Dynamic Mosaics

Rahul Garg
University of Washington
rahul@cs.washington.edu

Steven M. Seitz
University of Washington, Google
seitz@cs.washington.edu

Abstract

Past mosaicing approaches stitch a set of photos into a single static mosaic. We present a novel approach where we visualize a photo collection in an interactive viewer that allows the user to smoothly and seamlessly transition between a collection of local mosaics instead of a single static mosaic. Such an approach works with more general photo collections than possible with static mosaicing while preserving the straight lines in the scene. Moreover, the viewer dynamically selects optimal seams between images in a semi-online fashion that handles misalignment, scene motion and parallax and also recreates dynamic aspects such as moving objects and exposure changes.

1. Introduction

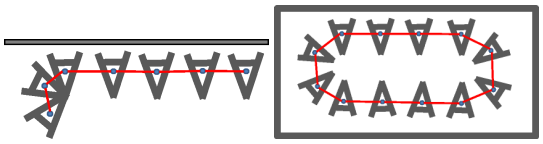
Image mosaicing techniques provide a way to composite several photos into a wider field-of-view composite. When rendered with an appropriate viewer, these techniques synthesize distortion-free, perspective-correct interactive scene visualizations. The fundamental limitation, however, is that perspective-correct mosaicing is possible only if the camera does not translate, or if the scene is planar (or distant). Recent work on non-perspective panoramas enables more general camera motions [21, 19, 16], at the expense of introducing distortions (e.g., straight lines in the scene become curved), by relaxing perspective-correctness. Rather than introducing scene distortion, we propose to relax the requirement that all photos be composited into a *single* mosaic. We compute a collection of local mosaics that can be interactively traversed in a new kind of dynamic scene viewer. Further, each of these local mosaics preserve straight lines in the scene. The key aspects of our approach are:

- **Dynamic Projection:** The viewer uses a dynamic projection model to transition between local mosaics. Transitions are subtle as to be barely noticeable, resulting in a continuous and nearly distortion-free scene visualization. The approach generalizes to a class of image collections which we call *locally stitchable*, and subsumes rotational and translational mosaics.
- **Dynamic Seam Selection and Blending:** Misalignment between images, scene motion and parallax lead to ghosting artifacts in traditional mosaicing. The proposed viewer dynamically selects and blends across a set of precomputed seams to avoid artifacts and allows us to recreate dynamic aspects of the scene, e.g., people walking, trees waving, etc. as the user navigates.
- **Dynamic Exposure Selection:** Rather than removing exposure differences entirely, the viewer dynamically adjusts the exposure based on the part of the scene currently visible. This imitates the behavior of the human eye, e.g., when looking at a bright light source, rest of the scene appears dark.

We call such mosaics *dynamic mosaics.*

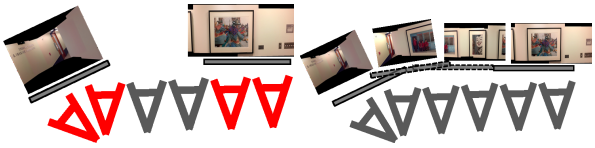
There are three major challenges that we address. First, we pose the problem of computing a local mosaic as an optimization problem which minimizes the distortion in the current view and can be solved in real time. Secondly, blending techniques (e.g., graph cuts [4], multi band blending [7]) are not fast enough to work in real-time. We come up with a semi-online approach that allows us to do blending in real time while recreating scene motion. Finally, we come up with a method to dynamically compute locally optimal exposure.

Mosaics are a simple form of Image Based Rendering (IBR), a class of techniques that includes more sophisticated methods like lightfields [15, 11], plenoptic modeling [17], Photo Tourism [23], and so forth. A key difference, however, is that mosaics rely only on 2D warps (homographies) and 2D navigation (pan, zoom). This reliance on 2D warps makes mosaics robust enough to work in practice, and the simple 2D navigation appeals to the most users. These factors have enabled mosaics to enjoy a level of penetration and widespread use that other IBR methods have not. Our work seeks to achieve a greater range of behaviors (generalized camera paths, scene motion, dynamic exposure) than is possible with existing mosaic techniques, while staying in the mosaic framework (2D warps, 2D navigation).



(a) (b)

Figure 1: Locally Stitchable Collections. (a) A locally stitchable collection that is a combination of translational and rotational motion. (b) Adjacent images need to be only approximately stitchable. This shows an example of a rectangular room whose walls have been captured by a combination of translational and approximately rotational motion. A single rotational mosaic will not provide a uniform sampling of the walls.



