Mobile Panoramic Imaging System

Kari Pulli, Marius Tico, Yingen Xiong
Nokia Research Center
955 Page Mill Road, Palo Alto, CA, USA
firstname.lastname@nokia.com

Abstract

We introduce a mobile system for creating high-resolution panoramic images. The user can rotate the camera around an arbitrary axis to create a 2D sweep and see a miniature preview panorama in real-time. The system tracks camera motion and automatically captures high-resolution images and generates a high-quality wide-view panoramic image. We employ a coarse-to-fine method for high-quality registration, and a seam-finding method to remove ghosting effects due to moving objects. The proposed system has been tested on several camera phones, and the tests reveal that the system can efficiently provide a high-quality panoramic image in spite of the low computational power and memory available in such devices.

1. Introduction

A panorama can provide a stunning wide-angle representation of the scene beyond a normal photograph. However, capturing a good panorama can be a painstaking process. Typically, the user has to fix the camera on a tripod to capture the images. Those images are then stitched into a high-resolution panorama via computationally demanding PC software, such as Adobe Photoshop. This two-step procedure makes it difficult to infer the appearance and quality of the final result during acquisition. This disadvantage can be overcome by mobile implementations [4, 6, 7]. However, designing a high-quality mobile panoramic system is challenging due to limited computational resources on such devices. Therefore, previous methods often use low-resolution input images or impose strong assumptions on the camera motion and input data, in order to reduce the computational effort needed for panorama creation. For example, most systems can only capture images along a one-dimensional swipe, or require the user to manually align the images to simplify or bypass the registration step. Also, they all use fast methods to blend the captured images, and hence they cannot handle moving objects, which may result in serious ghosting effects.

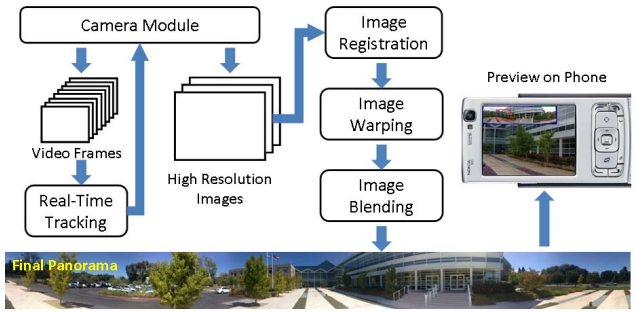

Figure 1. System flow of the proposed solution. The final 360° panorama on the bottom is generated from 18 input images.

2. The proposed system

We propose a complete solution for mobile panoramic imaging that can handle an unlimited angle of view (both horizontally and vertically) and moving objects, and generate results comparable to PC software. The system flow is shown in Fig. 1. Our system allows the user to rotate the camera arbitrarily by tracking camera motion in real-time, and automatically determines when to take high-resolution images. Typically, panorama images are one-dimensional (1D) in the sense that the component images are captured by rotating the camera from left to right (or vice-versa) in the horizontal plane, around the vertical axis that passes through optical center of the camera [6]. Our application supports also two-dimensional (2D) panoramas in the sense that it allows the user to capture a wider area of the scene by rotating the camera not only around the vertical axis, but around any axis [3, 1].

The high-resolution images are then registered on a spherical manifold. After warping, all images are stitched together using a high-quality blending method that removes artifacts. The user can then interactively examine the resulting panorama on the phone.

2.1. Image Capture

Camera motion is estimated by tracking consecutive low-resolution (320×240) viewfinder frames captured at

Figure 2. As the user moves the camera, the motion is tracked in real time and the current view is shown on the minimap on left top. The next image is automatically captured when the motion is large enough.

30 frames per second [1]. During capture, a miniature panorama generated from previously captured images is displayed to help the user to determine the areas that have already been captured, and a moving box indicates the position of the current frame. This allows the user to easily decide the size and shape of panorama (see Fig. 2). When the camera motion with respect to the previous image exceeds a threshold, a high-resolution image is automatically captured.

2.2. Identification of overlapping input images

Assuming that the input images are captured in an arbitrary order, the first processing step consists of identifying the images that overlap. To do this we consider the rough relative position of each image in the final panorama, provided by the tracker algorithm. Knowing the image size and positions, we can determine the overlapping area between each image pair. Two images are considered to overlap if their overlapping area exceeds a certain percentage of the image area.

