Fast Panorama Stitching for High-Quality Panoramic Images on Mobile Phones

Yingen Xiong and Kari Pulli, *Member*, IEEE

Abstract — *This paper addresses the problem of creating high-resolution and high-quality panoramic images from long image sequences with very different colors and luminance in source images. A fast stitching approach is proposed for combining a set of source images into a panoramic image using little memory, and implemented on mobile phones. In this approach, color correction reduces color differences of source images and balances colors and luminance in the whole image sequence, dynamic programming finds optimal seams in overlapping areas between adjacent images and merges them together, and image blending further smoothens color transitions and hides visible seams and stitching artifacts. A sequential panorama stitching procedure constructs panoramic images. The advantages include fast processing speed using dynamic programming for optimal seam finding, reducing memory needs by using the sequential panorama stitching, and improved quality of image labeling and blending due to the use of color correction. The approach has been tested with different image sequences and it works well on both indoor and outdoor scenes*[1].

Index Terms — Mobile panorama, image stitching, fast labeling, image blending.

I. INTRODUCTION

A panoramic image has a wide field of view, much wider than is available on normal cameras such as those in mobile phones. By stitching together a sequence of overlapping normal images, we can create a panoramic image. Image stitching is a very important step in creating panoramas. A simple pasting of overlapping images into the final panorama produces visible seams due to changes of scene illumination and camera responses, or spatial alignment errors.

The task of image stitching is to find optimal seams in overlapping areas of source images, merge them along the seams, and minimize merging artifacts. In this paper, we are creating high-resolution and high-quality panoramic images on mobile phones, so that a user can capture an image sequence of a wide range of scenes with a camera phone and see a panoramic image created immediately on the phone.

A. Background

Mobile phones are not only efficient communication tools, but also capable computational devices equipped with high-resolution digital cameras, high-quality color displays, and GPU hardware. Applications such as mobile augmented reality, mobile local search, and mobile image matching and recognition used to only work on desktop computers, but can now run on mobile phones. Here we are building panorama applications on these devices.

A panorama construction process requires a lot of computation and memory. Mobile phones only have limited resources. It is necessary to develop efficient stitching methods to fit mobile applications.

B. Related Work

There are two main categories of current image stitching approaches: transition smoothing and optimal seam finding.

Transition smoothing approaches reduce color differences between source images to make seams invisible and remove stitching artifacts. Alpha blending [1] is a widely used simple and fast transition smoothing approach, but it cannot avoid ghosting problems caused by object motion and small spatial alignment errors. Recently, gradient domain image blending approaches [5]-[8] have been applied to image stitching. These algorithms can reduce color differences and smooth color transitions using gradient domain operations, producing high-quality composite images.

Optimal seam finding approaches [4], [9]-[12] search for seams in overlapping areas along paths where differences between source images are minimal. The seams can be used to label each output image pixel with the input image that should contribute to it label which input image contributes to each output pixel.

The combination of optimal seam finding and transition smoothing for image stitching has also been used in panorama applications [4], [13], and [15]. Source images are combined by compositing along optimal seams. If the seams and stitching artifacts are visible, transition smoothing is applied to reduce color differences to hide the artifacts.

Current panorama stitching approaches running on camera phones can be found in [13], [2], and [3]. In [13], graph cut is used for finding optimal seams to merge the source images together and Poisson blending is used for smoothing color transitions. High-quality panoramic images can be obtained. However, computational and memory costs are high. In [2] and [3], source images are stitched together with a procedure including color correction, seam finding, and simple band-linear blending. The stitching process is simple. However, the quality of panoramic images is not high. There are several problems in this approach. Pixels are easy saturated in color correction. It does not work well for source images in very different colors and luminance. The simple band-linear blending is not sufficient when color correction can not

[1] Yingen Xiong and Kari Pulli are with Nokia Research Center, Palo Alto, CA 94304, USA (e-mail: yingen.xiong@nokia.com; kari.pulli@nokia.com).

Manuscript received April 7, 2010
Current version published 06 29 2010;
Electronic version published 07 06 2010.

remove color differences efficiently, which results in low-quality panoramic images. Like other linear blending, moving objects on the overlapping areas will cause ghosting artifacts. All these problems are solved in our proposed approach.

We have created a fast image stitching approach that uses relatively little memory. It includes color correction, image labeling, and image blending operations. We perform color correction for all source images to reduce color differences and smoothen remaining color transitions between adjacent images. Since the RGB pixel values of input images are gamma-corrected and therefore non-linear, we calculate the color averages used to find color correction coefficients using linearized RGB values. A global adjustment process is applied to reduce magnitude of average color correction to lower the chance of saturating pixel values during color correction. In the image labeling operation, an error surface is constructed with squared differences between overlapping images. A low-cost path is found through the error surface by dynamic programming and used as an optimal seam to create labeling. The overlapping images are merged together along the optimal seam. Compared to the commonly used graph cut method, the labeling process is much faster and memory consumption is much lower. In order to further smoothen color transitions between adjacent source images, we perform image blending after the source images are merged using image labels. A simple linear blending is used when source images are similar in color and luminance. When the colors remain too different, Poisson blending hides visible seams. The use of color correction for the source images can improve qualities of image labeling and image blending. It can also speed up the blending process. A sequential panorama stitching procedure is created with the fast image stitching approach. In this way, we can produce high-resolution panoramic images from large source images with low computational and memory costs.