(a) (b)

Figure 2: (a) Photos at either end of the collection can be stitched into local mosaics by projecting them onto different planar surface. (b) Our system smoothly interpolates the planar projection as the user navigates allowing smooth transitions between these local mosaics.

2. Overview

Let us first define *locally stitchable image collection*s. We consider two images as *stitchable* if they overlap and can be aligned via a 2D transform, i.e., a 3×3 homography matrix. E.g., images captured by a rotating camera or images of a planar scene are *stitchable*. Consider a graph with images as nodes and an edge between every pair of stitchable images. We call an image collection *locally stitchable* if the resulting graph is connected (Fig. 1a).

One can relax the definition of stitchablity. E.g., two images can be aligned well if they are taken from *approximately* the same spot or if the scene is *approximately* planar. This allows us to extend the definition to cases like those shown in Figure 1b. Any collection which consists of sequences of rotating camera or translating camera (in front of a planar scene) is included in the definition.

We now briefly describe the three key dynamic aspects of our system.

2.1. Dynamic Projection

Consider the photo collection shown in Figure 2a, a combination of rotational and translational motion. Photos at the left end of the collection can be composited into a rotational mosaic. Similarly photos at the right end can be composited into a translational mosaic since the scene is planar. The proposed viewer interpolates between the two mosaics by smoothly varying the projection as the user navigates from one end to the other. This is achieved by selecting a planar projection based on the current view which changes smoothly as the user moves around (Fig. 2b).

(a) (b)

Figure 3: (a) An arbitrary planar projection. (b) Optimized planar projection.

How should one select the optimal planar projection for a given view? An arbitrary planar projection may lead to geometric distortion (Fig. 3a).

We formulate the planar projection selection problem as an optimization problem that seeks to minimize the geometric distortion of images in view (Fig. 3b). A planar projection is represented by a 3×3 homography matrix H. We need to quantify the geometric distortion induced by a homography H. For H to not introduce any affine or perspective distortion, it should preserve the original rectangular shape of the input image. While this can be measured in a number of ways, one way is to see where the horizontal direction $[1, 0, 0]$ and vertical direction $[0, 1, 0]$ are mapped by H. Hence, a measure of distortion can be $||H[1, 0, 0]^T - [1, 0, 0]^T||^2 + ||H[0, 1, 0]^T - [0, 1, 0]^T||^2$ (This also penalizes $2D$ rotation and uniform scaling which do not introduce any affine or perspective distortion but we talk about that later). Such a penalty can be written in the form $||H^{res} - I||^2$ where H^{res} is simply H where the last column has been replaced by $[0, 0, 1]^T$, i.e., the translation free part of H. This can also be interpreted algebraically – the distance between the translation free component of H and the identity transform. Such a penalty is also consistent with Zorin and Barr's [28] observation that for a linear perspective camera, objects in the center of the image never look distorted.

Given this measure of geometric distortion, the viewer chooses a projection that minimizes a weighted sum of distortions of the images in view. We show in Section 4.1 that such a penalty measure leads to a linear objective which can be solved and continuously updated at frame rate as the user navigates.

2.2. Dynamic Seam Selection and Blending

Finding optimal seams between images to avoid ghosting artifacts due to scene motion and parallax is a common approach [24, 9, 2, 1]. However, in our case we do not have a single mosaic for which we can precompute seams beforehand. As it is impractical to compute seams in real time for each local mosaic we render, we precompute seams for a set of local mosaics and then transition between those seams at render time based on user's viewpoint.

Such an approach has two advantages. First, the seams are *locally optimal,* i.e., they prefer images which show least distortion in the current view. Second, as we cross-fade between different local seams as the user navigates, it recreates the dynamic aspects of the scene like parallax and scene motion. While the basic formulation computes seams whose purpose is to minimize the stitching artifacts, one can add additional constraints to explicitly control how seams are correlated with the motion in the scene (Sec. 5.1). Further, we also implement real time multi band blending across these seams (Sec. 4.4).

2.3. Dynamic Exposure

Cameras in auto-mode change exposure dynamically based on scene brightness. Exposure differences lead to artifacts when stitching a single static mosaic and sophisticated exposure adjustment and blending algorithms are required to avoid such artifacts [24, 1, 6, 14]. However, exposure differences across images provide valuable cues about variation in brightness of the different parts of the scene. The system estimates these differences in an offline process and uses them to dynamically adjust the exposure in the online viewer as the user navigates providing a more realistic experience.