2.3. Pair-wise image registration

Image registration of overlapping high-resolution image frames is essential for ensuring an accurate and seamless representation of the panoramic scene. The main approaches to image registration can be classified in two categories: feature-based and image-based methods [14]. Feature-based methods rely on determining the correct correspondences between visual features extracted from the images. In some applications, the feature-based methods are the most effective, as long as the images contain specific salient features (e.g., minutiae in fingerprint images [11]). On the other hand, with very low-resolution images where the number of detectable features is insufficient due to down-sampling, or images with few reliably localized features, a more robust alternative is to use an image-based registration approach that uses directly the pixel intensities, without searching for specific visual features.

The main requirements for the registration in the context of panoramic application are (a) robustness to illumination changes, (b) robustness to object motion in the scene, and (c) low computational complexity.

In order to achieve robustness with respect to illumination changes, we employ only similarity metrics that are invariant to illumination changes (i.e., normalized cross-correlation). Robustness to object motion in dynamic scenes is achieved by using RANSAC procedure to find and eliminate incorrect feature correspondences between images. In order to achieve an unlimited field of view the final panorama must be mapped onto a spherical manifold. For low computational complexity it is important that each image is warped only once. In particular, we don't want to keep warping the images during the registration process, like some other image-based methods do [2, 10]. Instead, we only warp the coordinates as we look for matching features in the neighboring image, and warp small patches around features to account for rotation effects.

Our registration approach follows a coarse-to-fine strategy, and it is hybrid in the sense that it relies on image matching for aligning coarse resolution levels, and on feature matching for refining the alignment parameters at finer resolution levels. With the exception of coarse levels we base our registration on feature matching. The main reason behind this choice is the need to use RANSAC to cope with outliers that are either represented by moving object, or caused by the parallax effects in case of camera translation. A secondary reason for using feature matching at high resolution levels is the computational complexity. By using features we do not need to re-warp the images during the registration process, or compare images on areas that have uniform intensity (sky, floor, etc.). Feature-based matching is not suitable at coarse resolution levels, for reasons mentioned before. However, given the small sizes of the coarse resolution levels, the penalty of using an image-based approach for these levels is negligible. Both image and feature matching operations are carried out using similarity metrics that are invariant to illumination changes. The registration algorithm comprises the following steps:

Algorithm 1

Input: input image A and template image B.

Output: registration parameters that align A to B.

1. Multi-resolution decomposition of A and B images.

2. Image-based registration at coarse resolution levels.

3. Feature-based registration at subsequent finer levels.

The multi-resolution representation of each image is achieved by hierarchical decomposition, where each resolution level is calculated by linear filtering and down-sampling of the previous level. We use a 3 tap linear filter $[\frac{1}{4}, \frac{1}{2}, \frac{1}{4}]$ that can be implemented with bit shift-operations instead of multiplications.

Following the typical multi-resolution registration strategy, we aim to estimate large displacements at the coarse resolution levels, and progressively refine these estimates at finer levels. Observing that image features such as corners are less reliable at coarse resolutions, we employ an image-

matching approach at the pyramid levels whose width is below 100 pixels. At each coarse level, image matching is carried on within a specific search range around the solution estimated at the previous level, using normalized cross-correlation.

Feature-based registration is used at finer resolution levels. We start with an initial guess provided by the result estimated at the previous level, and the feature matching approach works as follows:

Algorithm 2

Input: input image A and template image B at current resolution level; registration estimate from the previous level.

Output: new registration parameter estimates.

1. Identify the overlap between A and B based on the current registration parameters.

2. Detect Harris corners in A inside the overlap area. In order to ensure a uniform spatial distribution of the detected corners, the overlap area of A is first tessellated into blocks, and only one corner is retained inside each block.

3. For each corner in A, find the best matching position in B by performing a block matching procedure within a specific search range $[-S, S] \times [-S, S]$, where S is initialized to 1.