C. Contributions

We (i) propose a fast panorama stitching approach with color correction, fast labeling, and image blending for creating high-resolution and high-quality panoramic images on mobile phones; (ii) improve qualities of optimal seam finding and transition smoothing by combining color correction with image labeling and image blending; (iii) reduce computation of the Poisson blending process with pre-smoothing color differences of source images; (iv) create a sequential image stitching procedure for mobile applications to quickly construct high-resolution panoramas with long image sequences using little memory; (v) present various examples and compare performance with other approaches to demonstrate advantages of the proposed approach; (vi) implement it on mobile phones.

II. SUMMARY OF OUR APPROACH

Fig. 1 shows the workflow of the fast panorama stitching procedure. We start with setting the stitching order (S_0, S_1, …, S_n) of the source images by sorting their offsets with respect to the final panorama. We calculate color correction coefficients for each neighboring image pair in the linearized RGB color space for all source images, and then compute a global adjustment factor that reduces cumulative color correction and the risk of saturating colors. Next, we find an image with more realistic colors in the image sequence, and adjust the first image using a chain of relative color corrections, modified with the global correction factor, so that the colors of the best image remain after correction as they were. After allocating memory for the final panoramic image I_c and initializing it with the first image S_0, we start to stitch other source images to the panoramic image sequentially.

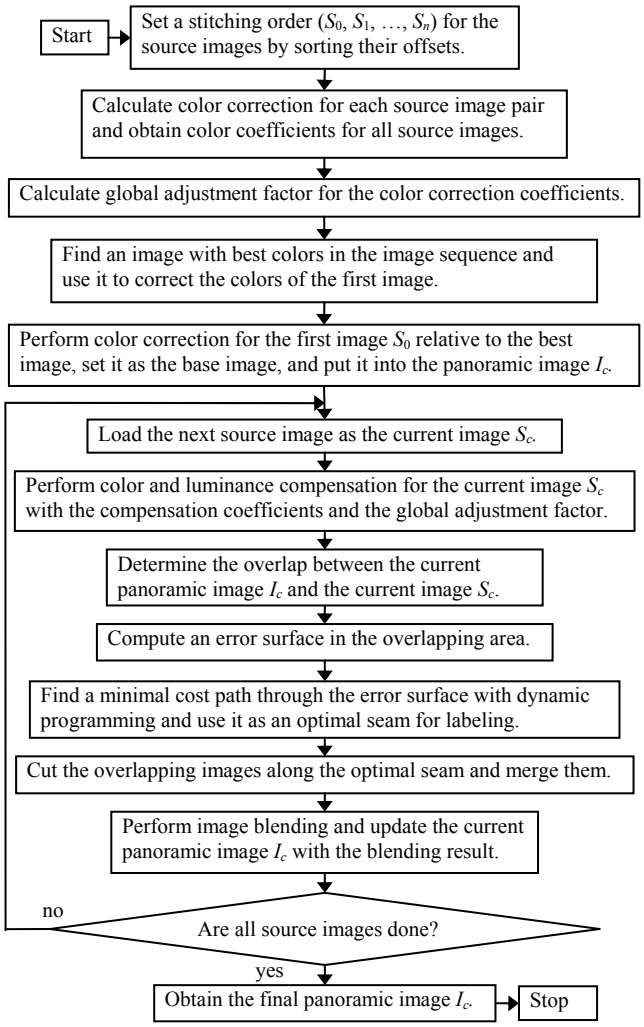


Fig. 1. Workflow of the fast panorama stitching approach.

We load the next source image as the current processing image S_c and perform color correction with the color correction coefficients and the global adjustment factor. In order to merge the current image with the current panorama, we extract the overlapping area between these two images and compute a squared difference between the overlapping images as an error surface. We find a minimal cost path through the error surface with dynamic programming. That path is used as an optimal seam to cut the overlapping images and merge them together. We perform image blending to further reduce

color differences and smooth color transitions between the current source image S_c and the panoramic image I_c. With the blending result, we can update the panoramic image I_c.

The process is repeated for all source images, until we obtain the final panoramic image. Unlike the global image stitching in [13], we do not need to keep all source images in memory due to the sequential stitching. The use of dynamic programming for optimal seam finding allows image labeling much faster than using graph cut. The combination of color correction and image blending allows us to construct high-quality panoramic images.

Although we describe the approach using the 1D stitching case, i.e. cameras move horizontally or vertically, it has been already extended to 1D stitching, i.e. cameras move in any arbitrary direction and source images can be stitched together in any arbitrary order.

