Photographing Long Scenes with Multi-Viewpoint Panoramas

Aseem Agarwala[1] Maneesh Agrawala[2] Michael Cohen[3] David Salesin[1,4] Richard Szeliski[3]

[1]University of Washington [2]University of California, Berkeley [3]Microsoft Research [4]Adobe Systems

Figure 1 A multi-viewpoint panorama of a street in Antwerp composed from 107 photographs taken about one meter apart with a hand-held camera.

Abstract

We present a system for producing multi-viewpoint panoramas of long, roughly planar scenes, such as the facades of buildings along a city street, from a relatively sparse set of photographs captured with a handheld still camera that is moved along the scene. Our work is a significant departure from previous methods for creating multi-viewpoint panoramas, which composite thin vertical strips from a video sequence captured by a translating video camera, in that the resulting panoramas are composed of relatively large regions of ordinary perspective. In our system, the only user input required beyond capturing the photographs themselves is to identify the dominant plane of the photographed scene; our system then computes a panorama automatically using Markov Random Field optimization. Users may exert additional control over the appearance of the result by drawing rough strokes that indicate various high-level goals. We demonstrate the results of our system on several scenes, including urban streets, a river bank, and a grocery store aisle.

1 Introduction

Imagine trying to take a photograph of all the buildings on one side of a city street extending for several blocks. A single photograph from the street's opposite side would capture only a short portion of the scene. A photograph with a wider field of view would capture a slightly longer section of the street, but the buildings would appear more and more distorted towards the edges of the image. Another possibility would be to take a photograph from a faraway viewpoint. Unfortunately, it is usually not possible to get far enough away from the side of a street in a dense city to capture such a photograph. Even if one could get far enough way, the result would lose the perspective depth cues that we see when walking down a street, such as awnings that get bigger as they extend from buildings, and crossing streets that converge as they extend away from the viewer. The problems described here are not unique to streets. In general, single-perspective photographs are not very effective at conveying long scenes, such as the bank of a river or the aisle of a grocery store. In this paper we introduce a practical approach to producing panoramas that visualize these types of long scenes.

`http://grail.cs.washington.edu/projects/multipano/`

We are not the first to address this problem. Perhaps the best-known approach, explored by both artists and computer scientists, is to create a *slit-scan panorama* (also called a *strip panorama*) [Levin 2005]. Historically, these panoramas were created by sliding a slit-shaped aperture across film; the digital approach is to extract thin, vertical strips of pixels from the frames of a video sequence captured by a translating video camera. The resultant image can be considered *multi-viewpoint* (or *multi-perspective*), because the different strips of the image are captured from different viewpoints. Multi-viewpoint panoramas can be quite striking; moreover, they may be of practical use for a variety of applications. Images of the sides of streets, for example, can be used to visually convey directions through a city, or to visualize how proposed architecture would appear within the context of an existing street.

Strip panoramas have multiple disadvantages, however, as we discuss in Section 1.1. In this paper, we present a different approach to producing multi-viewpoint panoramas of long scenes. The input to our system is a series of photographs taken with a hand-held camera from multiple viewpoints along the scene. To depict the side of a street, for example, we walk along the other side and take hand-held photographs at intervals of one large step (roughly one meter). The output is a single panorama that visualizes the entire extent of the scene captured in the input photographs and resembles what a human would see when walking along the street (Figure 1). Rather than building the panorama from strips of the sources images, our system uses Markov Random Field optimization to construct a composite from arbitrarily shaped regions of the source images according to various properties we wish the panorama to exhibit. While this automatically composited panorama is often satisfactory, we also allow for interactive refinement of the result. The user can paint rough strokes that indicate certain goals, such as the use of a certain viewpoint in a certain area of the panorama. The major contribution of our work is a practical approach to creating high-quality, high-resolution, multi-viewpoint panoramas with a simple and casual capture method. To accomplish this goal, we present a number of novel techniques, including an objective function that describes desirable properties of a multi-viewpoint panorama, and a novel technique for propagating user-drawn strokes that annotate 3D objects in the scene (Section 2.4).

1.1 Related work

The term "panorama" typically refers to single-viewpoint panoramas, which can be created by rotating a camera around its optical center [Szeliski and Shum 1997]. Strip panoramas, however, are created from a translating camera, and there are many variants, e.g., "pushbroom panoramas" [Gupta and Hartley 1997; Seitz and Kim 2003], "adaptive manifolds" [Peleg et al. 2000], and "x-slit" images [Zomet et al. 2003]. Zheng [2003] and Roman et al. [2004]

both describe techniques designed specifically for creating strip panoramas of long streets.

In their simplest form, strip panoramas exhibit orthographic projection along the horizontal axis, and perspective projection along the vertical. This disparity in projection leads to distortions for scenes that are not strictly planar. Objects at a certain depth from the camera plane are shown with a correct aspect ratio, but objects further from the camera appear horizontally stretched while closer objects appear squashed. The depth at which objects are shown correctly can be adjusted by varying the width of the strips taken from the video frames. Automatic approaches to varying strip width either estimate scene depth [Rav-Acha et al. 2004] or attempt to minimize the appearance of vertical seams [Wexler and Simakov 2005]. Roman et al. [2004] take an interactive approach. In their system, the user can choose to include several separated strips of single-viewpoint perspective in the panorama; the system will then interpolate viewpoints between these regions. They demonstrate better results than traditional strip panoramas, but require a complex capture setup. Since they need a dense sampling of all rays that intersect the camera path (i.e., a 3D light field), they use a high-speed 300-frame-per-second video camera mounted on a truck that drives slowly down the street.

All of these variants of strip panoramas still exhibit distortions for scene with varying depths, especially if these depth variations occur across the vertical axis of the image. There are other problems with strip panoramas as well. The use of orthographic projection across the horizontal axis of the image sacrifices local perspective effects that serve as useful depth cues; in the case of a street, for example, crossing streets will not converge to a vanishing point as they extend away from the viewer. Another problem is that strip panoramas are created from video sequences, and still images created from video rarely have the same quality as those captured by a still camera. The resolution is much lower, compression artifacts are usually noticeable, and noise is more prevalent since a constantly moving video camera must use a short exposure to avoid motion blur. Finally, capturing a suitable video can be cumbersome; strip panoramas are not typically created with a hand-held video camera, for example.

