Fast Image Stitching and Editing for Panorama Painting on Mobile Phones

Yingen Xiong and Kari Pulli
Nokia Research Center
955 Page Mill Road, Palo Alto, CA 94304, USA
{yingen.xiong, kari.pulli}@nokia.com

Abstract

We propose a fast image stitching and editing approach for panorama painting and have implemented it on a mobile phone. A seam search using dynamic programming finds a good boundary along which to merge the source images, and a transition smoothing process using instant image cloning removes merging artifacts, yielding high-quality panoramic images. A sequential image stitching procedure requires only keeping the current source image in memory during image stitching, which saves memory and is more suitable to mobile phone implementations.

1. Introduction

We present an efficient approach for creating and editing panoramas on mobile phones. A user can paint a scene with a camera phone and see a panoramic image created for the scene immediately. By painting we mean a more complicated sweep than a simple left-to-right sweep, for example first sweeping one row of images from left-to-right, and then another row below it right to left. The user can also edit the panoramic image by adding her picture and other objects in.

1.1. Related work

The two main categories of current image stitching approaches are transition smoothing and optimal seam finding. Transition smoothing approaches merge source images by reducing color differences between the images. Color transitions in composite images are smoothed to remove stitching artifacts. Recently, gradient domain image blending approaches [9, 12, 8, 11, 7] have been applied to image stitching and editing. A new gradient vector field is created from the gradients of source images for constructing a Poisson equation. A high-quality composite image can be created from the new gradient vector field by solving the Poisson equation with boundary conditions. However, computational and memory costs are high, since a large system of linear equations needs to be solved. Farbman et al. [3] created an instant image cloning method that uses mean value coordinates to distribute color differences on the seam over all the pixels of the image to be blended. This approach is very simple and effective, and it speeds up the image blending process and reduces memory consumption.

Optimal seam finding approaches search for good seams along which to paste and composite images. Most methods look for paths where the overlapping images have minimal differences, though other optimization criteria (such as preferring strong edges) are possible. Based on the seams each output pixel gets a label specifying which input image contributes to it. Graph cut [4, 1, 14] is a very general labeling approach that can be used with an arbitrary image stitching order and when multiple images overlap in the same area. However, computational and memory costs are high, the computation is iterative and all the images have to be kept in the memory at the same time. Dynamic programming [13, 5, 6, 10, 2] allows a much faster labeling process, and uses little memory, but it is typically used only in 1D labeling, with which we mean a sequential left-to-right or right-to-left stitching of images. In this paper, we use it for fast 2D stitching, where the images can have also upper and lower (and diagonal) neighbors, not just left and right neighbors.

Combined optimal seam finding and transition smoothing has been used for panorama stitching both on desktop computers [1] and mobile devices [14]. Both of these systems find seams using graph cut, merge source images along the seams, and smooth color transitions using Poisson blending, requiring a lot of computation and memory.

1.2. Contributions

In this work, we propose a fast image stitching and editing approach for mobile panorama painting. It can be used for creating and editing high-resolution and high-quality panoramic images. In this approach, a sequential image stitching procedure is created by combining fast seam finding and efficient transition smoothing for creating high-quality panoramic images with as few computational re-