(a) (b)

Fig. 2. Image stitching without and with color correction

III. COLOR AND LUMINANCE COMPENSATION

We capture the images using automated settings for focus, exposure, and white balance. As illumination changes across the scene, different images have different values for exposure and white balance, leading sometimes to large differences in colors in neighboring images. If no further color processing is done, visible artifacts may be created in panorama stitching.

Fig. 2 (a) shows an example, where the upper row shows three source images with different colors. The bottom row shows a stitching result. In this case, we can clearly see color differences and seams between the source images. It is necessary to perform color correction for the source images to reduce the differences and improve the stitching quality.

In order to better match the colors, we compute light averages in the overlap area using linearized RGB values instead of the default gamma-corrected RGB. In an image sequence S_0, S_1, … S_i, … S_n, suppose S_{i-1} and S_i are adjacent images, and S_{i-1}^o and S_i^o are where image overlap. We compute color correction coefficients for image S_i by linearizing the gamma-corrected RGB values as

$$\alpha_{c,i} = \frac{\sum_p (P_{c,i-1}(p))^\gamma}{\sum_p (P_{c,i}(p))^\gamma} \qquad c \in \{R,G,B\}\ (i = 1,2,3,\cdots,n), \tag{1}$$

where $P_{c,i-1}(p)$ is the color value of pixel p in image S_{i-1}^o; $P_{c,i}(p)$ is the color value of pixel p in image S_i^o; and γ is a gamma coefficient. Usually we set γ to 2.2. For the first image S_0, we set $\alpha_{c,0}$ to 1. To avoid saturating color values, we perform a global adjustment for color in the whole image sequence. We calculate a global adjustment factor g_c for each color channel c so that the overall adjustments $g_c\alpha_{c,i}$ approximate 1 by solving the least-squares equation

$$\min_{g_c} \sum_{i=0}^{n} (g_c \alpha_{c,i} - 1)^2 \qquad c \in \{R,G,B\}. \tag{2}$$

Equation (2) is a quadratic function in adjustment g_c which can be solved in closed form by setting the derivative to 0,

$$g_c = \frac{\sum_{i=0}^{n} \alpha_{c,i}}{\sum_{i=0}^{n} \alpha_{c,i}^2} \qquad c \in \{R,G,B\}\ (i = 0,1,\ldots,n). \tag{3}$$

With the correction coefficients $\alpha_{c,i}$ and the global adjustment factor g_c, we perform color correction for the whole image S_i,

$$P_{c,i}(p) \leftarrow (g_c \alpha_{c,i})^{1/\gamma} P_{c,i}(p), c \in \{R,G,B\}\ (i = 0,1,\ldots,n), \tag{4}$$

where $P_{c,i}(p)$ is the color value of pixel p in image S_i in color channel $c \in \{R, G, B\}$. Since the input and output values are gamma-corrected, we also gamma-correct adjustments $g_c\alpha_{c,i}$.

As described before, we use a best image found in the image sequence to correct colors for the first image. It is difficult to automatically determine the image with the best colors, since that is also partially an esthetic judgment call, and ideally the user should select the image whose colors she likes the best. As a heuristic, we select the image with most similar means in the R, G, and B channels, using the gray world assumption often used in white balancing.

Fig. 2 (b) shows the results of color correction and image stitching for the source images shown in Fig. 2 (a). From the results we can see that color correction reduces the color differences so that hardly any seam remains visible.

There are two main advantages in the way we do the color correction. Linearizing the light while calculating the correction factors matches the colors better than if the averages were calculated in gamma-corrected RGB, and the global adjustment for color correction coefficients attenuates the corrections, reducing accumulation of corrections that may lead to color saturation.

(a) (b)

Fig. 3. Ghosting artifacts caused by object motion and deghosting.

IV. OPTIMAL SEAM FINDING AND IMAGE LABELING

Object motion and spatial alignment errors may cause ghosting artifacts during image stitching. Fig. 3 (a) shows an example where a person was moving while the image sequence was captured. From the stitching result we can see the ghosting problem caused by the motion.

The objective of optimal seam finding is to find seams in the overlapping areas of source images, create labeling for all pixels in the composite image, and merge source images along the optimal seams. Since each pixel in the composite image comes from only one source image, the ghosting problems can be avoided. In mobile settings, we want a method that finds optimal seams quickly with using little memory, so that it can be applied for creating high-resolution panoramic images on mobile phones. Like [11] and [2], we also use dynamic programming to find optimal seams.