Strip panoramas are not the only way to image in multiple perspectives. Multi-perspective imaging can take many forms, from the more extreme non-photorealism of Cubism to the subtle departures from linear perspective often used by Renaissance artists [Kubovy 1986] to achieve various properties and arrangements in pictorial space. Several researchers have explored multi-perspective renderings of 3D models [Agrawala et al. 2000; Yu and McMillan 2004a]. Yu and McMillan presented a model that can describe any multi-perspective camera [2004b]. Multi-perspective images have also been used as a data structure to facilitate the generation of traditional perspective views [Wood et al. 1997; Rademacher and Bishop 1998; Zomet et al. 2003]. In the field of photogrammetry [Kasser and Egels 2002], aerial or satellite imagery from varying viewpoints are stitched together to create near-orthographic, top-down views of the earth (e.g., Google Earth [2005]).

1.2 Approach

Our approach to generating effective multi-viewpoint images is inspired by the work of artist Michael Koller [2004], who creates multi-viewpoint panoramas of San Francisco streets that consist of large regions of ordinary perspective photographs artfully seamed together to hide the transitions. There is no obvious standard or ground truth by which to evaluate whether a specific multi-viewpoint panorama visualizes a scene well, even if the 3D scene geometry and appearance were known. Koller's images, however, are attractive and informative, so we attempt to define some of their properties:

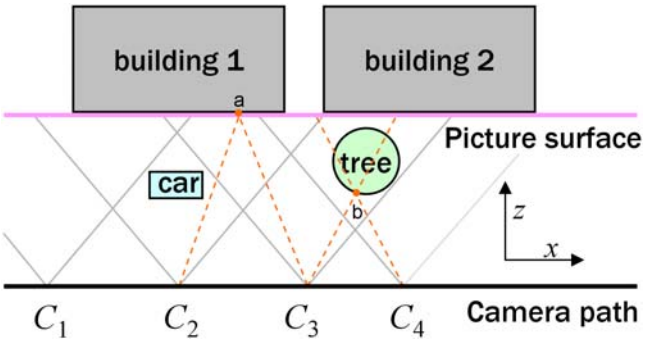

Figure 2 A plan view (xz slice) of a hypothetical scene captured with four photographs taken from viewpoints $C_1, \ldots, C_4$. The picture surface is placed at the front of the two buildings so that their projections are consistent inbetween the different views. However, the projections of objects off of the picture surface will not be consistent. For example, point a on the front of building 1 will project from each camera to the same pixel on the picture surface, while point b on the tree will project to different places.

1. Each object in the scene is rendered from a viewpoint roughly in front of it to avoid perspective distortion.

2. The panoramas are composed of large regions of linear perspective seen from a viewpoint where a person would naturally stand (for example, a city block is viewed from across the street, rather than from some faraway viewpoint).

3. Local perspective effects are evident; objects closer to the image plane are larger than objects further away, and multiple vanishing points can be seen.

4. The seams between these perspective regions do not draw attention; that is, the image appears natural and continuous.

We thus designed our system to generate multi-viewpoint panoramas that exhibit these properties as well as possible. Note, however, that maximizing these properties may not produce an effective multi-viewpoint panorama for any arbitrary scene. Koller's images are particularly good at visualizing certain types of scenes: those that are too long to effectively image from a single viewpoint, and those whose geometry predominantly lies along a large, *dominant plane* (for example, the fronts of the buildings in Figure 1). Because of this latter property, these scenes can be effectively visualized by a single two-dimensional image whose image plane is parallel to the dominant plane of the scene. More three-dimensional phenomena are unlikely to be well-summarized by a single image. We have not, for example, attempted to create multi-viewpoint panoramas that turn around street corners, or that show all four sides of a building, since they are likely to be less comprehensible to the average viewer. Note also that the dominant plane need not be strictly planar. For example, the river bank we depict in Figure 13 curves significantly (as is evident from the plan view in Figure 5).

Our system requires the user to specify a *picture surface* that lies along the dominant plane. Our system's success depends on a key observation (see Figure 2): images projected onto the picture surface from their original 3D viewpoints will agree in areas depicting scene geometry lying on the dominant plane (assuming Lambertian reflectance and the absence of occlusions). This agreement can be visualized by averaging the projections of all of the cameras onto the picture surface (Figure 7). The resulting image is sharp for geometry near the dominant plane because these projections are consistent and blurry for objects at other depths.[1] Transitions between

[1]This image bears some resemblance to synthetic-aperture photogra-

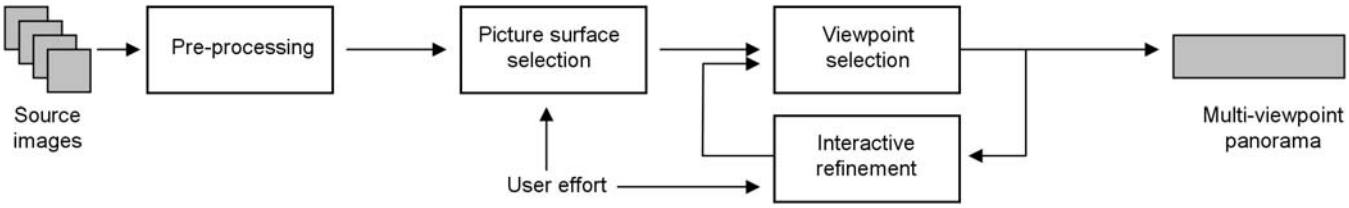

Figure 3 An overview of our system for creating a multi-viewpoint panorama from a sequence of still photographs. Our system has three main phases. In the preprocessing stage, our system takes the source images and removes radial distortion, recovers the camera projection matrices, and compensates for exposure variation. In the next stage, the user defines the picture surface on which the panorama is formed; the source photographs are then projected onto this surface. Finally, our system selects a viewpoint for each pixel in the output panorama using an optimization approach. The user can optionally choose to interactively refine this result by drawing strokes that express various types of constraints, which are used during additional iterations of the optimization.

different viewpoints can thus be placed in these aligned areas without causing objectionable artifacts. Our system uses Markov Random Field optimization to choose one viewpoint for each pixel in the output panorama; the optimization tries to place seams between viewpoints in aligned areas, while also maximizing the four desirable properties of a multi-viewpoint panorama listed above.

