Parallel Implementation

We divide the problem into a number of blocks, while refining the seam line on detailed level, this allow us to process unlimited amount of data and perform parallel optimization in each individual blocks of information. OpenMP [22] library is used for parallel data processing. Implementation details are presented in Algorithm 2. Coarse step is done at lines 2-7. After the end of the first step, we divide the received coarse seam line into blocks and perform the second and the third steps (line 9) of parallel refinement of seam line (lines 11-23). Due to the fact that the sizes of the image fragments corresponding to the blocks of the coarse seam line depend on the shape of the line segment, we use the dynamic partition of the problem into parallel blocks (line 11). Note, that parallel reading of image data from the disk leads to a decrease in performance, this operation is separated to the critical section (line 16).

```
   Data: I_1 - first image, I_2 - second image
   Result: S - optimal seam line vertites
 2 O_1, O_2 ← readImageOverview(I_1, I_2);
 3 V_b, V_e ← computeIntersectionPoints(I_1, I_2);
 4 W_coarse ← computeEnergy(O_1, O_2);
 5 S_coarse ← shortestPath(W_coarse, V_b, V_e);
 7 B ← splitPathToBlocks(S_coarse);
 9 for step = 2; step <= 3; ++step do
11 |  #pragma omp parallel for
12 |  for i = 0; i < B.size(); ++i do
13 |  |  box ← computeBoundingBox(B_i);
14 |  |  V_b, V_e ← computeEndPoints(B_i);
16 |  |  begin #pragma omp critical
17 |  |  |  F_1, F_2 ← readFineImage(I_1, I_2, box);
18 |  |  end
19 |  |  W_fine ← computeEnergy(F_1, F_2);
20 |  |  S_fine,i ← shortestPath(W_fine, V_b, V_e);
21 |  end
23 |  S ← constructPathFromSegments(S_fine);
24 |  if step == 3 then
25 |  |  break;
26 |  end
27 |  B ← splitPathToBlocks(S);
28 end
29 return S;
```

Algorithm 2: Simplified pseudo code of parallel hierarchical seam line optimization

4 Results

Several datasets were used to test hierarchical seam line optimization algorithm:

- Colored (RGB) 8-bit per channel middle resolution images acquired by Landsat-8 satellite system 16000 × 16000 pixels each (Figure 3);
- Colored (RGB) 8-bit per channel high resolution images acquired by aerial camera 4368×2912 pixels each with roads digitized (Figure 5);
- Panchromatic 16-bit high resolution images acquired by Resurs-P satellite system 36000 × 105000 pixels size.

The first and second datasets were used to study the visual quality of the algorithm, and the latter was used to estimate the performance.

Algorithm was tested on a workstation equipped with a quad-core Intel Core i7 processor and 8 GB RAM.

Figure 4 shows the output mosaic obtained using the hierarchical seam line optimization algorithm, it can be seen that the seam line is practically indistinguishable both on a coarse scale and on a fine scale. The result of the algorithm on high resolution images using a vector layer of roads as forbidden zones is shown in Figure 6. It can be seen that the seam line crosses the roads as perpendicular as possible. But

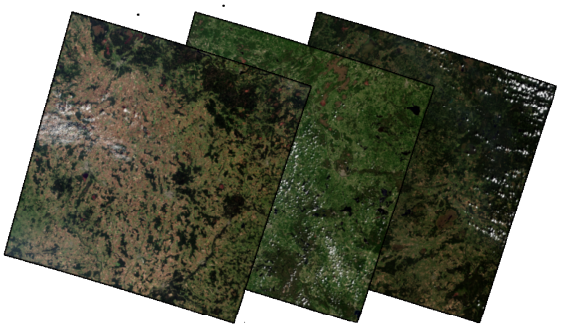


Figure 3: Middle resolution Landsat-8 images dataset

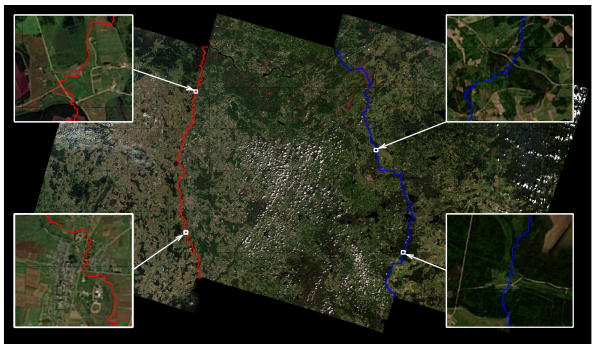
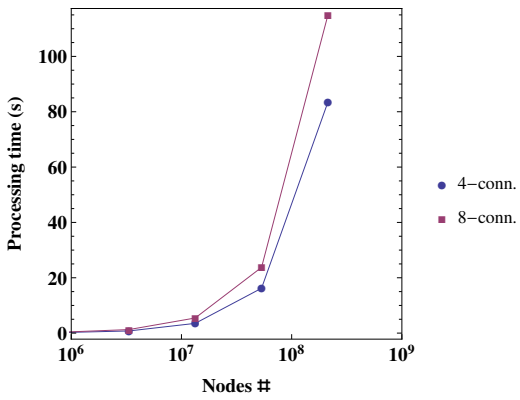