Brown and Lowe [6] use a simple linear gain model, i.e., a multiplicative factor to model exposure changes which is equivalent to multiplying the color channels of an image by a single scalar. We relax this model by assuming that three independent scalars can be used for the three color channels. This allows to correct for not only exposure changes but also color balance. This is equivalent to using a diagonal transform color correction model which suffices for many cases [10]. We compute these scalars using an approach similar to Brown and Lowe [6] (Section 3.4). This simple model in conjunction with seam selection and blending avoids almost all of the artifacts.

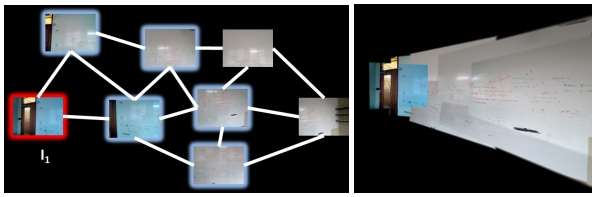
While rendering, Kopf *et al.* [14] use dynamic tone mapping of HDR images based on the local part of the mosaic the user is viewing. Analogously, we compute locally optimal diagonal transform as the user navigates. The transform is optimal in the sense that it tries to preserve the original exposure and color balance of the mosaic currently in view.

We now proceed to give technical details of the approach which can be broken down into two parts – offline processing and online viewer.

3. Offline Processing

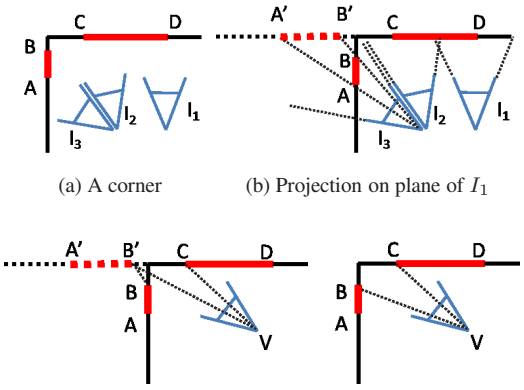
3.1. Image Matching

We match images pairwise using SIFT features and RANSAC [12]. We declare two images to be *stitchable* if the homography found by RANSAC has at least 40 inliers. Since the graph of images is connected w.r.t. stitchability relation (locally stitchable), we can determine the homography between every pair of images by finding the shortest path between them and chaining homographies along the path. This may lead to accumulation of errors along long paths but images connected by long paths are not likely to overlap and hence such errors do not lead to stitching artifacts in practice.



(a) (b)

Figure 4: (a) The neighbor set S_1 (images with blue border) corresponding to image I_1. It consists of images that can be projected onto the plane of I_1 to generate a local mosaic. (b) The local mosaic generated by compositing images in the set S_1 onto the plane of I_1.



(a) A corner (b) Projection on plane of I_1

(c) Projection seen through virtual camera V (d) Single perspective view

Figure 5: A local mosaic is a piecewise perspective projection of the scene that preserves straight lines.

3.2. Local Mosaics

For each image I_i, we define the set of neighbors S_i as images that can be stitched to I_i to generate a local mosaic (Figure 4a). Let us say that I_j is *compatible* with I_i if the optical axis of I_j intersects the plane of I_i in the positive direction, i.e., I_j can be projected onto the plane of I_i without flipping the image. Then, we define S_i as the largest connected set of images in the graph containing I_i which are compatible with I_i. Algorithmically, we run breadth first search starting from I_i to build the set S_i.

Images in the set S_i can be stitched to I_i to generate a local mosaic (Figure 4b). We prove that each such local mosaic is a piecewise-perspective projection of the scene that preserves straight lines.

Notice that if the images in the neighbor set S_i correspond to a rotating camera or if the scene is planar, then

the resulting local mosaic is a single perspective projection. Hence, the challenging case faced in this paper is a non planar scene captured by a camera which is not purely rotating. Such a scene can be captured by a locally stitchable collection if a rotating camera covers the planar discontinuities in the scene (Figure 5a). Let AB and CD be two line segments in the scene on two different planes. The local mosaic corresponding to S_1 is generated by projecting all images onto the plane of I_1, thus projecting AB as $A'B'$ (Figure 5b). Now, while rendering, S_1 is viewed through a virtual camera V (Section 2.1) which sees the image $A'B'$ of AB instead of the true AB (Figure 5c and 5d). However, $A'B'$ is also a straight line. Hence, while the two planes of the corner appear under different perspective transforms in V, straight lines in the scene, which are confined to these planes, remain straight. The observation is generalizable as long as the homographies between *stitchable* images correspond to real world planes (holds if the image matching does not make errors). Zelnik-Manor *et al*. [27] also use such piecewise-perspective projections that are selected interactively to reduce distortion in spherical panoramas.