978-1-4244-7030-3/10/$26.00 ©2010 IEEE

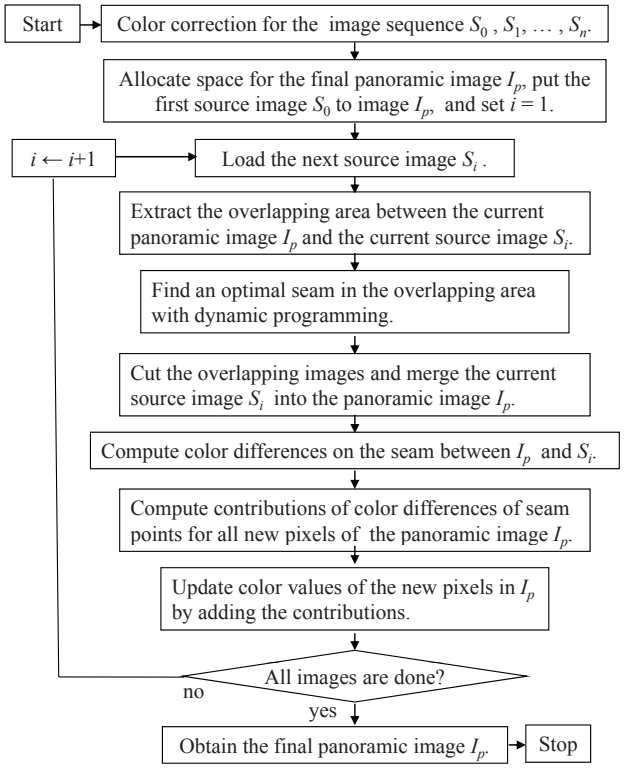

Figure 1. Work flow of the panorama painting approach.

sources and as little memory as possible. Good performance and results have been obtained in different tests on a mobile camera phone.

Our contribution is to find the seams using dynamic programming instead of graph cut and blend images with modified instant cloning instead of Poisson blending (c.f. [14]), as well as process the images one at a time to minimize RAM consumption and supporting 2D sweeps instead of 1D sweeps (c.f. [13]). We compare our method to the previous approaches to demonstrate its advantages. We also demonstrate how the same approach can be used to post-process panoramas by adding new objects into the image.

1.3. Organization of the paper

In the following sections we introduce our work flow (2), explain the details of the image stitching and editing approach for panorama painting (3), discuss applications and analyze results (4), and summarize the paper (5).

2. Summary of the approach

Figure 1 shows the work flow of our fast image stitching and editing system. The sequential image stitching procedure starts by performing color and luminance compensation for the image sequence to reduce color differences and smooth color transitions between source images, allocating

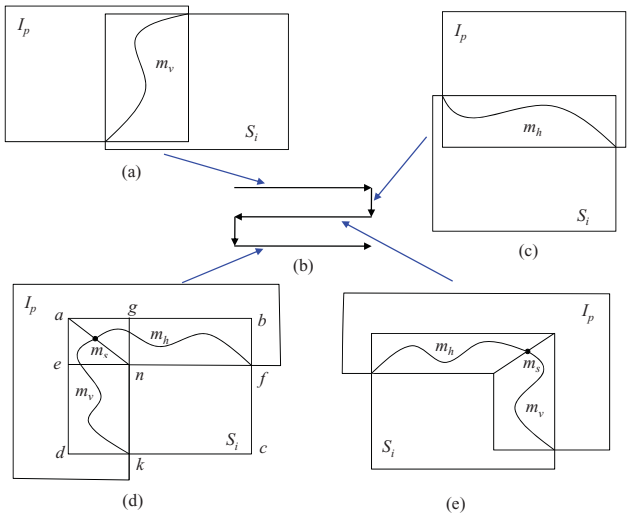

Figure 2. Camera motion and optimal seams.

memory for the final panorama I_p, and initializing it with the first source image S_0.

Then the next source image S_i is loaded into memory. Using the registration transformations we detect the overlapping area between it and the current panorama I_p, and find a good seam where the images have similar color values within this area using dynamic programming. In the 2D panorama painting case, there are three kinds of seams: vertical, horizontal, and composite seams. We cut the overlapping area along the seam and merge the current source image to the current panorama.

We next blend the images to smoothen color transitions between them. We keep the pixels within the previous panorama unchanged and modify the pixel colors in the new source image that is being added. In order to do this, we compute color differences on the seam between the two images and distribute the color differences to all pixels in the new source image to create a smooth transition. We update color values of the new pixels by adding the interpolated color differences from the seam to all individual pixels.

The process is repeated until all source images have been added, creating the final panoramic image I_p.

3. Fast image stitching and editing

3.1. Seam finding

Figure 2 (b) shows a possible camera motion to capture a 2D sweep of images, and Figures 2 (a), (c), (d), and (e) show possible seams when stitching the source images captured by the camera with this kind of motion.