We want to merge the images on places where they differ the least. Suppose that *abcd* is the overlapping area between the current composite image I_c and the current source image S_c. I_i^o and S_i^o are the overlapping images in the area *abcd* of I_c and S_c respectively. We compute squared differences e between I_i^o and S_i^o as an error surface,

$$e = (I_c^o - S_c^o)^2. \tag{5}$$

We apply dynamic programming to find a minimal cost path through this surface. We scan the error surface row by row and compute a cumulative minimum squared difference E

$$E(h,w) = e(h,w) + \min(E(h-1,w-1), E(h-1,w), E(h-1,w+1)), \tag{6}$$

where $h = 2, \ldots, n_r$ and $w = 2, \ldots, n_c$ are the indices of the rows and columns of the error surface, respectively.

The optimal path m_c can be obtained by tracing back the paths with a minimal cost from bottom to top.

On the last row, the pixel with the minimum value is at the end (h_0, w_0) of the optimal path. On the previous row, the minimum $E(h_0-1,w)$, $w \in \{w_0-1, w_0, w_0+1\}$ denotes the position (h_0-1,w) of the optimal path at this row. Similarly, we can follow the path up one row at a time.

Fig. 4 shows the process of optimal seam finding with dynamic programming. Fig. 4 (a) and (b) are the overlapping areas of I_c and S_c, respectively. The error surface e shown in Fig. 4 (c) is computed as the squared differences between images Fig. 4 (a) and (b). Using that, the cumulative minimum squared difference E is computed and is shown in Fig. 4 (d). Fig. 4 (e) shows all possible paths. After tracing back with dynamic programming, we obtain the optimal path shown in Fig. 4 (f), along which the two images in (a) and (b) match best. We use that path as an optimal seam to create labeling.

We update the current composite image I_c by merging the current image S_c with the labeling information and continue the labeling process with the next source image. After all source images are processed, we obtain the final composite image. Since optimal seams are used in the image stitching process, the ghosting problems can be avoided. Fig. 3 (b) shows the result obtained by image stitching with optimal seam finding for the source images shown in Fig. 3 (a). From the result we can see that ghosting artifacts in the overlapping area have been removed by the optimal seam finding process. In this case the paths resulted in a copy of the moving person, other choices for the path might result in the left, right, or neither version of the person. In any case we avoided transparent copies, or paths splitting the person in two.

Color correction can improve quality of optimal seam finding and image labeling. We want to find a path where the images agree. This is more difficult to do if the colors of the two images disagree as much as in the sequence of **Fig. 5**. There is a moving object (car) in the scene, and we would like to find a path that does not intersect the car, as the other images do not contain it. However, on top left, where the images have not been color-corrected, the minimum difference path goes through the car. In top right, when the colors have been corrected before the path search, the path avoids the car, and a consistent panorama could be created.

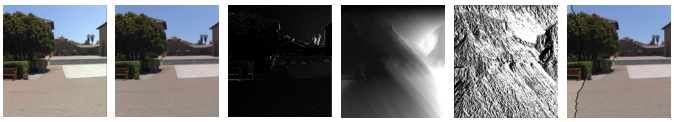


Fig. 4. Process of optimal seam finding with dynamic programming.

Fig. 5. Color-correction improves the seam quality.

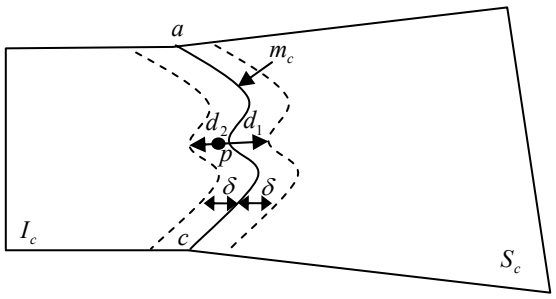


Fig. 6. Simple linear blending.

V. TRANSITION SMOOTHING WITH IMAGE BLENDING

Color correction reduces the differences between the images, which makes blending easier and faster. In our fast panorama stitching approach, we have two image blending processes that can be used.

A. Simple Linear Blending

For the source images that are similar in color and luminance after color correction, we perform a simple image blending on a band that is δ pixels wide on both sides of the seam, as shown in Fig. 6. The new color value of pixel p in the overlapping area can be calculated by a weighted combination of the corresponding pixels

$$P_{I_{c,new}}(p) = \frac{d_1^n P_{I_c}(p) + d_2^n P_{S_c}(p)}{d_1^n + d_2^n}, \tag{7}$$

where d_1 and d_2 are distances from pixel p to boundaries; $P_{I_{c,new}}(p)$ is the new color of pixel p; $P_{I_c}(p)$ is the color of

pixel p in image I_c; $P_{S_c}(p)$ is the color of pixel p in image S_c; different values of n result in different color transitions.

Linear blending is simple and computational and memory costs are low. However, moving objects in the blending band areas will cause ghosting artifacts. Furthermore when source images differ, linear blending is not enough to get rid of seams and stitching artifacts; more intensive blending is required.

