Hindawi Publishing Corporation
Journal of Sensors
Volume 2016, Article ID 5258473, 8 pages
http://dx.doi.org/10.1155/2016/5258473

Research Article

A Novel Seam Finding Method Using Downscaling and Cost for Image Stitching

Jinwook Jeong and Kyungkoo Jun

Department of Embedded Systems Engineering, University of Incheon, Incheon 22012, Republic of Korea

Correspondence should be addressed to Kyungkoo Jun; kjun@incheon.ac.kr

Received 23 January 2016; Accepted 18 April 2016

Academic Editor: Yurong Qian

Seaming finding is an important step for creating panorama images because it smoothes away differences observed at boundaries between stitched images. We propose an improved seam finding method in which we define a cost function to measure the discrepancies that boundary pixels cause. We are also able to improve computing complexity by avoiding finding a seam over the whole area of overlapped region. Instead, we use a downscaled version of overlapped area to approximate a seam and then interpolate the seam to the original region. From experiments to generate panorama images, we compare our method with three other existing seam finding algorithms and observe that our method is able to produce better quality panorama image than the existing methods, while the processing time is comparable to those of the others.

1. Introduction

Panorama images contribute to providing realistic experience by having a wider field of view and have various application areas such as CCTV and action cams [1–3]. In general, cameras equipped with wide angle lenses are used to capture panorama images, or postprocessing software creates panorama by stitching multiple images taken by the same or different cameras.

Postprocessing software for panorama generation consists of two steps. The first step is called registration in which feature points are detected and matched between neighbor images to compute parameters such as focal distance and homography. These parameters are then used to warp images to create a large panorama. The second step is blending which balances exposure differences between images and removes seams and ghost effects from moving objects, aiming to remove stitching trace as much as possible [4–7].

The steps involved in panorama creation depend on various parameter settings, making it harder to create complete and perfect panorama images. Diverse research efforts have been made to improve the quality of resulting panorama. Various blending methods have been proposed [8–11]. A technique based on camera motion estimation was proposed to improve the performance of the registration [8]. Seam finding algorithm was proposed by using image patches obtained by graph-cut algorithm [9]. The performance of seam finding was improved by image segmentation based on watershed algorithm [10]. Dynamic programming technique was able to improve both speed and precision even though it worked on pixel level [11].

This paper proposes a seam finding method for the blending step, which is the second step of the panorama creation. We aim to improve performance while preserving the precision of seam location. For performance, we scale down images by resizing them to half their size, resulting in less computation time. The seam line that we find in the small sized images is just the approximation of the perfect seam line. Based on the approximation, since the region where the seam line exists is known, we can limit the area where seam finding should be performed rather than applying to whole area. It results in improved performance, while maintaining precision.

This paper is organized as follows. Section 2 describes the proposed seam finding method in detail and Section 3 evaluates the performance of the method and compares it with those of existing methods. And Section 4 concludes the paper.