In the first row, the camera moves horizontally. We need to create vertical seams as shown in Figure 2 (a) to stitch the source images together. In a general case shown in Figure 2 (a), the seam has to start and end at locations where the new image S_i intersects the current panorama I_p. If the images

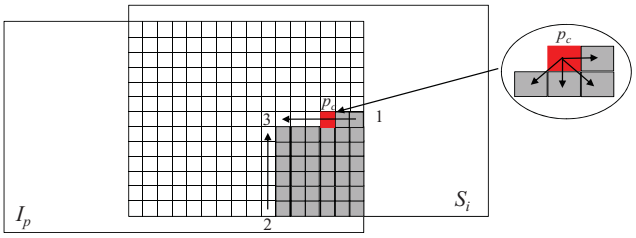

Figure 3. Optimal seam finding with dynamic programming.

have been cropped so that their top and bottom rows align, as was the case in [13], the path may start from any column of the top row within the overlap area and it may end at any column at the bottom row of the overlap area. When the camera moves vertically, we need to create horizontal seams (Figure 2 (c)) for image stitching. When the camera moves horizontally from left to right in a row other then the first row, a seam as shown in Figure 2 (d) is created. In a similar way, when the camera moves horizontally from right to left in a row other then the first row, we need to find a seam shown in Figure 2 (e). Again, the seam has to start and end at locations where the new image intersects the current panorama.

Like [13], we search for optimal seams in the overlapping areas of the source images using dynamic programming, cut the overlapping images, and merge them together. Different from [13], we also consider 2D image stitching cases.

Figure 3 shows the general case of two images overlapping. The seam between the two images has to start and end at the intersections of the images, and if the images are of the same size, the intersections are on the diagonal corners of the overlap area. Given the current source image S_i and the current panoramic image I_p and their overlap areas S_i^o and I_p^o, we want to find a seam where they match the best, cut the images along with the seam, and combine them.

In order to do this, we compute a squared difference d between S_i^o and I_p^o as an error surface:

$$d = (I_c^o - S_c^o)^2. \quad (1)$$

We apply dynamic programming optimization to find a minimal cost path through this surface. We scan the error surface and compute a cumulative error surface D_v. In the case of Figure 3 we start from the lower right corner. At that corner pixel we have simply $D_v(w,h) = d(w,h)$. We grow D_v by first extending it above by one row, then by one column on the left, and finally by filling the new diagonal pixel. In the general case denoted in Figure 3, when we are processing a new pixel on the top row at (i,j), we get

$$\begin{aligned} D_v(i,j) \quad = \quad & d(i,j) + \min(D_v(i+1,j), \quad (2) \\ & D_v(i-1,j+1), D_v(i,j+1), \\ & D_v(i+1,j+1)). \end{aligned}$$

Figure 4. Image stitching with source images in different colors.

The left columns are grown in a similar fashion, and the diagonal element chooses the minimum of the three existing neighbors (below, right, and diagonally below right).

Once all of the overlapping area has been filled with the cumulative error surface, minimum path can be obtained by starting from the upper left corner, and following the path of least cumulative error all the way down to the starting point.

The case shown in Figure 2 (d) is more complicated. There the seam is composed of two seams, a vertical seam m_v and a horizontal seam m_h, and we call it a composite seam. As shown in Figure 2 (d), we can divide the overlapping area as $agne$, $agkd$, and $abfe$. We need to find a vertical seam for the area $agkd$, a horizontal seam for the area $abfe$. The two seams meet in the shared area $agne$.

We first calculate the cumulative error surface D'_v for the block $agkd$ starting from the intersecting point k. Then we calculate another cumulative error surface D'_h for the block $abfe$, starting from the intersection point f. We add D'_v and D'_h on the diagonal line in the shared area $agne$ and find the point m_s on that line where the summed cumulative error is at minimum. Finally, we find the horizontal seam m_h and vertical seam m_v by tracing back from left to right and top to bottom starting from the optimal point m_s.

For other possible camera motions during panorama painting, the seams can be created using similar ways as above by rotating or flipping the error surface in Equation 1 and rotating or flipping the results back.