The operating range of our approach is thus limited to scenes whose geometry intersects the dominant plane often enough for natural transitions between viewpoints to occur. The required frequency of these intersections varies inversely with the field of view of the input photographs, since our approach is unlikely to work well if a portion of the scene larger than the field of view of one camera is entirely off of the dominant plane. For this reason we often use a fisheye lens to insure a wide field of view. Our approach is not, however, limited to strictly planar scenes (which are trivial to stitch). In fact, regions off of the dominant plane provide valuable local perspective cues that improve the overall composition. Our goal is to leave these areas intact and in their correct aspect ratios (unlike strip panoramas, which squash or stretch regions off of the dominant plane). This goal can be accomplished by restricting transitions between viewpoints to areas intersecting the dominant plane. Objects off of the dominant plane will thus be either depicted entirely from one viewpoint, or omitted altogether (by using viewpoints that see around the object, which generally works only for small objects).

It is not always possible to limit transitions to areas intersecting the picture surface. For example, the bottom of Figure 1 contains cars, sidewalk, and road, none of which lie near the picture surface located at the front of the buildings. In this case, transitions between viewpoints must "cheat" and splice together image regions that do not actually represent the same geometry. The transitions between these image regions should be placed where they are least noticeable; for example, splicing together separate regions of sidewalk in Figure 1 will likely not be objectionable. Such decisions, however, are subjective and not always successful. We thus allow the user to easily refine the results by drawing rough strokes indicating various constraints, as described in Section 2.4. The result in Figure 1, however, required no interactive refinement.

2 System details

An overview of our system is shown in Figure 3. We now describe each step of our system in detail.

We begin by capturing a sequence of photographs that depict the scene for which we wish to produce a panorama. Our image capture process is fairly simple; we walk along the scene and take handheld photographs roughly every meter. We use a digital SLR camera with auto-focus and manually control the exposure to avoid large exposure shifts. For some data sets we used a fisheye lens to ensure capture of a wide field of view.

phy [Levoy et al. 2004], though with much sparser sampling.

2.1 Preprocessing

After capturing the photographs, we use the freely available software PtLens [2005] to remove radial distortion and determine the field of view of the fisheye lens.

We then recover the projection matrices of each camera so that we can later project the source images onto a picture surface. If there are n cameras, we recover a 3D rotation matrix R_i, a 3D translation t_i, and a focal length f_i for each camera, where $1 \leq i \leq n$. Given these parameters, the location of the i'th camera in the world coordinate system can be defined as $C_i = -R_i^T t_i$. We recover these parameters using a structure-from-motion system [Hartley and Zisserman 2004] built by Snavely et al. [2006]. This system matches SIFT features [Lowe 2004] between pairs of input images, and uses these point matches as constraints for an optimization procedure that recovers the projection parameters as well as a sparse cloud of 3D scene points. Brown and Lowe [2005] also combine SIFT and structure-from-motion in a similar fashion. The structure-from-motion results sometimes contain small errors, such as slow drift that can cause an otherwise straight scene to slowly curve; we describe a solution to this problem in Section 2.2.

The next pre-processing step is to compensate for exposure variation between the source photographs, which is a common problem for panoramic stitching [Uyttendaele et al. 2001]. This problem is exacerbated in our case since the data capture occurs over a longer period of time, and outdoor lighting conditions can change significantly. We therefore need to adjust the exposure of the various photographs so that they match better in overlapping regions. One approach that is well studied is to recover the radiometric response function of each photograph (e.g., [Mitsunaga and Nayar 1999]). For our application we do not need highly accurate exposure adjustment, so we take a much simpler approach. We associate a brightness scale factor k_i to each image, and for two photographs I_i, I_j we assert that $k_i I_i = k_j I_j$ for pixels that depict the same geometry. Each SIFT point match between two images gives us three linear constraints of this form (one for each color channel). We solve for the values of k_i that best meet these constraints in a least-squares sense by solving a linear system. (The linear system is defined only up to a global scale, so we include a weak prior that each $k_i = 1$.)

2.2 Picture surface selection

The next step of our approach is for the user to define a picture surface upon which the panorama will be formed; this picture surface should be roughly aligned with the dominant plane of the scene. The picture surface is defined by the user as a curve in the xz plane (i.e., the plan view); the curve is then extruded in the y direction to

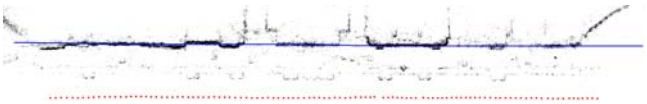

Figure 4 A plan view (xz slice) of the scene shown in Figure 1. The extracted camera locations are shown in red, and the recovered 3D scene points in black. The blue polyline of the picture surface is drawn by the user to follow the building facades. The y-axis of the scene extends out of the page; the polyline is swept up and down the y-axis to form the picture surface.

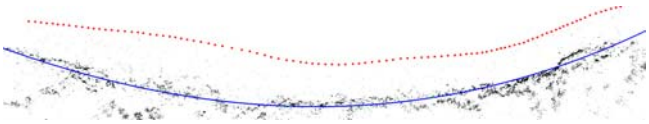

Figure 5 A plan view (xz slice) of the river bank multi-viewpoint panorama in Figure 13; the extracted camera locations are shown in red, and the 3D scene points in black. The dominant plane of the scene is non-obvious, so the blue picture surface is fit to a subset of the scene points selected by the user (as described in Section 2.2).

the extent necessary to contain the four corners of each projected source photograph.

Our system helps a user to define the picture surface in two steps. The first step is to define the coordinate system of the recovered 3D scene. This step is necessary because the recovered 3D scene is in some unknown coordinate system. If the picture surface is to be extruded in the y dimension, the scene and camera locations should first be transformed so that the xz axes span the scene ground plane, and y points to the sky. The second step is to actually draw the curve in the xz plane that defines the picture surface.