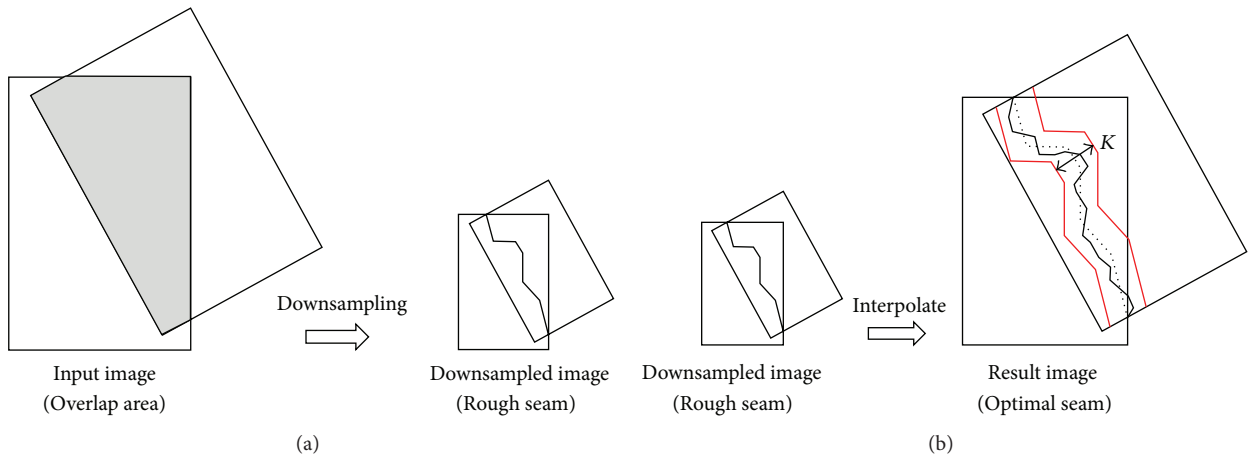


Figure 1: Seam finding steps using enhanced dynamic programming: (a) step 1 and (b) step 2.

2. Downscaling and Cost Function Based Seam Finding Method

We propose using downscaled images for estimating seam because it is able to reduce time. But it is challenging not to sacrifice precision of resulting seams. The proposed scheme consists of two sequential steps. The first step is to estimate an approximated seam from an overlapped region between two images as shown in Figure 1(a). Note that, before staring seam finding by dynamic programming, we reduce the size of the overlapped area by r times. Thus it has the effect to reduce the computation load and time required for seam finding. However, the resulting seam is just an approximation close to the real seam. In the second step shown in Figure 1(b), the reduced region is restored back to the original size. During this phase, the seam is also extended, filling up missing parts by interpolation. With the interpolated seam, we set a new area which is within K pixels away from the seam. Then, we are sure that a real seam is contained in the area. Finally, we perform the dynamic programming again over the area to find the real seam:

$$e_i = \sqrt{(G_i)^2 + (I_i)^2},$$

$$G_i = \frac{\left\|\nabla P_1^i + \nabla P_2^i\right\|}{\max\left(\left\|\nabla P_1^i + \nabla P_2^i\right\|\right)}, \tag{1}$$

$$I_i = \frac{\left\|P_1^i - P_2^i\right\|}{\max\left(\left\|P_1^i - P_2^i\right\|\right)},$$

$$E_{(i,j)} = e_{(i,j)} + \min\left(E_{(i-1,j-1)}, E_{(i,j-1)}, E_{(i+1,j-1)}, E_{(i-1,j)}, E_{(i+1,j)}\right). \tag{2}$$

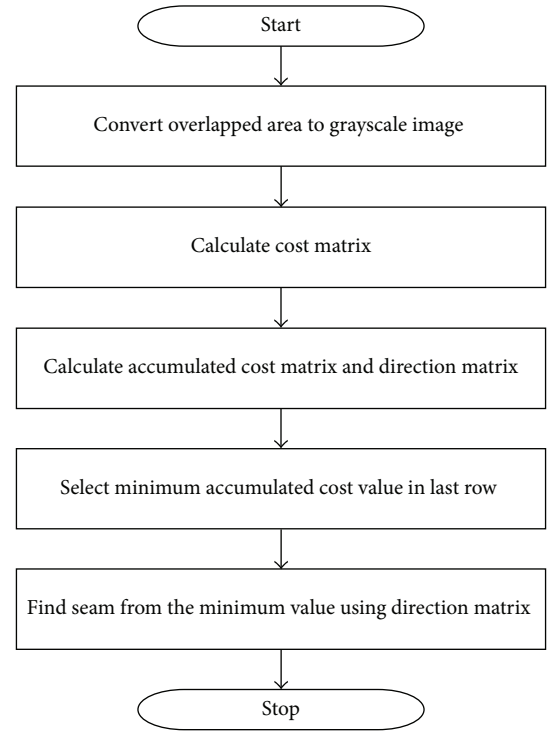


Figure 2: Flow chart of the seam finding.