3.2. Image blending

When source images differ in colors and luminance, the seams between the source images in the composite image may remain visible. Figure 4 (top left) shows one of the examples. In the case, there are three source images in different colors shown in Figure 4 (bottom), especially between the second and third images. Figure 4 (top left) shows a stitching result without any color processing. From the result we can see the color differences and seams clearly. An image blending process is needed to reduce color differences between source images, smoothing color transitions, and removing stitching artifacts.

Poisson blending [11] is an intensive image blending ap-

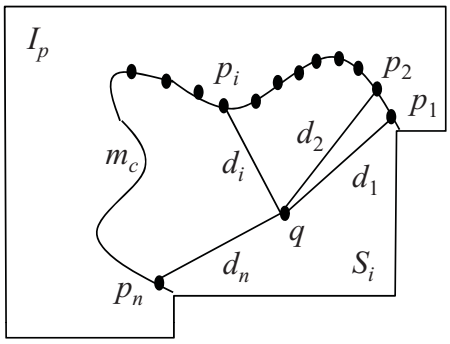

Figure 5. Image blending.

proach. It performs image blending in the gradient domain and can provide high-quality blended images.

Farbman et al. [3] showed that one can obtain as good results as with Poisson blending using a much simpler approach. Figure 5 shows how we use their method to blend the current image S_i to the current panoramic image I_p. We keep the color values on the composite seam m_c as they were in the previous version of the panorama I_p and modify colors of pixels in the current source image S_i so that no visible boundary remains. We do this by first computing the color differences of pixels on the seam between the current panorama and the new source image, and then distribute and add the color differences to the rest of the source image.

Let $p_1, p_2, \ldots, p_n$ be the n points on the seam m_c, $P_1, P_2, \ldots, P_n$ be the color differences at those points between the current panoramic and source image, and q be a pixel of the current source image S_i. We then interpolate the color differences at pixel q by

$$P(q) = \sum_{i=1}^{n} w_i(q) P(p_i), \tag{3}$$

where the weights are the inverse coordinate distances to the boundary pixels, normalized so that they sum up to 1:

$$w_i(q) = \frac{1/||p_i - q||}{\sum_{j=1}^{n} 1/||p_j - q||}. \tag{4}$$

Note that our weights are much simpler than the mean value coordinates introduced by Farbman et al. [3]. We noticed that as long as the boundary is not very convoluted and it is evenly sampled, the tangent terms of the mean value coordinates are not needed to create a smooth interpolation, and omitting them reduces the required amount of computation. We also accelerate the method by only calculating accurate weights at pixels increasingly far away from the boundary and (bi)linearly interpolating the color differences between those pixels, however we use quadtrees instead of a triangular mesh. Finally, we add the interpolated color differences to pixels over the new source image S_i.

Figure 4 (top right) shows a panoramic image created by the image blending process with the same source images shown in Figure 4 (bottom). From the result we can see that the color differences between the source images are reduced and color transitions in the panoramic image are smoothed. All stitching artifacts are removed. The blending process is very efficient even if the source images are very different in colors and luminance. We have tested it with more source images captured in different conditions. Good performance has been obtained.

We use the same approach for panoramic image editing. We can copy objects and paste them to panoramic images by blending the differences on the boundaries of the pasted objects over the objects.

4. Applications and result analysis

The fast image stitching and editing approach is integrated into a sequential procedure and implemented in mobile phones for panorama painting. It can produce high-resolution and high-quality panoramic images on mobile devices. It has been tested and applied under different conditions, performing well in both indoor and outdoor scenes. We compare the results obtained by the fast image stitching approach and the commonly used image stitching using graph cut [14] and a fast Poisson blending [12] to demonstrate advantages of the fast stitching approach in processing speed and memory consumption. The results in this section were obtained on a Nokia N95 8GB mobile phone with an ARM 11 332 MHz processor and 128 MB RAM, the size of source images was 1024×768, but it also works on larger images.