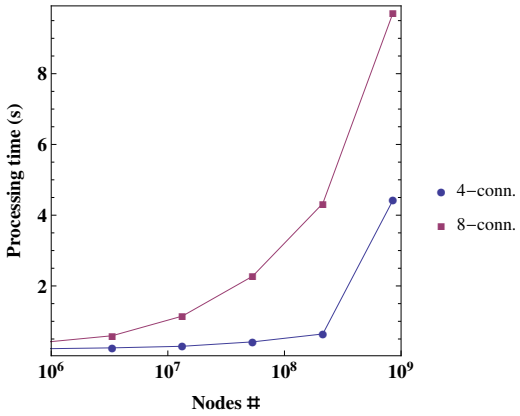
Figure 4: Output Landsat-8 mosaic (29140×16764 pixels size) overview with optimal seam line highlighted

Figure 5: High resolution aerial image dataset with roads digitized

Figure 6: Output mosaic of aerial images with a seam line that bypasses the roads in the optimal way



a)



b)

Figure 7: The dependence of the time of optimal seam line constructing on the number of vertices in the graph, representing the energy function, (a) one processor with loading entire energy function matrix into memory (RAM), (b) the proposed algorithm

additional restrictions on the seam line in the form of forbidden zones lead to the fact that near the zone, the visual similarity of images on the different sides of the line is lost.

The dependence of the time for optimal seam line constructing on the number of vertices in the graph is shown in the Figure 7.

The construction of the seam line for maximum size graph, with one-processor version with full data loading into memory was not performed due to high consumption of memory. As studies have shown, when constructing an eight-connected graph, the visual quality of the seam line is somewhat higher, because it includes diagonally oriented edges, but the processing time increases due to the increase in the number of edges in the graph and the non-integer weight of the diagonal edges. Also on the Figure 7b it is seen that for the size of graph 10^9 nodes the processing time sharply increases, which is due to the predominance of the time of data loading from the hard drive over the optimization time.

5 Conclusion

In this paper the approach to construct an optimal seam line for the Earth remote sensing image mosaic, based on the graph shortest path finding is presented. The construction of an energy function matrix taking into account both informativeness and similarity of input images is discussed. Graph data structure design from an energy matrix is presented, which makes it possible to reduce the amount of RAM. The hierarchical parallel implementation of the optimization algorithm is presented with regard to the proposed graph data structure. The amount of RAM required for the optimization process is estimated.

The proposed hierarchical image processing approach not only allows us to reduce a seam line construction time, but also to make seam line unnoticeable both at the detailed and at the overview scale of the output mosaic. Using proposed approach, the time for optimal seam line creation was reduced by more than 10 times, which make it possible to produce seamless image mosaic with minimal CPU time usage.

References

[1] Federal Service of Geodesy and Cartography of Russia., *Instruction on photogrammetric work in the creation of digital topographic Maps and plans*, Moscow, 2002.

[2] Skvortsov V.A., *The Delaunay Triangulation and Its Applications.* Tomsk. Univ. Press, 2002. 128 p. http://www.ict.edu.ru/lib/index.php?id_res=4503 (accessed:10.10.2017)

[3] Chon J., Kim H. *Seam-line determination for image mosaicking: A technique minimizing the maximum local mismatch and the global cost. ISPRS Journal of Photogrammetry and Remote Sensing, No. 65 (1)*, 2010. pp. 8692. https://doi.org/10.1016/j.isprsjprs.2009.09.001 (accessed:10.10.2017)

[4] Chon J., Wang J., Slankard T., Ristevski J., *High-Quality Seamless Panoramic Images. Special Applications of Photogrammetry, ed. Dr. Daniel Carneiro Da Silva*, 2012, InTech http://cdn.intechopen.com/pdfs/36193.pdf (accessed:10.10.2017)

[5] Zhao Y., Han T., Feng S., Miao C., *Remote Sensing Image Mosaic by Incorporating Segmentation and the Shortest Path. In: Bian F., Xie Y., Cui X., Zeng Y. (eds) Geo-Informatics in Resource Management and Sustainable Ecosystem. Communications in Computer and Information Science, vol 398. Springer, Berlin, Heidelberg, 2013* https://doi.org/10.1007/978-3-642-45025-9_67 (accessed:10.10.2017)

[6] Kerschner M., *Twin snakes for determining seam lines in orthoimagemosaicking. // IAPRS, Vol. XXXIII, Amsterdam,. 2000. pp. 1-8.* http://geo.tuwien.ac.at/fileadmin/editors/IPFpublications/mk_isprs2000/paper725.pdf (accessed:10.10.2017)