Our system offers two approaches for choosing the coordinate system: one automatic, and one interactive. If we assume that the photographer's viewpoints were at a constant height and varied across both dimensions of the ground plane, we can fit a plane to the camera viewpoints using principal component analysis (PCA). The dimension of greatest variation (the first principal component) is the new x-axis, and the dimension of least variation the new y-axis. This approach was used for the scenes in Figures 11 and 13. Alternatively, the user can interactively define the coordinate system. First, the system uses the recovered camera projection matrices to project the 3D scene points into the original source photographs; then, the user selects a few of these projected points that lie along the desired axes. The first two selected points form the new y-axis. These two points can be any that lie along the desired up vector, e.g., two points along the vertical edge of a building. The user then selects two points to form the new x-axis. The two selected vectors are unlikely to be orthogonal, however, so the system takes the cross product of the two selected vectors to form the z-axis, and the cross product of z and y to form the x-axis.

After using one of these two approaches to find the world-coordinate frame, we can generate a plan view (an xz slice of the scene) that visualizes the locations of the camera and the 3D cloud of scene points (see Figure 4). We then ask the user to draw the picture surface in this plan view as a polyline (though other primitives such as splines would be straightforward to allow as well). Once the polyline is drawn, the system simply sweeps it up and down the y-axis to form a picture surface.

For street scenes it is easy for the user to identify the dominant plane in the plan view and draw a polyline that follows it. For other scenes, it is not clear from the plan view (such as the river bank in Figure 5) where the dominant plane is located. We thus allow for a second interactive approach to designing the picture surface. The

Figure 6 One of the source images used to create the panorama in Figure 1. This source image is then projected onto the picture surface shown in Figure 4, after a circular crop to remove poorly sampled areas at the edges of the fisheye image.

Figure 7 First row: the average image of all the projected sources for the scene shown in Figure 1. Notice that the street curves up towards the right. Second row: the average image after unwarping to straighten the ground plane and cropping.

system projects the 3D scene points into the original photographs, and then asks the user to select clusters of scene points that should lie along the picture surface. The user might typically select such clusters in roughly every tenth source photograph. Then, the system fits a third-degree polynomial $z(x)$ to the z-coordinates of these 3D scene points as a function of their x-coordinates (after filtering the points to remove any outliers). This function, swept up and down the y-axis, defines the picture surface. This approach was used to define the picture surface in Figure 5.

Once the picture surface location is defined in the plan view, the system samples the picture surface to form a regular 2D grid. We refer to $S(i, j)$ as the 3D location of the (i, j) sample on the picture surface. Each sample on this surface will form one pixel of the output panorama. The system projects each source photograph onto the picture surface by projecting each sample $S(i, j)$ of the surface into the source photographs using their recovered projection matrices. One example of a projected source image can be seen in Figure 6.

Once the source images are projected, the system produces a simple average image as shown in Figure 7. The user can then perform two additional steps: warping and cropping. Warping is sometimes required because of small drifts that can accumulate during structure-from-motion estimation and lead to ground planes that slowly curve, as shown in Figure 7. In this case, the user clicks a few points along the average image to indicate y values of the image that should be warped straight. The system then resamples the image to straighten the clicked points. Finally, the user can decide how to crop the picture surface by examining the average image. Figure 7 shows an example unwarped, cropped, average image.

2.3 Viewpoint selection

We can now create a panorama by choosing from among the possible viewpoints for each pixel of the panorama. The above steps result in a series of n images I_i of equivalent dimension, one of which is shown at the bottom of Figure 6. Image I_i represents the i'th viewpoint; that is, the projection of the i'th camera. We can thus create

Figure 8 First row: six of the 107 source photographs used to create the panorama in Figure 1. Second row: the seams between the different regions of linear perspective highlighted in red. Notice that these seams are very different from the vertical cuts required in strip panoramas.

a panorama by choosing a color for each pixel $p = (p_x, p_y)$ from one of the source images $I_i(p)$. This choice should be guided by the properties described in Section 1.2 that we wish the panorama to exhibit. We thus formulate an objective function that approximately measures these properties, and then minimize it using Markov Random Field (MRF) optimization. The optimization computes a labeling $L(p)$, where $L(p) = i$ if pixel p of the panorama is assigned color $I_i(p)$.[2] Our objective function has three terms, which we now describe.

The first term reflects the property that an object in the scene should be imaged from a viewpoint roughly in front of it. The notion of "in front" depends on the orientation of the scene geometry, so we use the picture surface as a proxy for this geometry. Consider a line that extends from a sample of the picture surface $S(p)$ in the direction of the normal to the picture surface at $S(p)$. Any camera viewpoint along this normal will be the most "in front," or, put another way, will view the scene in the most straight-on manner. We can thus evaluate this heuristic for a pixel p that chooses its color from I_i by measuring the angle between the vector $S(p) - C_i$ and the normal of the picture surface at $S(p)$. Note that the optical axis of the camera (i.e., the direction the camera is facing) is not used. Our system approximates this heuristic using a simpler and more computationally efficient method that is accurate if the cameras are roughly the same distance from the picture surface. We find the pixel p_i whose corresponding 3D sample $S(p_i)$ on the picture surface is closest to camera location C_i; in the case of a planar picture surface, this sample is exactly the sample for which camera C_i gives the most straight-on view. Then, if pixel p chooses its color from I_i, we approximate the heuristic as the 2D distance from p to p_i. Specifically, we define the cost function

$$D(p, L(p)) = |p - p_{L(p)}|.$$

The second term of the objective function encourages transitions between different regions of linear perspective to be natural and seamless. Previous work [Kwatra et al. 2003; Agarwala et al. 2004] shows that a seamless transition will minimize the cost function

$$V(p, L(p), q, L(q)) = |I_{L(p)}(p) - I_{L(q)}(p)|^2 + |I_{L(p)}(q) - I_{L(q)}(q)|^2 \quad (1)$$

between each pair of neighboring pixels p and q.