Figure 2 is the flow chart of the seam finding. We find a seam between two images, Img_1 and Img_2. At first, an overlapped region between Img_1 and Img_2 is determined. Then, the subimage of Img_1 is located which belongs to the overlapped region and it is converted to gray scale. We call it Gray_1. In the same way, we locate Gray_2, the subimage from Img_2. It is obvious that there exists a corresponding pixel within Gray_2 for every pixel from Gray_1. We call a pixel i of Gray_1 and its corresponding Gray_2 pixel as P_1^i and P_2^i,

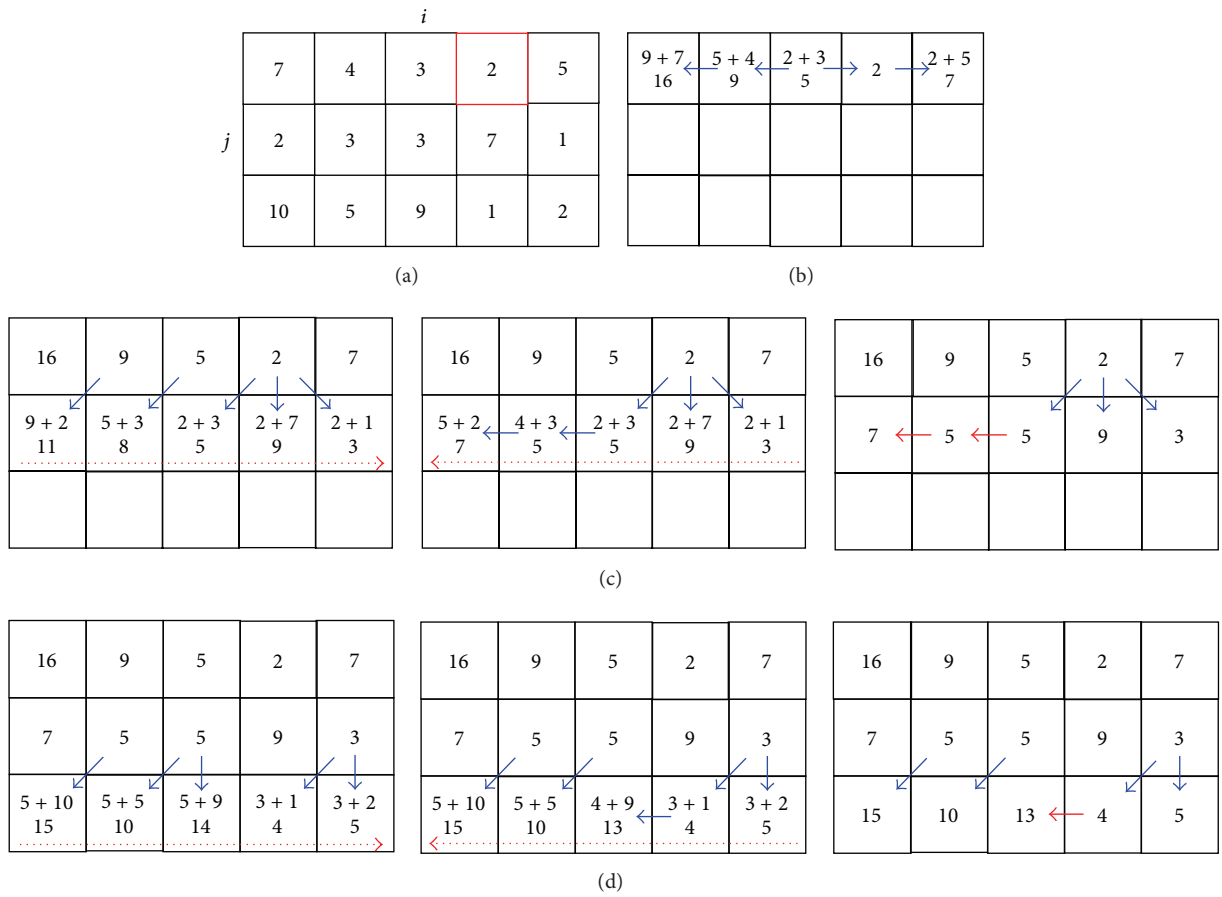


FIGURE 3: Process of accumulated cost matrix calculation (a) M_{cost}, (b) first row calculation of $M_{\text{a\_cost}}$, (c) second row calculation of $M_{\text{a\_cost}}$, and (d) third row calculation of $M_{\text{a\_cost}}$.

respectively. And we call P_i the pixel of the overlapped region which has corresponding pixels P_1^i and P_2^i.