B. Poisson Blending

Poisson blending is an intensive image blending approach that performs image blending in the gradient domain. In Poisson blending, we create a gradient vector field (G_x, G_y) with gradients of source images using the labeling obtained using optimal seams. In the sequential image stitching procedure, the gradient vector field is copied from the current source image S_c, up until the seam between it and the current panoramic image I_c (in Fig. 6 all the pixels of S_c to the right of the calculated seam). A divergence field $div(G)$ is then computed from the gradient vector field,

$$div(G) = \frac{\partial G_x}{\partial x} + \frac{\partial G_y}{\partial y}. \tag{8}$$

We use the divergence field as a guidance to construct a Poisson equation

$$\nabla^2 I(x,y) = div(G), \tag{9}$$

where ∇^2 is the Laplacian operator

$$\nabla^2 I(x,y) = \frac{\partial^2 I(x,y)}{\partial x^2} + \frac{\partial^2 I(x,y)}{\partial y^2}. \tag{10}$$

In practical implementation, we need to use the discrete form of Equation (9)

$$\begin{aligned} &I(x+1,y) + I(x-1,y) + I(x,y+1) + I(x,y-1) \\ &-4f(x,y) = G_x(x,y) - G_x(x-1,y) + G_y(x,y) \\ &- G_y(x,y-1) \end{aligned} \tag{11}$$

Equation (11) is a linear partial differential equation, which we solve by fixing the colors at the seam and solving new colors $I(x, y)$ over the gradient field. We can solve the equation using an iterative conjugate gradients solver.

Fig. 7 shows a comparison between the results created by simple linear blending and Poisson blending for the color-corrected source images shown in Fig. 2. The upper figure shows the result using simple linear blending. A faint seam can still be seen between the two source images. However, no visible seam can be observed in the result created by the Poisson blending shown on the bottom.

By comparison, the linear blending is simple and fast, but blending quality is low. The Poisson blending has higher quality; however it needs more computation and memory.

While color correction was crucial for good quality in linear image blending, it can also help to speed up the Poisson solver. Fig. 8 shows three source images with very different colors and luminance. The top row shows the results after 20 iterations of Poisson solver, on the left starting from the original inputs, on the right starting from color-corrected inputs. Hundreds of further iterations would be needed to obtain comparable results without color correction. With longer sequences differences become even more pronounced.

Fig. 7. Results of linear blending and Poisson blending.

Fig. 8. Improved blending quality and speed with color correction.

VI. IMPLEMENTATION

A sequential panorama stitching procedure is created with the fast image stitching approach. We have two implementations for the procedure: keep the full resolution panoramic image in memory; create a low-resolution panoramic image in memory for display and save the full resolution one to disk block by block while it is created.

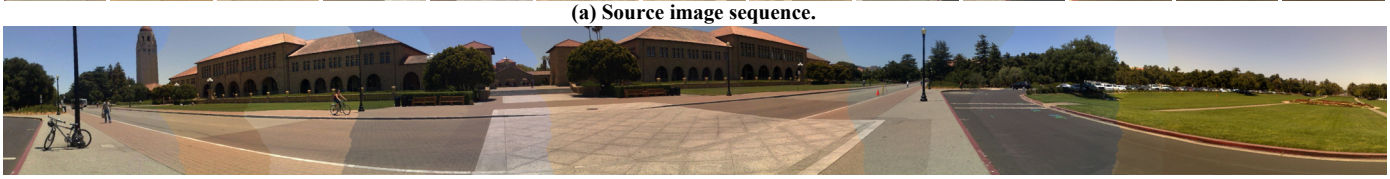
By comparison, the previous one is faster and more convenient for the viewing process. It can keep stitching with frames as long as there is enough memory for the full resolution panoramic image, the current source image, and some work arrays. The latter one has no limitation for the number of frames as long as there is enough memory for the low-resolution panoramic image, the current source image, and some work arrays. It needs to re-load the full resolution panoramic image for viewing. By comparing with the global image stitching [13], both implementations use much less memory. A comparing result is given in Section VII.B.

VII. EXAMPLES AND RESULT ANALYSIS

We have implemented the fast panorama stitching approach on mobile phones for creating high-resolution and high-quality panoramic images. We have tested it on both indoor and outdoor scenes and obtained good results. We present examples for various scenes, including long image sequences with source images with very different colors and luminance, and compare performance with other approaches to demonstrate advantages of our panorama stitching in processing speed and memory consumption.

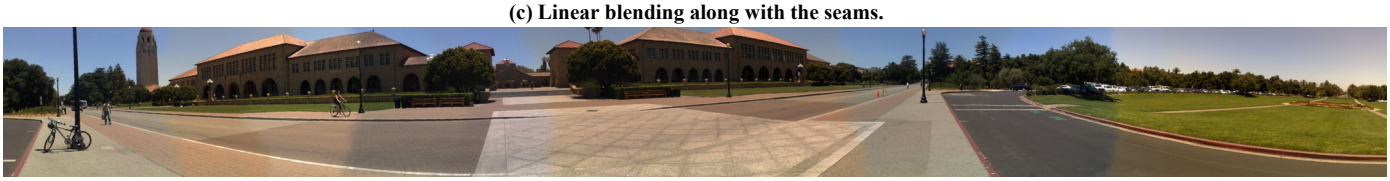
In this paper, the example applications and results are obtained on a mobile phone with a 332 MHz processor and 128 MB RAM. It can also be run on other mobile devices. In these applications, the size of source images is 1024×768. We have also applied it to larger source images, with good results.