The third term of the objective function encourages the panorama to resemble the average image in areas where the scene geometry intersects the picture surface. To some extent this resemblance will occur naturally since there will typically be little variance between the different viewpoints in these areas of the panorama. However, motion in the scene, specular highlights, occlusions, and other such phenomena can detract from our goal of accurately visualizing scene geometry near the dominant plane. We thus calculate the mean and standard deviation of each pixel in the panorama among the various I_i in a robust fashion to discount outliers. We use a vector median filter [Astola et al. 1990] computed across the three color channels as a robust mean, and the median absolute deviation (MAD) [Huber 1981] as a robust standard deviation. The MAD is calculated as the median L_2 distance from the median color. We refer to the median image as $M(x,y)$ and the MAD as $\sigma(x,y)$. Assuming that image color channels vary from 0 to 255, we define the cost function

$$H(p, L(p)) = \begin{cases} |M(p) - I_{L(p)}(p)| & \text{if } \sigma(p) < 10 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

to minimize the difference between the median image and the image defined by the current labeling for those pixels whose robust standard deviation is low. Note that this approach will fail to identify cases where scene geometry at the picture surface is frequently occluded. A more complicated alternative, which we hope to explore, would be to compute *view-dependent* information about which pixels in which view lie along the dominant plane using multi-view stereo techniques [Kang et al. 2001].

Finally, we mention a constraint: any one camera has a limited field of view and will not project to every pixel of the panorama (e.g., the black pixels in Figure 6). We encode pixels in image I_i to which the i'th camera does not project as *null*, and do not allow $L(p) = i$ if $I_i(p) = null$. We thus wish to compute a panorama that minimizes the overall cost function

$$\sum_p (\alpha D(p, L(p)) + \beta H(p, L(p))) + \sum_{p,q} V(p, L(p), q, L(q)),$$

which sums the three terms over each pixel p and each pair of neighboring pixels p and q. This cost function has the familiar form of a Markov Random Field and can be minimized in several ways; we do so using min-cut optimization [Kolmogorov and Zabih 2002] in a series of alpha-expansion moves [Boykov et al. 2001]. We determine the weights experimentally and use the same values $\alpha = 100$ and $\beta = .25$ for all the results used in this paper. However, these weights have natural meanings that could be exposed to the user. Higher values of α encourage pixels from more straight-on views at the expense of more noticeable seams. Lower values of both α and β are more likely to remove objects off of the dominant plane (such as power lines or cars, in the case of a street).

Two additional steps are required to finish the panorama. We first compute the panorama at a lower resolution so that the MRF optimization can be computed in reasonable time (typically around 20

[2] Similar optimization-based techniques that composite panoramas by choosing one source image per pixel have been explored in the context of traditional, single-viewpoint panoramas [Davis 1998; Agarwala et al. 2004]. The optimization framework used here is identical to Agarwala et al. [2004], except that the objective function is modified for this problem.

minutes). We then create higher-resolution versions using the hierarchical approach described by Agarwala et al. [2005]. Finally, some artifacts may still exist from exposure variations or areas where natural seams do not exist. We thus composite the final panorama in the gradient domain [Pérez et al. 2003; Agarwala et al. 2004] to smooth errors across these seams.

2.4 Interactive refinement

As described in Section 1.2, there is no guarantee that the user will like the transitions between regions of linear perspective determined by the MRF optimization. We thus allow for high-level interactive control over the result; the user should be able to express desired changes to the panorama without tedious manual editing of the exact seam locations. Also, the interaction metaphor should be natural and intuitive, so in a manner similar to the Photomontage system [Agarwala et al. 2004] we allow the user to draw various types of strokes that indicate user-desired constraints.

Our system offers three types of strokes. "View selection" strokes allow a user to indicate that a certain viewpoint should be used in a certain area of the panorama. A "seam suppression" stroke indicates an object through which no seam should pass; we describe a novel technique for propagating this stroke across the different positions the object might take in the different viewpoints. Finally, an "inpainting" stroke allows the user to eliminate undesirable features such as power lines through inpainting [Bertalmio et al. 2000].

2.4.1 View selection

The MRF optimization balances two competing concerns: creating a seamless composite, and rendering geometry from straight-on viewpoints. In some cases, the user may want greater control over this trade-off. We thus allow the user to paint strokes that indicate that a certain viewpoint should be used for a certain region of the composite. The mechanism of this stroke is simple: the user selects a projected source image I_i, and draws a stroke where that image should appear in the final panorama. We then constrain $L(p) = i$ for each pixel under the stroke during a second run of the MRF optimization. An example of this stroke (drawn in green) can be seen in Figure 10.

2.4.2 Seam suppression

Seam suppression strokes (drawn in red) allow the user to indicate objects in a scene across which seams should never be placed. As discussed in Section 1.2, the MRF optimization will try to route seams around objects that lie off of the dominant plane. However, in some cases no such seams exist, and the optimization is forced to find other transitions that are not visually noticeable. Sometimes the result is not successful; in Figure 9, for example, the white truck on the left and two cars on the right have been artificially shortened. While these seams may have a low cost according to equation (1), our knowledge of vehicles tells us that something is awry. This type of knowledge is difficult to encode in an algorithm, so we allow the user to indicate objects that should not be cut through. For example, the user can draw a stroke through the cars to indicate that no seams should cross that stroke.

Unlike view selection strokes, however, the semantic meaning of these strokes is specific to an object rather than a source image, and the position of the object may vary from viewpoint to viewpoint. Requiring the user to annotate an object in each source image would be tedious; instead, we ask the user to add a stroke to an object in one image only. The system then automatically propagates the stroke using knowledge of the 3D geometry of the scene provided by the structure-from-motion algorithm. We first establish a simple geometric proxy for the object annotated by the stroke, as described below, and then use that proxy to project the stroke into the other source images. Once the stroke is propagated, a stroke drawn over a pixel p in a source image I_i indicates for each pixel q adjacent to p that $L(p) = i$ if and only if $L(q) = i$. This constraint is easy to add to the cost function in equation (1).

The user draws a seam suppression stroke in one of the original, un-projected source images. If the user draws a stroke in the i'th source image, we project this stroke into another source image by assuming that the stroke lies on a plane parallel to the image plane of the i'th camera.[3] The system then selects the scene points that project within the stroke's bounding box in the i'th image. After transforming these points to the coordinate system of the i'th camera, a depth d for the plane is calculated as the median of the z coordinates of the selected scene points. Finally, a 3D homography is calculated to transform the stroke from the i'th camera to the j'th camera. The homography induced by a 3D plane [Hartley and Zisserman 2004] from camera i to camera j is

$$H_{ij} = K_j(R - tn^T/d)K_i^{-1},$$

where $R = R_j R_i^T$ and $t = -R_j t_i + t_j$. The vector n is the normal to the plane, which in this case is $(0,0,1)$. The matrix K_i is the matrix of intrinsic parameters for the i'th camera, which contains the focal length f_i. Once the homography is calculated, the system calculates its inverse to perform inverse warping of the stroke to camera j from camera i. Each stroke is projected to each source image from which it is visible. Finally, the stroke images are projected onto the picture surface; an example can be seen in Figure 9.