Then, for every pixel P_i, we compute its cost e_i. The cost, also called energy, is a real value assigned for each pixel and represents how important the pixel is. Often, gradient or intensity of a pixel is used as the cost. In this paper, we compute e_i as (1), where ∇P_1^i and ∇P_2^i are the gradients of P_1^i and P_2^i.

Then, we can obtain a cost matrix M_{cost} for the overlapped region. Each element of M_{cost} corresponds to P_i and has e_i as its value. From M_{cost}, two other matrices are computed. One is an accumulated cost matrix $M_{\text{a\_cost}}$ and the other is a direction matrix M_{dir}. By using information from both matrices, we are able to determine where a seam can be drawn without noticeable differences between Img_1 and Img_2. $M_{\text{a\_cost}}$ is used to find candidate pixels on which a seam is drawn, while M_{dir} provides pixel-by-pixel directional information of the seam.

Figure 3 shows how $M_{\text{a\_cost}}$ matrices are made from M_{cost}. Each element of $M_{\text{a\_cost}}$, $E(i, j)$, is called an accumulated cost of a pixel at ith column and jth row. It is calculated as (2), where $e(i, j)$ is the cost of the pixel at (i, j) from M_{cost}. $E(i, j)$ of a pixel is computed by adding its cost to the minimum accumulated cost of its neighboring pixels. Note that, among eight neighboring pixels, the three at the lower row are ignored from consideration. Also, the top row pixels do not consider the upper row pixels because they do not exist.

For ease of understanding, we describe step by step how each $E(i, j)$ is computed. At start, among the top row pixels, we find the pixel with the minimum $e(i, j)$ and set $E(i, j) = e(i, j)$ which is the fourth pixel with the value 2 in Figure 3(a). Then, accumulated costs of neighbor pixels $E(i - 1, j)$ and $E(i + 1, j)$ are computed according to (2). This procedure is repeated in a flooding way until all the pixels at the top row are processed, resulting in Figure 3(b). We then move on to the next row. Note that when calculating $E(m, n)$, still $E(m+1, n)$ is not determined yet. Thus we need to iterate twice; however the direction of calculation changes: from left to right first and then from right to left as shown in Figure 3(c): the leftmost figure of Figure 3(c) shows how the calculation proceeds from left to right, the middle one shows that the calculation progresses from right to left, and the leftmost one is the result of the calculation. Figure 3(d) shows how the third row is processed. The whole process continues until the last row is finished.

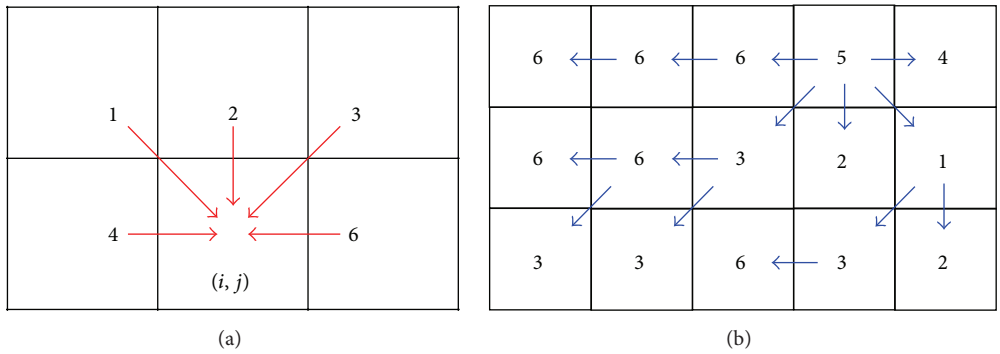


Figure 4: Direction description of M_{dir}: (a) direction of M_{dir} and (b) example of M_{dir}.