4. Apply RANSAC to remove inconsistent point matches.

5. Evaluate the reliability of the estimated transformation based on the number of inliers found by RANSAC.

6. If the number of inliers is below a threshold, double the search range S and go to step 3, unless S exceeds the maximum allowed value (32 in our work). If S exceeds the maximum value, we assume that the registration will not improve and the current registration parameters are provided as the output of the registration algorithm.

Assuming that different images are captured by rotating the camera around its optical center, the final panorama can be mapped on a spherical manifold, where the position of each image pixel is defined by two angular dimensions azimuth and elevation, i.e., (ϕ, θ).

The radius of the sphere is directly related to the scale at which the panorama image is going to be created, and since it is the same for all pixels, we don't need to specify it until the warping stage. Another parameter needed to map each image pixel to the spherical surface is the camera focal length. As mobile cameras are typically equipped with lenses of fixed focal length, it can be known before-hand.

The motion model between images on the sphere can be well described by a rigid model (i.e., translation and rotation). This simplifies registration and provides more robustness as there are fewer parameters (only three) to be estimated. In contrast, if we used a homographic motion model between pairs of planar images, we would have eight registration parameters for each image pair.

In the Step 3 of Algorithm 2, we perform a search for corresponding corners, as illustrated in Fig. 3. Using the previously estimated transformation in spherical domain, we can map one corner position from A onto the corresponding position in B by converting first its coordinates to spherical coordinates, next applying the transformation in spherical coordinates, and finally converting the transformed spherical coordinates back into the image B coordinates. The position found in this manner in B is then adjusted by local block matching. Noting that spherical manifold mapping changes the relative coordinates between features but has only a small deformation effect in the immediate neighborhood around each feature, we perform this local block matching operation directly in the image coordinates. The search range S, used for this local block matching, is adaptively increased in our algorithm by evaluating the validity of the estimated transformation. A small search range (e.g., 1, 2 pixels) is often sufficient. However, as shown in Step 6, a validity check is performed based on the number of inliers estimated by the RANSAC procedure. When this number is below a threshold, the search range is increased and the feature detection step is repeated.

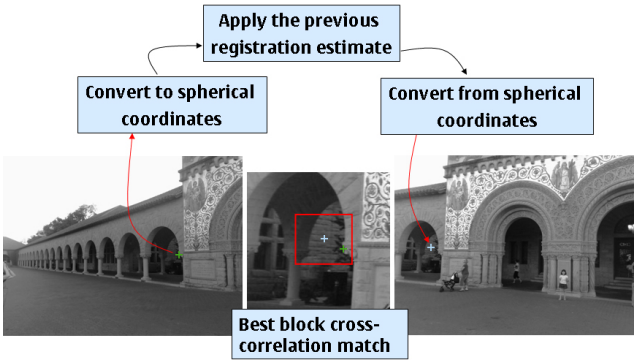

Figure 3. Matching corresponding image features.

2.4. Global registration adjustment

At the end of pair-wise registration, we have for each pair of overlapping images the corresponding corner positions in both images, as well as the registration transformation in spherical coordinates.

Based on this information we can evaluate the position of each image in the final panorama. We distinguish between two cases: **1D panorama** where image has at most two neighbors, and **2D panorama** where some images have more than two neighbors. In order to optimize the computational cost, we treat these cases differently.

2.4.1 1D panorama

In the case of 1D panorama the transformations estimated between images by the pair-wise registration algorithm are used to align all images. Taking one of the images as reference, we calculate the composite transformations of all

other images to align them with the reference image. Next, we need to adjust the global orientation of the panoramic image so that its longer dimension is aligned with the horizontal axis of the final image. In order to do this, we first estimate the global panorama orientation angle α with respect to the horizontal axis, and then adjust the rotation of each individual image by α.

2.4.2 2D panorama

With a 2D panorama, we use the corresponding points between overlapping images to estimate a suitable 3D rotation matrix for each image. Denoting the 3D rotation matrix of the camera at the moment it captures the k-th image by R_k, and the i-th pair of corresponding points between the k-th and ℓ-th by $(\mathbf{x}_{k,i}, \mathbf{x}_{\ell,i})$, we can formulate the following global objective function to be minimized by the rotation matrices:

$$\sum_{k,\ell,i} ||R_k^{-1}\mathbf{x}_{k,i} - R_\ell^{-1}\mathbf{x}_{\ell,i}||^2 \quad (1)$$

This objective function aims to align the 3D ray directions of the corresponding points. Only a small number of iterations (10 or fewer in our experiments) are needed to minimize this objective function, using the regular gradient descent method proposed in [9].