2.4.3 Inpainting

The final type of stroke indicates areas that should be inpainted by the system; we added this type of stroke because of the many unsightly power lines that often exist in a street scene. Power lines are very thin structures that generally lie off of the dominant plane, so it is virtually impossible to find seams that line them up. The cost function in equation (2) will sometimes automatically remove power lines, as in Figure 1. However, we lightly weight this term, since otherwise it sometimes removes desirable structures off of the dominant plane such as building spires.

Instead, we offer the user a simple inpainting approach to remove some of these artifacts that is inspired by Perez et al. [2003]. The user can draw strokes to indicate areas that should be filled with zero gradients during gradient-domain compositing. An example of this stroke, drawn in blue, can be seen in Figure 10; only a few minutes were required to remove the power lines in the sky, where they are most noticeable. More advanced hole-filling techniques (e.g., [Sun et al. 2005]) may allow removal of all of the power lines.

3 Results

We demonstrate our system by creating six multi-viewpoint panoramas: four of street scenes, one of a riverbank, and one of a grocery store aisle. The results in Figure 1 and 13 required no interactive refinement. View selection strokes were used in Figures 10 and 12, seam suppression strokes in Figure 9, and inpainting in Figure 10. All of the interactive refinement steps are detailed in the respective captions. Due to limited space, however, some of the source photographs, images that visualize seams, and images of the strokes are available only on the project website (which also includes full-resolution versions of all our panoramas). The effort by the user to

[3] More complicated geometric proxies are possible; for example, we could fit an oriented plane to the 3D scene points, rather than assuming the plane is parallel to the image plane. However, this simple proxy works well enough for the scenes we have tried.

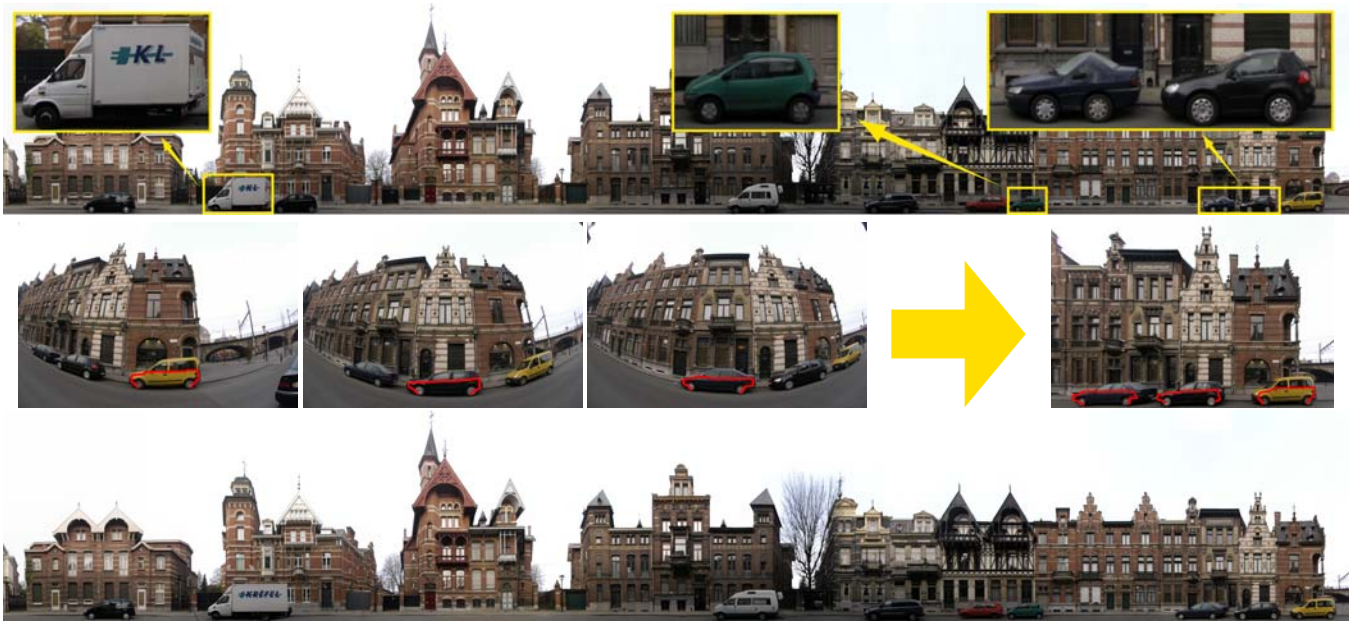

Figure 9 A multi-viewpoint panorama of a street in Antwerp composed from 114 photographs. First row: the initial panorama computed automatically. The result is good except that several vehicles (highlighted in yellow) have seen shortened. Second row: to fix the problem, the user draws one stroke on each car in some source photograph; shown here are strokes on the first, fifth, and eighth sources. Far right: the strokes are automatically propagated to all of the projected sources, of which the third is shown here. Ten strokes were drawn in all, one for each vehicle. Third row: the final result after interactive refinement.

produce the panoramas was fairly modest; for all of our results, the user interaction time was less than the time required to capture the input photographs. Our largest result (Figure 10) took roughly 45 minutes to capture, and less than 20 minutes of interaction (ignoring off-line computation).

The MRF optimization is not always able to produce good results automatically. The first row of Figure 9 shows shortened cars caused by poor seams placed in areas in front of the dominant plane. The middle of the scene in Figure 10 contains a large region off of the dominant plane; the automatically stitched result in the first row is thus unnatural. Both of these errors were fixed by interactive refinement. While careful examination of the final results will reveal occasional artifacts, we feel that our images successfully visualize the scenes they depict. Notice that our scenes contain significant geometry off of the dominant plane, such as crossing streets, trees, bushes, cars, etc., that are depicted in their correct aspect ratios (which strip panoramas would squash or stretch). Also notice that our panoramas are composed of large regions of ordinary perspective, rather than thin strips.