With $M_{\text{a\_cost}}$, we can determine for $E(i, j)$ which neighboring pixel is selected to contribute to the computation. Figure 4(a) shows five candidate pixels each given numbers 1 to 6, except 5 which is reserved for the first pick pixel at the top row. M_{dir} stores for each pixel which neighboring pixel is selected for its $E(i, j)$. For example, Figure 4(b) shows an example M_{dir} corresponding to Figure 3(d). Note that the fourth pixel at the top row has the value 5, because no neighboring pixel is considered. The arrows are drawn for ease of understanding.

Once completing $M_{\text{a\_cost}}$ and M_{dir}, a seam can be determined. It starts from the bottom row and moves up in an antiarrow direction, drawing a seam over visited pixels. To describe the procedure in detail, among the pixels on the last row of $M_{\text{a\_cost}}$, we select the one with the minimum $E(i, j)$. Then, we follow the arrow in a reverse way to move to one of the neighboring pixels; for this we use the direction information from M_{dir}. Finally, if we connect all the pixels that were visited before reaching the top row, those pixels consist of a seam. Since the seam consists of only the pixels with least distinction, the resulting boundary provides smooth transition between two images.

There is an unusual case in which more than one pixel has the minimum cost at the top row in M_{cost}. In this case, we cannot avoid increasing computation load, because, for each pixel, we compute $M_{\text{a\_cost}}$ and M_{dir} independently and separately. Thus, we produce multiple sets of resulting matrices. However, we choose only the set of which $M_{\text{a\_cost}}$ contains the least value $E(i, j)$ at its bottom row, ignoring the rest. Then, the seam determination is processed in the same way as explained.

3. Performance Evaluation

We evaluate the performance of the proposed scheme through experiments that create panorama images by stitching multiple overlapping images. In the experiments, we observed and compared the resulting stitching quality between the proposed method and existing methods. A representative seam finding algorithm [7] based on dynamic programming and a scheme depending on graph cut [8] and a Voronoi diagram method were chosen for comparison. We refer to the former as the seam finding and the latter as the graph cut.

Table 1: Experiment PC specification.

Processor	Intel® Core™ i7-4790 CPU @ 3.60 Ghz
Memory	16 GM
OS	Ubuntu 14.04 LTS 64 bits (VMware)

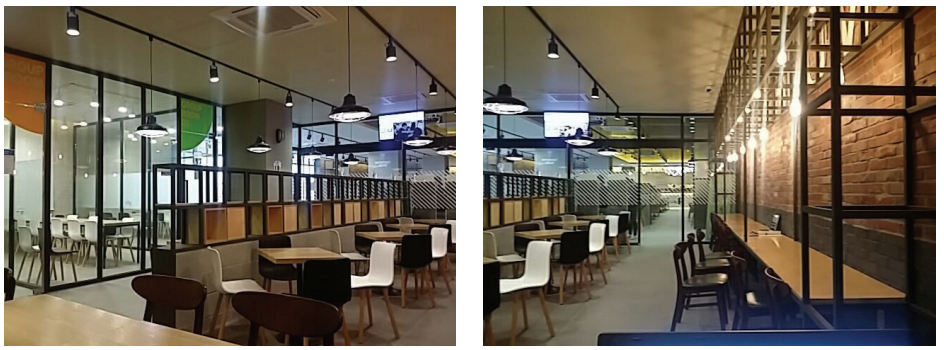
We used four different sets of test images. Each set consists of two overlapping images each corresponding to left and right. Each image has the resolution of 640 ∗ 480. We used the computer with the specification of Table 1 for the experiments and implemented our algorithm by using an open source library, OpenCV 2.4.9 [12], with the parameters r and K set as 3 and 40, respectively.

Figure 5 shows the four sets of images which are used for the experiments. Each set consists of two images: left and right. The two images have common overlapping area by which they are stitched together. We select the test images such that overlapping areas are different features.

Figure 6 shows the stitching results of test image (a). Figures 6(a), 6(b), 6(c), and 6(d) are the results of the dynamic programming, the graph cut, the Voronoi diagram, and the proposed method, respectively. We can observe that, in all the methods except our method, the rail installed on the ceiling is not continuous, and a similar discrepancy is also found at the chair leg on the floor. However, the result of the proposed scheme is free from those flaws.