(a) Source image sequence.



(b) Optimal seam finding.

(c) Linear blending along with the seams.



(d) Poisson blending with 5 iterations.

(e) Poisson blending with 150 iterations.

(f) Source images after color correction.

(g) Linear blending for source images after color correction.

(h) Poisson blending with 5 iterations for source images after color correction.

Fig. 9. Application to a long image sequence with very different colors and luminance in source images.

A. Long Image Sequences with very Different Colors

Fig. 9 shows an example of a long image sequence with images that have very different colors and intensities. With the results of this example, we can also demonstrate the performance of each process in the approach.

Fig. 9 (a) shows the original source images in the image sequence. There are 13 source images with very different colors and luminance in the image sequence. While it is captured, some objects move in the scene. The different colors and luminance between the source images are caused by the use of automated settings of the camera.

Fig. 9 (b) shows the composite image obtained by optimal seam finding. From the result we can see that the optimal seam finding process in our panorama stitching can find the best way to label images and merge them to a composite image. Although there are moving objects in overlapping areas of source images, there is no ghosting or blurring problems in the composite image. The use of dynamic programming for optimal seam finding is one of the main reasons why the proposed panorama stitching works fast.

Fig. 9 (c) shows the result created by the simple linear blending for the composite image obtained by optimal seam finding. From the result we can see that the color differences across the optimal seams are reduced to some extent. This means that the simple linear blending process can smooth color transitions across the seams. The processing speed of the blending is very fast. However, since the source images are very different in colors and luminance, the color differences in the whole composite image still can be seen. Other processing is needed to further reduce the differences.

Fig. 10. Panoramic image created by the approach in [2].

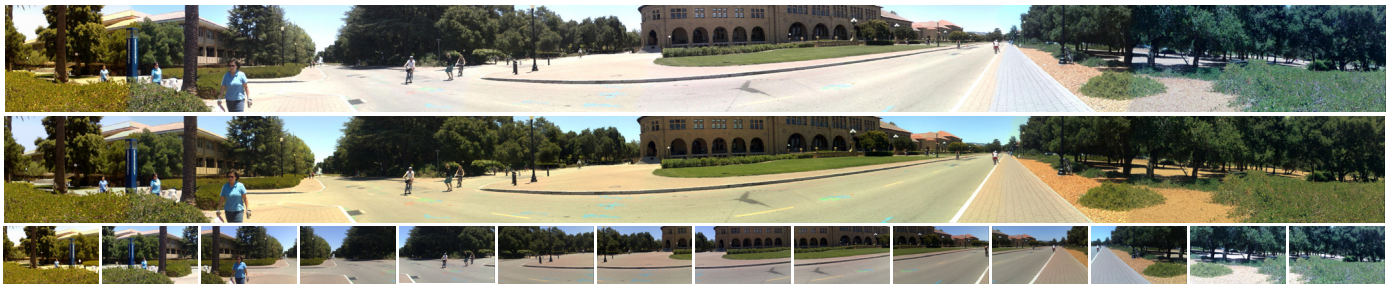


Fig. 11. Comparison of color correction approaches in [2] and in the proposed fast panorama stitching approach.

Fig. 9 (d) shows a result produced by Poisson blending in which the linear solver uses 5 iterations. From the result we can see that the effect of the blending is almost the same as the simple linear blending shown in Fig. 9 (c). It is still far away from a satisfying result. The color differences in the composite image can still be seen clearly.

Fig. 9 (e) shows a result obtained by Poisson blending after 150 iterations. The result is improved much compared to the result shown in Fig. 9 (d). This means that much more computation is needed to obtain a better result. However, the result is still not satisfying. We can still see color discontinuity in the composite image, especially on the right side.

Fig. 9 (f) shows the source images after color correction. From the result we can see that the color correction process can reduce differences in colors and luminance between two images and adjust colors globally in the whole image sequence. Although the original source images are very different, the differences are smoothed after color correction. Also, there are no pixel saturation artifacts after the color correction process. We can see that the performance of the color correction process is very satisfying.

Fig. 9 (g) shows a result created by the simple linear blending process after color correction. As we can see, combination of color correction with the simple linear blending can produce very satisfying panoramic images. Both processes of color correction and linear blending are simple and use little memory. The combination is suitable for mobile implementation and applications.

Fig. 9 (h) shows a result produced by Poisson blending with the source images after color correction. We can see that there are no visible artifacts. In this case, Poisson blending still uses 5 solver iterations, however, the result is much better than in Fig. 9 (e), which uses 150 iterations for the source images that are not processed by color correction. Again, the conclusion is that color correction can improve Poisson blending quality and speed up the processing speed. The combination makes Poisson blending much more suitable for mobile devices.