Our system is not able to produce effective multi-viewpoint panoramas for every scene. For example, the streets that we demonstrate here are fairly urban; suburban scenes are more challenging because the houses frequently lie at a range of different depths. Our system works best when the visible scene geometry frequently intersects the dominant plane, and the quality of the result degrades gracefully as the frequency of these intersections decreases. Some examples of failure cases are included on the project website.

4 Future work

There are several ways we can improve our results. A simple approach we are beginning to explore would allow us to handle shifts in the depth of the dominant plane. We have already shown that geometry at the dominant plane is aligned between the projected source images. If the picture surface is parallel to the camera path, however, a horizontal translation of the projected source images along the picture surface can be used to align geometry at any plane parallel to the dominant plane (this observation is true for the same reasons that a horizontal disparity can explain variations in depth for a stereo pair of images). Such translations could be used to stabilize geometry in a scene where the dominant plane shifts.

Another type of seam transition between viewpoints that our system does not currently exploit is the depth discontinuity. When we see around the silhouette of an occluding object, we expect to see geometry at some greater depth behind it. However, we are unlikely to notice if that geometry is depicted from a viewpoint slightly different from our own. We have experimentally confirmed that such transitions appear natural; taking advantage of them automatically, however, requires accurate recovery of scene depth and depth discontinuities. Our experiments with multi-view stereo techniques [Kang et al. 2001] suggest that this recovery is challenging for street scenes, since they contain many windows and other reflective surfaces.

Finally, although we currently only demonstrate panoramas of "long" scenes, our approach should also work for "long and tall" scenes. That is, one can imagine capturing a set of photographs along a 2D grid, rather than a line, and compositing them into one image. Such an approach could be used to image multiple floors of a mall or building atrium, which would be difficult to achieve with a video sequence and strip panorama. We hope to capture and experiment with such a data set in the near future.

Acknowledgements: We are deeply indebted to Noah Snavely for the use of his structure-from-motion system. This work was supported in part by a Microsoft fellowship and industrial gifts from Microsoft Research and Adobe Systems.

References

AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., DRUCKER,

S., COLBURN, A., CURLESS, B., SALESIN, D., AND COHEN, M. 2004. Interactive digital photomontage. *ACM Transactions on Graphics 23*, 3, 294–302.

AGARWALA, A., ZHENG, K. C., PAL, C., AGRAWALA, M., COHEN, M., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2005. Panoramic video textures. *ACM Transactions on Graphics 24*, 3 (Aug.), 821–827.

AGRAWALA, M., ZORIN, D., AND MUNZNER, T. 2000. Artistic multiprojection rendering. In *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, 125–136.

ASTOLA, J., HAAVISTO, P., AND NEUVO, Y. 1990. Vector median filters. *Proceedings of the IEEE 78*, 678–689.

BERTALMIO, M., SAPIRO, G., CASELLES, V., AND BALLESTER, C. 2000. Image inpainting. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, 417–424.

BOYKOV, Y., VEKSLER, O., AND ZABIH, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 11, 1222–1239.

BROWN, M., AND LOWE, D. G. 2005. Unsupervised 3D object recognition and reconstruction in unordered datasets. In *3D Imaging and Modeling (3DIM '05)*, 55–63.

DAVIS, J. 1998. Mosaics of scenes with moving objects. In *Computer Vision and Pattern Recognition (CVPR 98)*, 354–360.

EPAPERPRESS, 2005. http://epaperpress.com/ptlens/.

GOOGLE, 2005. http://earth.google.com.

GUPTA, R., AND HARTLEY, R. I. 1997. Linear pushbroom cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, 9, 963–975.

HARTLEY, R. I., AND ZISSERMAN, A. 2004. *Multiple View Geometry in Computer Vision*, second ed. Cambridge University Press.

HUBER, P. 1981. *Robust statistics*. John Wiley.

KANG, S. B., SZELISKI, R., AND CHAI, J. 2001. Handling occlusions in dense multi-view stereo. In *Computer Vision and Pattern Recognition (CVPR 2001)*, vol. 1, 103–110.

KASSER, M., AND EGELS, Y. 2002. *Digital Photogrammetry*. Taylor & Francis Inc.

KOLLER, M., 2004. http://www.seamlesscity.com.

KOLMOGOROV, V., AND ZABIH, R. 2002. What energy functions can be minimized via graph cuts? In *European Conference on Computer Vision (ECCV)*, 65–81.

KUBOVY, M. 1986. *The psychology of perspective and renaissance art*. Cambridge University Press.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics 22*, 3, 277–286.

LEVIN, G., 2005. An informal catalogue of slit-scan video artworks. http://www.flong.com/writings/lists/list_slit_scan.html.

LEVOY, M., CHEN, B., VAISH, V., HOROWITZ, M., MCDOWALL, I., AND BOLAS, M. 2004. Synthetic aperture confocal imaging. *ACM Transactions on Graphics 23*, 3 (Aug.), 825–834.

LOWE, D. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2, 91–110.

MITSUNAGA, T., AND NAYAR, S. K. 1999. Radiometric self calibration. In *Computer Vision and Pattern Recognition (CVPR '99)*, 374–380.

PELEG, S., ROUSSO, B., RAV-ACHA, A., AND ZOMET, A. 2000. Mosaicing on adaptive manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 10, 1144–1154.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Transactions on Graphics 22*, 3, 313–318.

RADEMACHER, P., AND BISHOP, G. 1998. Multiple-center-of-projection images. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, 199–206.

RAV-ACHA, A., SHOR, Y., AND PELEG, S. 2004. Mosaicing with parallax using time warping. In *2004 Computer Vision and Pattern Recognition Workshop (CVPRW'04)*, vol. 11.

ROMAN, A., GARG, G., AND LEVOY, M. 2004. Interactive design of multi-perspective images for visualizing urban landscapes. In *Proceedings of IEEE Visualization*, 537–544.

SEITZ, S. M., AND KIM, J. 2003. Multiperspective imaging. *IEEE Computer Graphics & Applications 23*, 6, 16–19.

SNAVELY, N., SEITZ, S., AND SZELISKI, R. 2006. Photo tourism: exploring photo collections in 3D. *ACM Transactions on Graphics 25*, 3, To appear.