Figure 7 shows the stitching results of test image (b). Figures 7(a), 7(b), 7(c), and 7(d) are the results of the dynamic programming, the graph cut, the Voronoi diagram, and the proposed method, respectively. We can observe that, in all the methods except our method, the rectangular pillar in the middle has a very large mismatch. However, the result of the proposed scheme is free from those flaws.

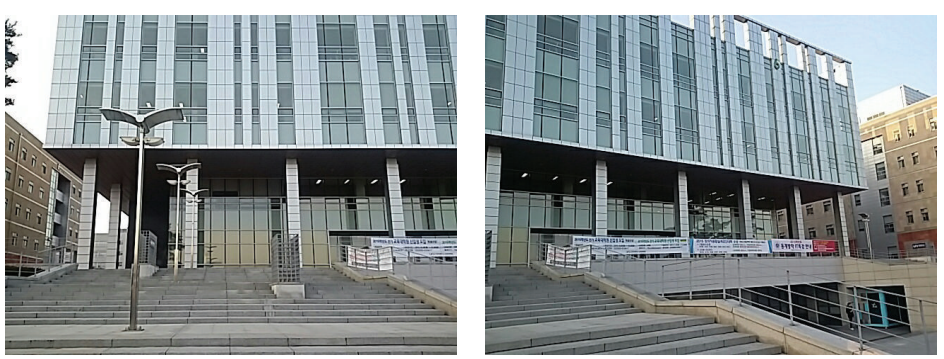
Figure 8 shows the stitching results of test image (c). Figures 8(a), 8(b), 8(c), and 8(d) are the results of the dynamic programming, the graph cut, the Voronoi diagram, and the proposed method, respectively. We can observe that, in all



(a)

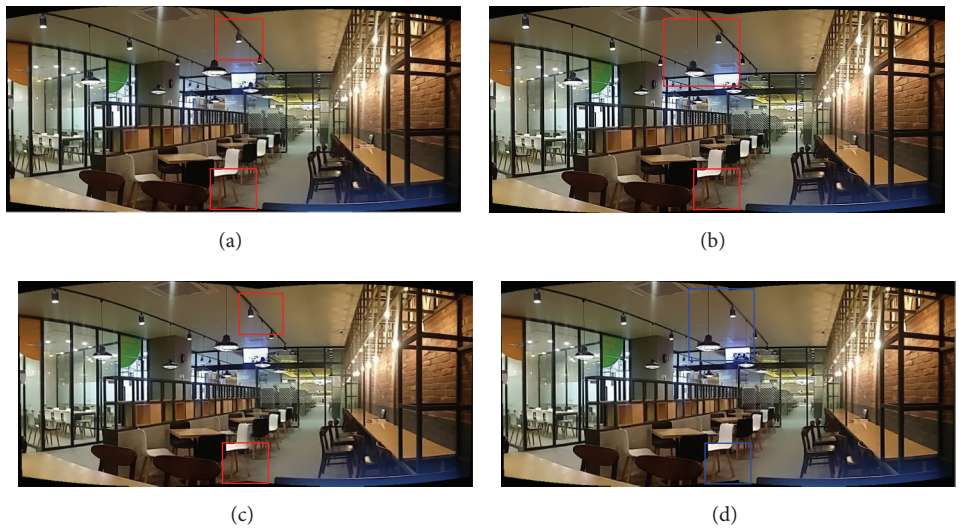
(b)

(c)



(d)