From the evaluation of this example we can see that each process of the proposed panorama stitching approach functions well. The approach can produce high-quality and high-resolution panoramic images on mobile phones. It can handle source images in long image sequences with very different colors and luminance.

B. Comparison with Other Approaches

Fig. 10 shows a panoramic image created by the approach proposed in [2] with source images shown in Fig. 9 (a). From the result we can see that color differences and seams between source images can be seen clearly. Since the source images are very different in colors and luminance, the color correction approach can not remove the differences completely and the simple band-linear blending can not smooth the color transitions, so that a low-quality panoramic image is obtained. Actually, this is one of the main disadvantages of the panorama stitching approach in [2]. It can not handle long image sequences with source images in very different colors and luminance. On the other hand, for same image sequence, the proposed fast panorama stitching approach can produce high-quality panoramic images shown in Fig. 9 (g) and (h) due to better color correction and image blending procedures. In general, the proposed approach can handle this kind of image sequences very effectively.

Fig. 11 shows a comparison of color correction results between the approach in [2] and the proposed approach. In this case, there are 14 source images with very different colors and luminance shown in Fig. 11 (bottom). Fig. 11 (top) shows the panoramic image created with the color correction in [2]. From the result we can see that the color correction approach does not work well. The color differences could not be removed. There is a main problem in this result that a large part of the pixels are saturated after color correction. Most details such as in the sky and road in this result are lost. Fig. 11 (middle) shows the panoramic image created with the color correction in the proposed approach. From the result we can see that all details are kept and pixels are not saturated after color correction. Colors in the whole panoramic image are very natural. Color transitions are smoothed. The good color correction promises to obtain high-quality panoramic images. Furthermore, Poisson blending can further improve quality of the final result.

Fig. 12. Panoramic image produced by the fast panorama stitching with 7 source images in an indoor scene with moving objects.

Fig. 13. Panoramic image produced by the fast panorama stitching with 8 source images in an outdoor scene with moving objects.

Fig. 14. A panoramic image created by our fast panorama stitching with 17 1024×768 source images on mobile phones.

TABLE I
COMPARISON OF MEMORY CONSUMPTION OF IMAGE STITCHING IN [13] AND THE PROPOSED APPROACH

A	2	3	4	5	6	7	8	9	10
B	11.4	13.3	15.1	16.9	19	20.5	21.4	23.4	24.2
C	10.1	10.8	11.4	12.2	13	13.6	13.7	14.7	15.0

We have compared memory consumptions between the proposed sequential panorama stitching with the global panorama stitching in [13] which needs to keep all source images in memory for global optimization during image stitching. The result is shown in Table I. In the table, row A means the number of source images used, B shows the memory consumption using global panorama stitching, and C shows the memory consumption of sequential panorama stitching. The unit of memory consumption is MB. From the results we can see that the more source images in panorama stitching, the more memory the sequential stitching saves. In this comparison, both implementations keep full panoramic images in memory during panorama stitching.

C. Image Stitching of an Indoor Scene

Fig. 12 shows an example of an indoor scene with 7 source images. The result created by the proposed fast panorama stitching is shown on the top of the figure. The stitching process takes 19 seconds and the graph cut approach [13] takes 672 seconds, about 35 times longer.

We can also notice some other aspects. Although the people in the scene are moving during the capture of the sequence, the stitching process finds good seams and avoids ghosting and blurring problems caused by these moving objects. Although there are some differences of the source images in colors and luminance, they are removed after color correction and image blending in the resulting panoramic image. The color transitions are smoothed in the final results.

D. Image Stitching of an Outdoor Scene

The outdoor image sequence in Fig. 13 (bottom) includes eight source images that are stitched together to create a panoramic image. Fig. 13 (top) shows the result created by the fast panorama stitching approach. The stitching takes 23 seconds and the graph cut [13] takes 756 seconds, about 32 times longer. Also here we find good seams, and selecting single input image per output pixel helps to avoid ghosting problems due to object motion.

E. Creating 360◦ Panoramas with very Long Image Sequences

Fig. 14 shows a 360° panoramic image. The top shows the created panoramic image and the bottom shows the 17 source images. From the image sequence we can see that the source images are very different in colors and luminance and there are some moving objects in the scene while the image sequence is captured. However, the approach still produces a high-quality panoramic image.

The panorama stitching process takes 34 seconds and again is much faster than the commonly used graph cut approach. According to our tests, the longer the image sequences, the greater the speed advantage of the fast panorama stitching is. Since the fast panorama stitching is a sequential image stitching procedure, it only needs to keep the panoramic image and the current source image in memory. As long as there is enough memory for the final panorama and the current source image, the approach does not care how many source images are processed. Fast processing speed and low memory consumption are the main advantages of the proposed approach, both very important in a mobile implementation.