SUN, J., YUAN, L., JIA, J., AND SHUM, H.-Y. 2005. Image completion with structure propagation. *ACM Transactions on Graphics 24*, 3 (Aug.), 861–868.

SZELISKI, R., AND SHUM, H.-Y. 1997. Creating full view panoramic mosaics and environment maps. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, 251–258.

UYTTENDAELE, M., EDEN, A., AND SZELISKI, R. 2001. Eliminating ghosting and exposure artifacts in image mosaics. In *Computer Vision and Pattern Recognition (CVPR 01)*, 509–516.

WEXLER, Y., AND SIMAKOV, D. 2005. Space-time scene manifolds. In *International Conference on Computer Vision (ICCV'05)*, vol. 1, 858–863.

WOOD, D. N., FINKELSTEIN, A., HUGHES, J. F., THAYER, C. E., AND SALESIN, D. H. 1997. Multiperspective panoramas for cel animation. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, 243–250.

YU, J., AND MCMILLAN, L. 2004. A framework for multiperspective rendering. In *Proceedings of the 15th Eurographics workshop on Rendering Techniques*, 61–68.

YU, J., AND MCMILLAN, L. 2004. General linear cameras. In *European Conference on Computer Vision (ECCV 04)*, 14–27.

ZHENG, J. Y. 2003. Digital route panoramas. *IEEE MultiMedia 10*, 3, 57–67.

ZOMET, A., FELDMAN, D., PELEG, S., AND WEINSHALL, D. 2003. Mosaicing new views: The crossed-slits projection. *IEEE Transactions on PAMI 25*, 6, 741–754.

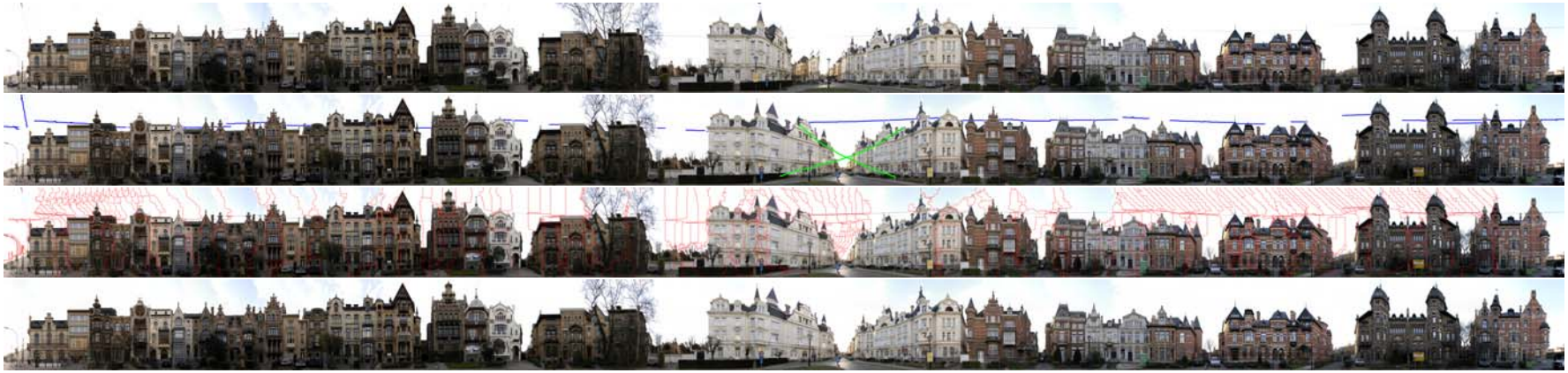

Figure 10 Our longest multi-perspective panorama of a street in Antwerp composed from 280 photographs. First row: the automatically computed panorama. Notice the artifact in the middle of the crossing street, due to the large region of geometry off of the dominant plane. Second row: a view selection stroke is drawn in green to choose a single view for the middle region, and blue inpainting strokes remove power lines across the sky. Third row: the computed seams for the final result shown in red. Fourth row: the final result.

Figure 11 A multi-perspective panorama of a grocery store aisle composed from 60 photographs. Notice the local perspective effects in the depiction of the crossing aisle; one view selection stroke was required to force this region to come from a single source image. Because of the low light in the scene, this sequence was shot with a camera on a tripod to allow for longer exposures. A video sequence would likely be much noisier, since long exposures would not be possible.

Figure 12 A multi-perspective panorama of a street in Charleston, South Carolina composed from 146 photographs. This result demonstrates that our system can handle some motion in the scene, such as the moving cars and the many walking people seen here. Two view-selection strokes are used to select the most straight-on views for the two crossing streets. The strokes and initial panorama can be seen on the project website.

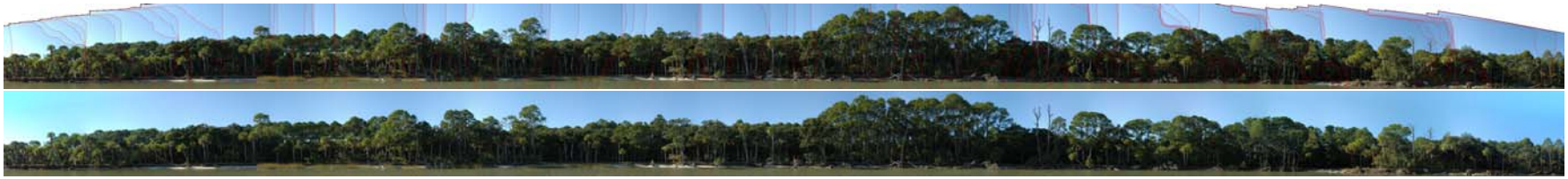

Figure 13 A multi-perspective panorama of a river bank near Beaufort, South Carolina. First row: the computed seams, composited on the panorama *before* gradient-domain compositing. Second row: the final result (no interactive refinement was necessary). Notice that some areas of the panorama are not covered by source photographs; we fill these areas with zero gradients during gradient-domain compositing. Some unusual variations in sky color can also be seen in the result. These variations are caused by a bright sun to the left of the field of view of the camera, causing a gradation from left to right in each source photograph. Note that gradient-domain compositing reduces but does not eliminate this variation entirely.