Like in 1D panorama case, an adjustment of the estimated rotation matrices is needed in order to align the final panorama with respect to the horizontal axis. We use the most central image of the panorama as reference, and all other images are aligned with respect to it.

In both 1D and 2D panorama cases, after estimating the geometrical transformations, we calculate the approximate bounding box of the warped image by mapping the four image corners using the estimated transformation. At this stage, the bounding box size and position with respect to the panorama origin are represented in angular spherical coordinates. Their final values in pixel coordinates are determined in the warping stage based on the specific scale of rendering.

2.5. Image Warping

Once the registration parameters are estimated, the images are warped using the spherical mapping. Since the registration is calculated directly in spherical coordinates, each image only needs to be warped once. Moreover, in order to reduce the computational time the warping operation is implemented in OpenGL ES 1.1 and executed by the graphical processing unit (GPU), which is available on many modern smartphones.

For each image we perform the following algorithm:

Algorithm 3

Input: input image; geometrical transformation; normalized bounding box; normalized position coordinates in the final panorama; scale.

Output: the warped image; bounding box; relative position in the final panorama.

1. Calculate the bounding box and the relative position in pixel coordinates by applying the scale factor.

2. Divide the input image into a triangular mesh.

3. Apply the estimated geometrical transformation at each vertex of the triangular mesh.

4. Provide the original image as a texture map to the GPU, along with the transformed vertex coordinates of the grid.

5. Perform image warping on the GPU.

6. Read the warped image back to CPU.

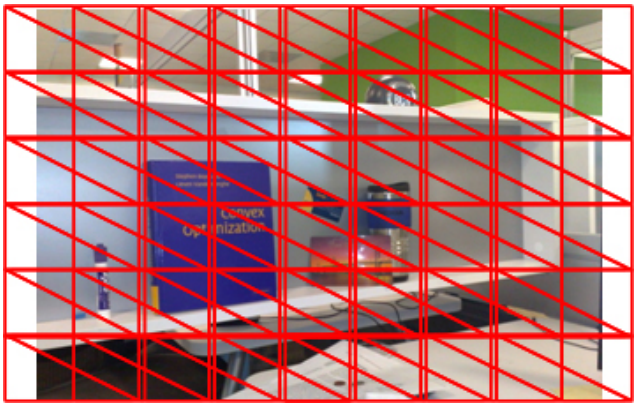

Figure 4. Image division into a triangular mesh for GPU warping.

3. Optimal seam finding and image labeling

After spatial registration and warping of the source images, we need to stitch them together to create a panoramic image. A simple copying and pasting of overlapping source images is likely to produce visible artificial edges in the seams between images due to differences in the auto-exposure and auto-white-balance settings that vary at each image, or create ghosting artifacts due to object motion in the scene during the image capture. Optimal seam finding can find the best way to stitch the source images together for creating a composite image.

We want to find optimal seams in the overlapping areas of source images, create labeling that maps only one input image pixel to each output pixel using the optimal seam information, and construct a composite image by copying data from the source images indicated by the labels. In this way, the ghosting and blurring problems caused by object motion and small spatial alignment errors in the overlapping areas can be avoided, as each pixel in the composite image comes only from one source image.

In our panoramic imaging system, we have experimented on two approaches to image labeling: graph cut optimization [13] and dynamic programming [12]. As we in practice

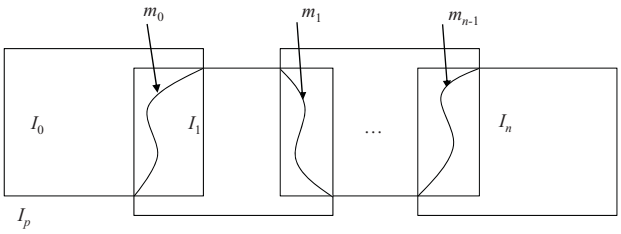

Figure 5. Optimal seam finding for a general case.