4.1. Outdoor scenes

Figure 6 shows an example outdoor scene with 6 source images. For tall and wide scenes such as tall gates and buildings, a 1D sweep is not enough to capture the whole scene, a 2D horizontal and vertical sweep is required. From the result we can see that our fast stitching can produce high-quality panoramic images for outdoor scenes. Although the luminances between the source images differ a lot. In effect, as the images were taken with auto-exposure, and the last three images with less sky and more ground have a longer exposure, we get an automatic tone-mapped higher dynamic range than we would have gotten with a single image. Notice that in the last input images the sky was completely saturated while the results retains also sky texture. Also notice that although the people in the scene are moving during the capture, avoiding alpha blending via labeling and fast cloning avoids ghosting problems caused by moving objects.

The fast stitching approach takes 59 seconds while our implementation of the method in [14] which uses graph cut

Figure 6. Panorama painting for a tall gate in an outdoor scene.

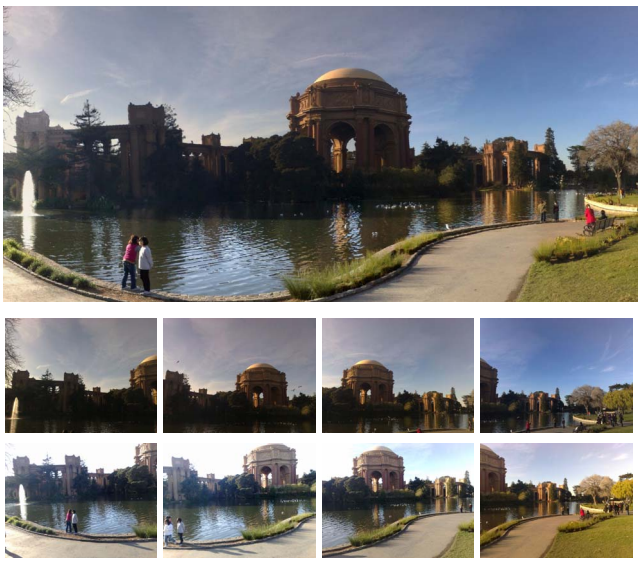

Figure 7. Outdoor scene with very different colors in input images.

and spline Poisson blending, took 1015 seconds, 17 times longer. In terms of memory consumption, the graph cut approach needs to keep all source images in memory for global optimization to find optimal seams in the overlapping areas. However, the fast stitching approach only needs to keep the current source image in memory. In this application, the graph cut keeps all six source images in memory, while the fast stitching approach only keeps one source image for processing.

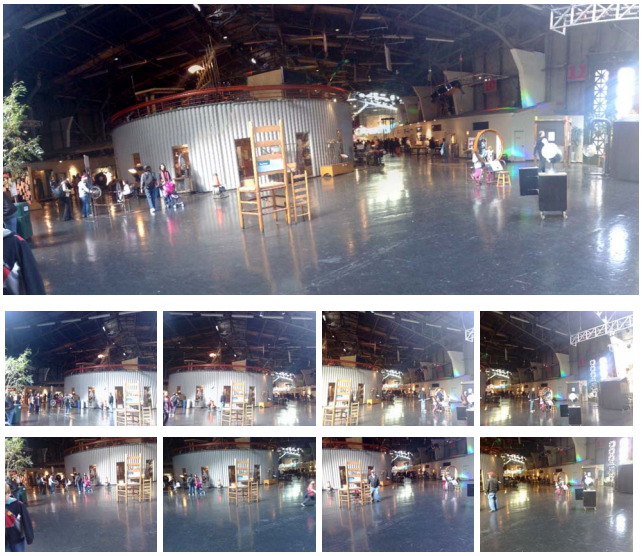

Figure 8. Panorama painting for an indoor scene.

Figure 7 shows another scene with even more extreme changes in the luminance and colors in the input images.

4.2. Indoor scene

Figure 8 shows an example indoor scene with 8 source images in two rows captured by 2D camera motion inside a science museum. We can see that the colors of the source images are very different, especially on the floor. Although lighting inside the building is difficult and there are more reflections, our approach can handle these problems and produce satisfying panoramic images.

The fast stitching approach takes 104 seconds in the panorama construction, while the graph cut approach takes 1575 seconds, 15 times slower.