[7] Adrov V.N., Drakin M.A., Sechin A.Y., *High performance photogrammetric processing on computer clusters. // International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XXXIX-B4, XXII ISPRS Congress. Melbourne, Australia. 25 August 01 September 2012. pp. 109-112.* http://dx.doi.org/10.5194/isprsarchives-XXXIX-B4-109-2012 (accessed:10.10.2017)

[8] Bielski C., Grazzini J., Soille P., *Automated Morphological Image Composition for Mosaicing Large Image Data Sets. // Spatial Data Infrastructures Unit Institute for Environment and Sustainability DG Joint Research Centre, European Commission I-21020 Ispra, (Va), Italy. pp. 1-4.* https://doi.org/10.1109/IGARSS.2007.4423743 (accessed:10.10.2017)

[9] Wan Y., Wang D., Xiao J., Lai X., Xu J., *Automatic determination of seamlines for*

aerial image mosaicking based on vector roads alone. // ISPRS Journal of Photogrammetry and Remote Sensing, No. 76, 2013. pp. 1-10. https://doi.org/10.1016/j.isprsjprs.2012.11.002 (accessed:10.10.2017)

[10] Chen Q., Sun M., Hu X., Zhang Z. *Automatic Seamline Network Generation for Urban Orthophoto Mosaicking with the Use of a Digital Surface Model. // Remote Sens. 2014, 6. pp. 12334-12359.* http://www.mdpi.com/2072-4292/6/12/12334 (accessed:10.10.2017)

[11] Kass M., Witkin A., Terzopoulos D., *Snakes: active contour models. // International Journal of Computer Vision, 1(4), 1988, 321331* http://web.cs.ucla.edu/~dt/papers/ijcv88/ijcv88.pdf (accessed:10.10.2017)

[12] Kwatra V., Schodl A., Essa I., Turk G., Bobick A., *Graphcut Textures: Image and Video Synthesis Using Graph Cuts. // ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2003, Volume 22 Issue 3, July 2003, pp. 277-286* http://gamma.cs.unc.edu/kwatra/publications/gc-final-lowres.pdf (accessed:10.10.2017)

[13] Li Li, Yao Jian, Lu Xiaohu, Shan Jie., *Optimal Seamline Detection for Multiple Image Mosaicking via Graph Cuts. // ISPRS Journal of Photogrammetry & Remote Sensing. February 2015. pp. 1-32.* http://cvrs.whu.edu.cn/publications/2015/GraphCutsSeamlineDetection-ISPRS2015.pdf (accessed:10.10.2017)

[14] Boykov, Y., Veksler, O., Zabih, R., *Fast approximate energy minimization via graph cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (11), pp 12221239. 2001.* http://www.cs.cornell.edu/rdz/Papers/BVZ-pami01-final.pdf (accessed:10.10.2017)

[15] Boykov Y., Kolmogorov V. An Experimental Comparison of Min-cut/Max-flow Algorithms for Energy Minimization in Vision. In: Figueiredo M., Zerubia J., Jain A.K. (eds) Energy Minimization Methods in Computer Vision and Pattern Recognition. EMMCVPR 2001. Lecture Notes in Computer Science, vol 2134. Springer, Berlin, Heidelberg https://doi.org/10.1007/3-540-44745-8_24 (accessed:10.10.2017)

[16] Y. Boykov and A. Delong, *A Scalable Graph-Cut Algorithm for N-D Grids. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Anchorage, June 2008* https://doi.org/10.1109/CVPR.2008.4587464 (accessed:10.10.2017)

[17] F. R. Schmidt, E. Toppe, D. Cremers, *Efficient Planar Graph Cuts with Applications in Computer Vision. IEEE CVPR, Miami, Florida, June 2009* https://doi.org/10.1109/CVPR.2009.5206863 (accessed:10.10.2017)

[18] Li Li, Jian Yao, Xiaohu Lu, and Jing Ren. *Superpixel-Based Optimal Seamline Detection via Graph Cuts for Panoramic Images. submitted to IEEE International Conference on Image Processing (ICIP), 2015* http://cvrs.whu.edu.cn/projects/SSD/papers/SuperpixelStitching-ICIP2015.pdf (accessed:10.10.2017)

[19] Moravec H., Towards automatic visual obstacle avoidance // Proceedings of the 5-th International Joint Conference of Artificial Intelligence Cambridge. 1977.

[20] Dijkstra E.W., A note on two problems in connexion with graphs // Numerische Mathematik, Vol. 1, 1959. P. 269271.

[21] Goldberg A.V. *A Simple Shortest Path Algorithm with Linear Average Time. // Proceedings of the 9th European Symposium on Algorithms (ESA '01), Springer Lecture Notes in Computer Science LNCS 2161. 2001. pp. 230-241.* https://doi.org/10.1007/3-540-44676-1_19 (accessed:10.10.2017)

[22] OpenMP [2017]. URL: http://openmp.org/ (accessed: 01.02.2017).