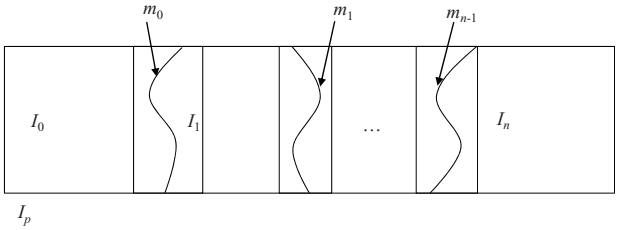

Figure 6. Optimal seam finding for cropped source images.

find good seams using dynamic programming and it performs much faster than graph cut, we only describe that approach.

3.1. Image labeling with dynamic programming

As Fig. 5 shows, the source images $I_0, I_1, \ldots, I_n$ are already registered into a composite image I_p. We need to find optimal seams $m_0, m_1, \ldots, m_{n-1}$, create labeling between all pixels of the composite image and the source images, and build the composite image I_p.

For the general case of two images overlapping shown in Fig. 5, 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. However, if the images have been cropped so that their top and bottom rows align (see Fig. 6), the optimal seam 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.

Suppose I_{k-1} and I_k are two overlapping images. We compute a squared difference s in the overlap between I_{k-1} and I_k as an error surface

$$s = (I_{k-1} - I_k)^2. \tag{2}$$

In the simple case of Fig. 6, we find a minimal cost path through the surface by scanning it row by row and compute a cumulative minimum squared difference S for all paths:

$$\begin{aligned} S(w,h) = s(w,h) + min(S(w-1,h-1), \\ S(w,h-1), S(w+1,h-1)), \end{aligned} \tag{3}$$

where w and h are the indices of the column and row of the error surface. The minimal cost path can be obtained by tracing back the paths from bottom to top by dynamic programming. In the more complex case of Fig. 5 we accumulate the errors in a diagonal fashion rather than row by row. We use the minimal cost path as an optimal seam to create labeling and merge the two images together.

Note that we find the seams after the images have been warped (see Fig. 7), and we have to make sure that the seams may only pass via valid pixels.

Figure 7. Warped images and invalid areas.

Figure 8. Composite image created by image labeling.

4. Image blending

After image labeling, the source images are merged into a composite image. If the source images are similar enough in colors and luminance, the seams and stitching artifacts may be invisible, and the composite image can be the final panorama. Figure 8 on the left shows an example. However, when the source images differ in colors and luminance, the stitching artifacts can still be seen in the composite image on the right. Further processing is needed to remove these artifacts.

Transition smoothing processes can be used to reduce color differences between source images to hide the seams and other stitching artifacts. Here we use a fast image cloning approach for transition smoothing [5].

4.1. Fast image cloning for image blending

Poisson blending [8] is an intensive image blending approach. It performs image blending in the gradient domain and can provide high-quality blended images. Farbman et al. [5] showed that one can obtain as good results using a much simpler approach. Fig. 9 shows how we use their method to blend the current image I_c to the current panoramic image I_p. We keep the color values on the 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 I_c so that no visible boundary remains. We do this by first

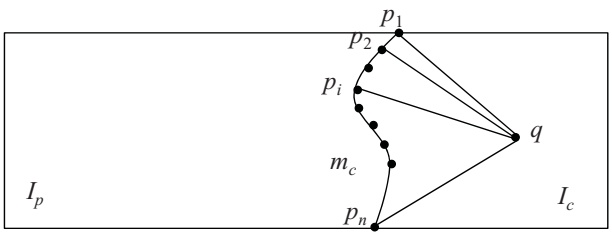

Figure 9. Image blending by interpolating the image differences at boundary.

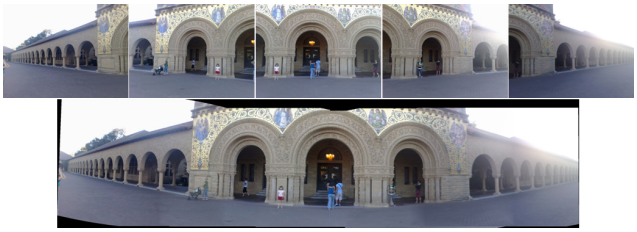

Figure 10. Image registration copes with large illumination changes between images.