Our approach has been tested with many image sequences with different cases on different types of mobile phones and it performs well.

VIII. DISCUSSION AND CONCLUSIONS

A fast panorama stitching approach that uses little memory is developed and implemented on mobile phones for creating high-resolution and high-quality panoramic images. It has been tested with different image sequences captured under different lighting conditions. It is much faster than the graph cut approach. The fast stitching approach can be applied to create high-resolution panoramic images with large source images as long as the system has enough memory for the final panorama and the current processing source image.

The fast speed of the proposed approach is mainly due to the fast labeling approach created with dynamic programming. It is very simple to implement. After the overlap between two images is located, an error surface is created by computing the squared differences of colors in the overlapping area. A low-cost path where the image values agree is found by dynamic programming. The path is used as the optimal seam to create labeling, and the two images can be cut along the seam and merged together. Labeling allows us also to avoid ghosting when objects move as the images are captured.

Two image blending processes can be used in this fast panorama stitching approach. When source images are sufficiently similar in colors after color correction, a simple and fast linear blending suffices. When source images are too different for the simple linear blending, a Poisson blending removes visible seams. Applying color correction helps also Poisson solver to find a good solution faster.

A sequential panorama stitching procedure is created and integrated with color correction, fast labeling, and image blending to create panoramic images. The integration allows us to create high-resolution panoramic images from several large source images quickly using little memory.

Future work includes speeding up the Poisson blending process and reducing its memory consumption.

REFERENCES

[1] Y. Xiong and K. Pulli, “Mask based image blending approach and its applications on mobile devices,” in SPIE Multispectral Image Processing and Pattern Recognition (MIPPR), 2009.

[2] S. Ha, H. Koo, S. Lee, N. Cho, and S. Kim, “Panorama mosaic optimization for mobile camera systems,” IEEE Transactions on, Consumer Electronics, vol. 53, no. 4, pp. 1217–1225, Nov. 2007.

[3] S. Ha, S. Lee, N. Cho, S. Kim, B. Son, "Embedded panoramic mosaic system using auto-shot interface," IEEE Transactions on Consumer Electronics, Vol. 54, No. 1, pp.16-24, Feb. 2008.

[4] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen, “Interactive digital photomontage,” ACM Trans. Graph, vol. 23, pp. 294–302, 2004.

[5] A. Levin, A. Zomet, S. Peleg, and Y. Weiss, “Seamless image stitching in the gradient domain,” in European Conference on Computer Vision (ECCV), 2004, pp. 377–389.

[6] P. Pérez, M. Gangnet, and A. Blake, “Poisson image editing,” ACM Trans. Graph., vol. 22, no. 3, pp. 313–318, 2003.

[7] J. Jia, J. Sun, C.-K. Tang, and H.-Y. Shum, “Drag-and-drop pasting,” in ACM SIGGRAPH, 2006, pp. 631–637.

[8] Z. Farbman, G. Hoffer, Y. Lipman, D. Cohen-Or, and D. Lischinski, “Coordinates for instant image cloning,” ACM Trans. Graph., vol. 28, no. 3, pp. 1–9, 2009.

[9] N. Gracias, M. Mahoor, S. Negahdaripour, and A. Gleason, “Fast image blending using watersheds and graph cuts,” Image Vision Comput., vol. 27, no. 5, pp. 597–607, 2009.

[10] D. L. Milgram, “Computer methods for creating photomosaics,” IEEE Trans. Comput., vol. 24, no. 11, 1975, pp. 1113–1119.

[11] A. A. Efros and W. T. Freeman, “Image quilting for texture synthesis and transfer,” in ACM SIGGRAPH, 2001, pp. 341–346.

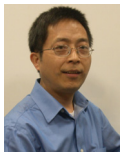
[12] J. Davis, “Mosaics of scenes with moving objects,” in IEEE Conference on CVPR, 1998, pp. 354–360.

[13] Y. Xiong and K. Pulli, “Gradient domain image blending and implementation on mobile devices,” in International Conference on Mobile Computing, Applications, and Services (MobiCase), 2009.

[14] V. Kolmogorov and R. Zabih, “What energy functions can be minimized via graph cuts,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, pp. 65–81, 2004.

[15] Y. Xiong and K. Pulli, “Sequential image stitching for mobile panorama,” in IEEE International Conference on Information, Communications and Signal Processing (ICICS), 2009.

BIOGRAPHIES



Yingen Xiong works at Nokia Research Center. His research interest areas include computer vision, pattern recognition, and computational photography. Previously he was a research professor in Virginia Polytechnic Institute and State University and Wright State University. He received PhD degree from Nanjing University of Aeronautics and Astronautics.

Kari Pulli is a research fellow at Nokia Research Center. He has been an active contributor to several mobile graphics standards and recently wrote a book about mobile 3D graphics. Pulli received a PhD in computer science from University of Washington and an MBA from University of Oulu. Contact him at kari.pulli@nokia.com.