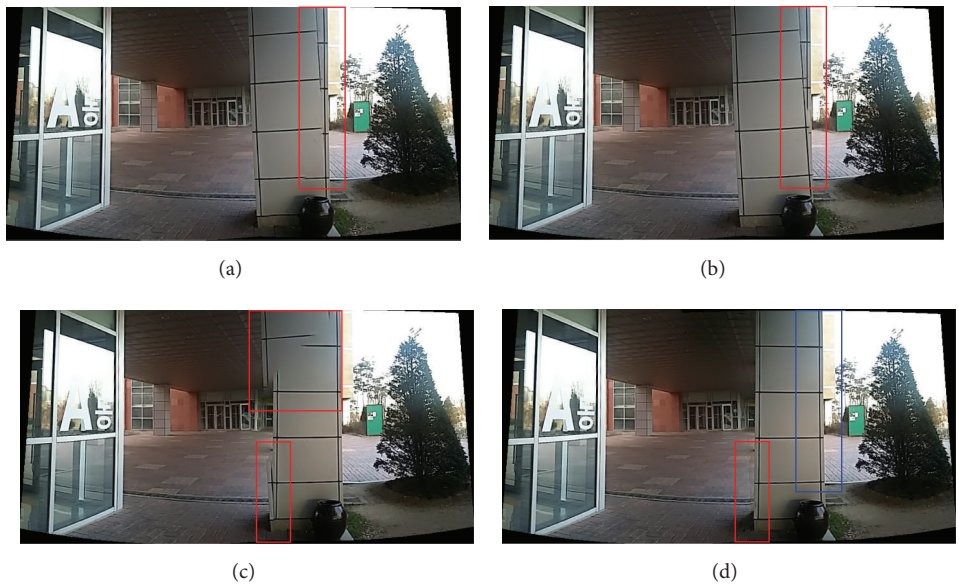
Figure 5: Four sets of test images. Each set consists of left and right images stitched to form a panorama.



(a) (b)

(c) (d)

Figure 6: Result comparison for the test image (a) by the seam finding methods: (a) dynamic programming, (b) graph cut, (c) Voronoi diagram, and (d) proposed method.



(a) (b)

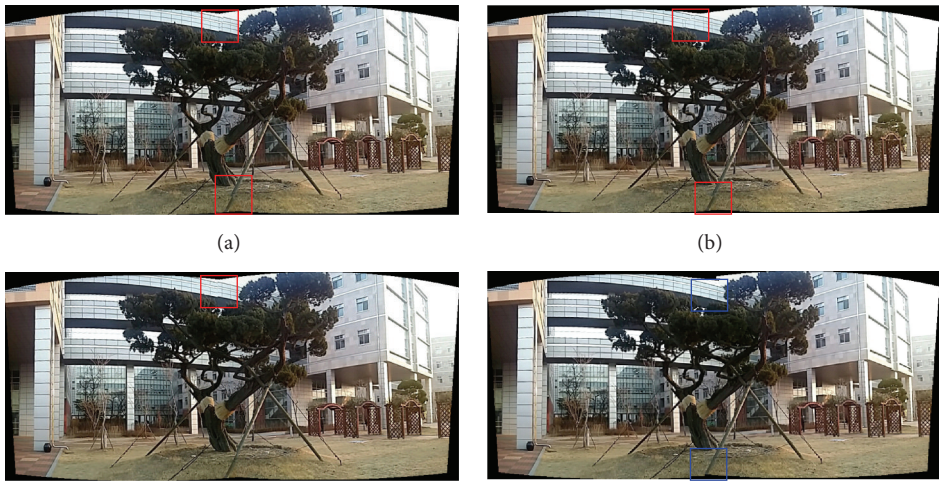
(c) (d)

Figure 7: Result comparison the test image (b) by the seam finding methods: (a) dynamic programming, (b) graph cut, (c) Voronoi diagram, and (d) proposed method.

the methods except our method, each image has a similar discrepancy at the top middle part and also at the lower middle part. However, the result of the proposed scheme is free from those flaws.

Figure 9 shows the stitching results of test image (d). Figures 9(a), 9(b), 9(c), and 9(d) are the results of the dynamic programming, the graph cut, the Voronoi diagram, and the proposed method, respectively. We can observe that all the methods show similar results without noticeable flaws or mismatches.

Figure 10 shows the processing times of the methods. Given a set of left and right video streams as input to each method, we measured the times consumed for stitching each frame. It is observed that the elapsed times vary as frames proceed. It is because the contents of stitched images are different according to frames, resulting in different processing times. The Voronoi diagram was the fastest with average of 2.6 msec, while the graph cut was the slowest with average of 52.5 msec. The proposed method scored 36.1 msec which was better than the graph cut but was slower than the others.