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 I_c. We then interpolate the color differences at pixel q by

$$P(q) = \sum_{i=1}^{n} w_i(q) P(p_i), \tag{4}$$

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

$$w_i(q) = \frac{1/||p_i - q||}{\sum_{j=1}^{n} 1/||p_j - q||}. \tag{5}$$

5. Applications and Result Analysis

5.1. Image registration and warping

The ability of the registration approach to cope with large illumination changes has been tested in a large number of panoramic image creations. One example is shown in Fig. 10.

The ability of the proposed method to eliminate outliers is illustrated by the example shown in Fig. 11 where the expected result of the registration algorithm is to estimate the camera motion with respect to the background, without getting distracted by the car that is passing in front of the camera.

Figure 11. Image registration copes with moving objects in the scene: four image frames before (left) and after registration (right).

Figure 12. Panoramic image created by the mobile panorama imaging system for an indoor scene with five source images.

Figure 13. Panoramic image created by the mobile panorama imaging system for an outdoor scene with five source images.

5.2. Indoor and outdoor scenes

We apply the mobile panoramic imaging system to an image sequence captured indoors (see Fig. 12). After the image sequence is captured, the source images are registered and warped. From the result we can see that the registration result is good although people are moving while the image sequence was captured. Next, the warped images are labeled. The optimal seams, found by graph cut or dynamic programming, are used to merge the source images and avoid ghosting problems caused by object motion. Finally, the composite image obtained from the labeling processing is processed by Poisson blending. From the final result we can see that the color transition in the whole panoramic image is good.

Figure 13 shows an outdoor scene with five source images with different colors and luminance. Again, we can see that the system can handle color changes and object motion

Figure 14. Effects of different blending approaches.

and produce a high-quality outdoor panoramic image.

5.3. Effects of blending approaches

Figure 14 shows the effects of different blending approaches. The top row shows an image sequence with four outdoor images with moving objects. After registration and warping, we perform different processing methods to see their effects in the area denoted with the red rectangle shown in the mid row.

We perform image blending using normal alpha blending. From the result in the bottom row left we can see ghosting artifacts caused by object motion. We apply image labeling to find optimal seams in the overlapping areas of the source image and merge them to create a composite image. From the result in the center we can see that the ghosting artifacts are avoided and there is no blurring in the overlapping areas. Since the source images differ in colors, the seams between the images are still visible. We perform transition smoothing for the result obtained by image labeling, and the result in the bottom right shows that visible seams and stitching artifacts are removed.

5.4. More panoramic images

Figure 15 shows more results created from different types of scenes. The first row shows a panoramic image created for an outdoor scene. The color transition is good in the whole panoramic image, although the source images differ in colors. The second row shows one more indoor result. Again, from the result we can see that the system can completely handle object motion during the capture of the image sequence. The third row is a panoramic image created by the system with inputs of a high-dynamic-range scene shown in the last row. Since the system uses intensive

Figure 15. More example panoramic images creating by the mobile panorama imaging system.

blending, it can also handle cases where the dynamic range of the input image sequence is large.

5.5. Application for 360 degree panoramic images

Figure 16 shows a 360 degree panoramic image with 19 source images captured for an outdoor scene. Although the source images are different in colors and luminance, especially on the right hand side of the building, the system can smooth the color transitions for the whole panoramic image. We also can notice that the system can handle the moving cars on the right by providing correct registration and optimal seams to avoid ghosting and blurring.

5.6. Performance measurements on mobile devices

We evaluated the computation time of our system on a Nokia N95 8GB mobile phone with an ARM 11 332 MHz processor and 128 MB RAM. For 4 input images that together cover a field of view (FOV) of 150^o, registration, warping with GPU, labeling, and blending take 17.6, 6.0, 3.5, and 12.5 seconds, respectively. When we add the time required to read in images from the camera, and save the result to mass memory, the total time is about 42.9 seconds. For 5 images with total FOV of 180^o, registration, warping, labeling, and blending take 22.9, 7.4, 4.5, and 15.6 seconds. The whole process takes 53.9 seconds. Image warping with CPU is about 10 times slower than with GPU (MBX from Imagination Technologies, OpenGL ES 1.1).