4.3. Panorama painting and editing

Figure 9 shows an application of panorama painting and editing. In this case, the building group is very tall and wide, so neither horizontal or vertical 1D sweep suffices to capture the scene. Instead, we capture a 3×2 2D image pattern. The top image shows the nice result created by the fast image stitching approach.

For the beautiful scene you visit, you might also want to put yourself into the picture. This can be done by cutting out your picture from other pictures and add it to the panoramic image using the same image cloning approach we use to blend in new inputs images one at a time. The middle image shows an example, it looks as if the person were there when the panorama was captured.

Figure 9. Panorama painting and editing.

5. Discussion and conclusions

In this paper, we have proposed a fast image stitching and editing approach for panorama painting and implemented it on mobile phones to create high-resolution and high-quality panoramic images.

The approach includes fast image labeling and efficient image blending. The fast image labeling is performed by dynamic programming. An error surface is created for each overlapping area and a minimal cost path can be found through the error surface. We use it as an optimal seam to cut the overlapping images and merge them together. Three kinds of seams can be created for 2D stitching. In order to obtain high-quality panoramic images, we apply a fast method that yields results very similar to Poisson blending for transition smoothing.

A sequential procedure is created and integrated with the fast image stitching approach. During image stitching, only the currently processed source image needs to be kept in memory, whereas the graph cut approach needs to keep all source images in memory to find a globally optimal solution. The combination of the sequential procedure and the fast image stitching saves memory resources.

Another main advantage is fast processing speed. By comparing with stitching using graph cut and Poisson blending, the proposed approach is much faster. In the case of six source images, it takes about 59 seconds and is about 17 times faster than the approach using graph cut and Poisson blending. According to our tests, the more source images in panorama construction, the more advantages of the fast stitching approach in speed and memory consumption.

References

[1] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen. Interactive digital photomontage. *ACM Trans. Graph*, 23:294–302, 2004.

[2] J. Davis. Mosaics of scenes with moving objects. In *CVPR*, pages 354–360, 1998.

[3] Z. Farbman, G. Hoffer, Y. Lipman, D. Cohen-Or, and D. Lischinski. Coordinates for instant image cloning. *ACM Trans. Graph.*, 28(3):1–9, 2009.

[4] N. Gracias, M. Mahoor, S. Negahdaripour, and A. Gleason. Fast image blending using watersheds and graph cuts. *Image Vision Comput.*, 27(5):597–607, 2009.

[5] S. J. Ha, H. Koo, S. H. Lee, N. I. Cho, and S. K. Kim. Panorama mosaic optimization for mobile camera systems. *Consumer Electronics, IEEE Transactions on*, 53(4):1217–1225, Nov. 2007.

[6] S. J. Ha, S. H. Lee, N. I. Cho, S. K. Kim, and B. Son. Embedded panoramic mosaic system using auto-shot interface. *Consumer Electronics, IEEE Transactions on*, 54(1):16–24, Feb. 2008.

[7] J. Jia, J. Sun, C.-K. Tang, and H.-Y. Shum. Drag-and-drop pasting. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 631–637, New York, NY, USA, 2006. ACM.

[8] M. Kazhdan and H. Hoppe. Streaming multigrid for gradient-domain operations on large images. *ACM Trans. Graph.*, 27(3):1–10, 2008.

[9] A. Levin, A. Zomet, S. Peleg, and Y. Weiss. Seamless image stitching in the gradient domain. In *ECCV*, pages 377–389, 2004.

[10] D. L. Milgram. Computer methods for creating photomosaics. *IEEE Trans. Comput.*, 24(11):1113–1119, 1975.

[11] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318, 2003.

[12] R. Szeliski, M. Uyttendaele, and D. Steedly. Fast poisson blending using multi-splines. In *Technical Report MSR-TR-2008-58, Microsoft Research*, 2008.

[13] Y. Xiong and K. Pulli. Fast image labeling for creating high-resolution panoramic images on mobile devices. In *IEEE Int. Symp. on Multimedia*, 2009.

[14] Y. Xiong and K. Pulli. Gradient domain image blending and implementation on mobile devices. In *MobiCase*, 2009.