(a) (b)

(c) (d)

Figure 8: Result comparison for the test image (c) by the seam finding methods: (a) dynamic programming, (b) graph cut, (c) Voronoi diagram, and (d) proposed method.

(a) (b)

(c) (d)

Figure 9: Result comparison for the test image (d) by the seam finding methods: (a) dynamic programming, (b) graph cut, (c) Voronoi diagram, and (d) proposed method.

However, considering that the resulting panorama quality of the proposed method is superior to the others, the increased time was spent for Gaussian filtering, entropy calculation to improve the results.

4. Conclusions

We proposed an improved seam finding method which is important to smooth out the boundary of stitched images in panorama creation. For this, we defined a cost function to determine how each pixel affects stitching results. We were also able to reduce time consumption; instead of finding a seam over the whole area of overlapped region, we worked on a downscaled version of the overlapped image and interpolated to the original size. We carried out panorama creation experiments in which our method as well as existing seam finding algorithms was used. Resulting panorama image quality was superior when our method was used, while the processing time of our method was comparable to those of others.

Before concluding the paper, it should be noted that the interpolation that was used to improve the performance of the proposed method can induce improper seam finding results, unless it is carefully tuned. Currently, we depended on algebraic interpolation. But the effects of such scheme will be evaluated and discussed in future research.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

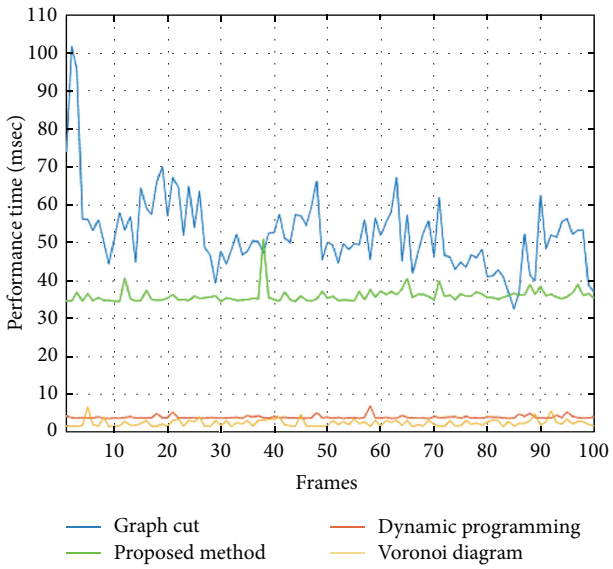


Figure 10: Performance comparison: elapsed times of the seam finding methods over video.

Acknowledgments

This work was supported by the Incheon National University Research Grant in 2015.

References

[1] Panono, https://www.panono.com/.

[2] Rotopan. http://rotopan.com/.

[3] Pelco, https://www.pelco.com/.

[4] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "SURF: speeded-up robust features," in *Proceedings of the European Conference on Computer Vision*, pp. 346–359, 2006.

[6] S. Avidan and A. Shamir, "Seam carving for content-aware image resizing," *ACM Transactions on Graphics* , vol. 26, no. 3, pp. 1–9, 2007.

[7] Y. Y. Boykov and M. P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images," in *Proceedings of International Conference on Computer Vision*, vol. 1, pp. 105–112, IEEE, Vancouver, Canada, 2001.

[8] B. Kim, S. Lee, and N. Cho, "Real-time panorama image synthesis by fast camera pose estimation," in *Proceedings of the Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC '12)*, pp. 3–6, Hollywood, Calif, USA, December 2012.

[9] V. Kwatra, A. Schodl, and I. Essa, "Graphcut textures: image and video synthesis using graph cut," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 597–607, 2003.

[10] N. Gracias, M. Mahoor, and S. Negahdaripour, "Fast image blending using watersheds and graph cuts," *Image and Vision Computing*, vol. 27, no. 5, pp. 597–607, 2009.

[11] H. Gu, Y. Yu, and W. Sun, "A new optimal seam selection method for airborne image stitching," in *Proceedings of the International Workshop on Imaging Systems and Techniques*, pp. 159–163, Shenzhen, China, May 2009.

[12] OpenCV. http://opencv.org.