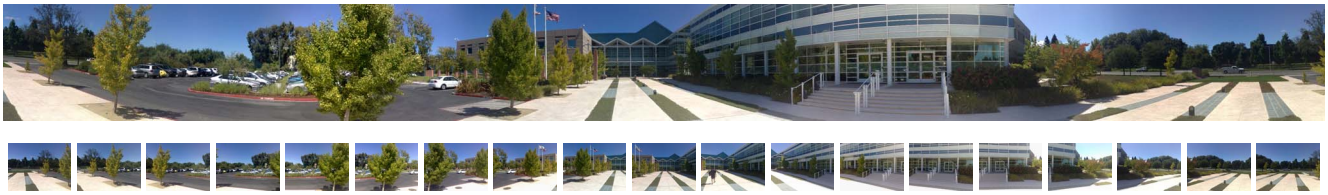

Figure 16. 360 degree panoramic image created by the panorama imaging system for an outside scene.

6. Discussion

We have implemented this mobile panorama system on several Symbian-based smartphones as well as a Linux-based smartphone. Being able to use an open-source cross-platform, GUI library Qt, made the porting task easier.

The user should rotate the camera around its center to capture correct rays for a panorama. If there is any translation involved, instead of a pure rotation, a parallax effect is introduced, where the images of nearby objects translate different distances on the image plane from the images of far-away objects.

The parallax effect makes perfect registration impossible, as there is apparent motion between objects at different distances. Our robust registration and labeling approach helps to minimize artifacts, whether they are due to object motion or parallax due to incorrect capture mechanics.

7. Conclusions

In this paper we have proposed a complete system for automatic panoramic imaging. Our system allows the user to rotate the camera freely instead of a fixed (e.g., horizontal) pattern, and generates an artifact-free panorama with unlimited viewing angles. The proposed system is efficient enough to implement on mobile phones, yet generates results comparable to complex PC software.

Acknowledgements

Many people have worked with us on the panorama system at various stages and roles. We'd like to thank Wei-Chao Chen, Natasha Gelfand, Amol Khadilkar, Chia-Kai Liang, Dingding Liu, Chris Paretti, Gururaj Putraya, John Schettino, Daniel Vaquero, and Xianglin (Shawn) Wang.

References

[1] A. Adams, N. Gelfand, and K. Pulli. Viewfinder alignment. *Computer Graphics Forum*, 27(2):597–606, 2008.

[2] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 2004.

[3] P. Baudisch, D. Tan, D. Steedly, E. Rudolph, E. Uyttendaele, C. Pal, and R. Szeliski. An exploration of user interface designs for real-time panoramic photography. *Australian Journal of Information Systems*, 13(2):151–166, 2006.

[4] J. Boutellier, O. Silvén, M. Tico, and M. Vehviläinen. Creating panoramas on mobile phones. In *Electronic Imaging Conference*, San Jose, California, USA, Jan. 2007.

[5] 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.

[6] S. Ha, H. Koo, S. Lee, N. Cho, and S. Kim. Panorama mosaic optimization for mobile camera systems. *IEEE Transactions on Consumer Electronics*, 53(4), 2007.

[7] S. Ha, H. Koo, S. Lee, N. Cho, and S. Kim. Photographing for on-line mobile panorama. In *International Conference on Consumer Electronics*, Jan. 2008.

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

[9] H.-Y. Shum and R. Szeliski. Construction of panoramic mosaics with global and local alignment. *International Journal of Computer Vision*, 36(2):101–130, 2000.

[10] P. Thevenaz and M. Unser. A Pyramid Approach to Subpixel Registration Based on Intensity. *IEEE Trans. on Image Processing*, 7(1):27–41, 1998.

[11] M. Tico and P. Kuosmanen. Fingerprint matching using an orientation-based minutia descriptor. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(8):1009–1014, 2003.

[12] Y. Xiong and K. Pulli. Fast image labelling for producing high resolution panoramic images and its applications on mobile devices. In *ISM: Proc. IEEE International Symposium on Multimedia*, 2009.

[13] Y. Xiong and K. Pulli. Gradient domain image blending and implementation on mobile devices. In *MobiCase: Proc. Int. Conf. on Mobile Computing, Applications, and Services*, 2009.

[14] B. Zitova and J. Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21:977–1000, 2003.