3.3. Optimal Seams via Graph Cuts

We precompute optimal seams for each of the local mosaics. At render time, as the user transitions between different local mosaics, we crossfade between corresponding seams.

Optimal seams should avoid stitching artifacts by minimizing pixel difference across them. Further, they should choose pixels from images which appear less distorted geometrically. We formulate an objective which captures these two goals and minimize it using Markov Random Field (MRF) optimization using an approach similar to that of Agarwala *et al*. [1]. We use $||H_j^{res} - I||$ to quantify distortion where H_j is the homography that aligns I_j with I_i. Further, we adjust the pixel values using the exposure compensation factors we compute in Section 3.4. Computed seams are stored as binary masks; for each $I_j \in S_i$, we have a mask M_{ij}. At run time, we do not need to store and load all the local mosaics, the system uses these single channel binary masks along with the input images to composite local panoramas on the fly.

3.4. Global Exposure Compensation

Brown and Lowe [6] first find pairwise gain factors for overlapping images by considering corresponding pixels, and then use pairwise factors to estimate per image gain factors by minimizing a global objective function. We use a similar approach to estimate per channel gain factors for each image. To estimate pairwise gain factors, we consider all corresponding pixels and use RANSAC to estimate the most common triplet of channel wise gain (r_{ij}, g_{ij}, b_{ij}) (r_{ij} denotes the gain between I_i and I_j). Such an approach is more robust than the mean used in Brown and Lowe's approach [6]. To estimate gain per image for each channel, e.g., r_i for red channel of I_i, we consider all pairwise equations $\frac{r_j}{r_i} = r_{ij}$ and solve using least squares for each color channel independently (and setting $r_1 = 1$). Brown and Lowe [6] use a slightly different objective which requires them to use a prior to encourage gains close to 1. Further, we weigh the equations by the number of RANSAC inliers to reflect the confidence in each equation. At run time, we use the estimated values to compensate for exposure differences between images and compute a locally optimal exposure value as the user navigates (Sec. 4.3).

4. Online Viewer

4.1. Dynamic Projection

The user interacts with the mosaic using drag and zoom interaction. The viewer updates the projection to provide the best view of the images currently visible in real time as the user drags the mouse.

We first describe how dragging works. Since we use a planar projection at all times, we can model the transformation of each image by a homography. Suppose the current transforms applied to the images $I_1, I_2, ..., I_n$ are $H_1, H_2, ..., H_n$ respectively, i.e., these are the optimal homographies for the current view (initialization is done by setting $H_1 = I$ and using the pairwise homographies computed in Sec. 3.1), and the user drags by $(\delta x, \delta y)$. Let $H^{trans}(\delta x, \delta y)$ be the homography that induces a translation of $(\delta x, \delta y)$. We first apply this homography to the current view, i.e., the new transformations of images are $H^{trans}(\delta x, \delta y)H_1, ..., H^{trans}(\delta x, \delta y)H_n$. Let H_i refer to the updated transformation $H^{trans}(\delta x, \delta y)H_i$. Solving for a new planar projection is equivalent to solving for a homography H that will be applied to all the images, i.e., the geometric transforms of the images under this new projection will be $HH_1, HH_2, HH_3, ..., HH_n$. We want to choose an H that minimizes the distortion of images.

As described in Section 2.1, distortion of a single image in the collection can be measured by $||(HH_i)^{res} - I||$. However, the global homography H only seeks to minimize the distortion in the images and not translate the image composite. Hence $H(0,2) = H(1,2) = 0$, i.e., the last column of H is $[0, 0, 1]^T$ and H is already of the form H^{res} and the distortion can be written as $||HH_i^{res} - I||$. We compute importance weights w_i for the images (explained later) and minimize the weighted sum of distortions. Mathematically,

$$H_{opt} = \arg\min_H \sum_{i=0}^{n} w_i ||HH_i^{res} - I||^2 \qquad (1)$$

However, it is a non linear optimization problem because of the dehomogenization operation, i.e., the bottom right

entry of the product HH_i^{res} needs to be 1. However, algebraically, the above objective is similar to

$$H_{opt} = \arg\min_H \sum_{i=0}^{n} w_i ||H^{-1} - H_i^{res}||^2 \quad (2)$$

as they both encourage the product HH_i^{res} to be close to I. Since H_i^{res} are known and can be normalized by setting the bottom right element to 1, it yields a simple least squares objective and leads to a simple closed form solution

$$H_{opt}^{-1}(a,b) = \frac{\sum_{i=1}^{n} w_i H_i^{res}(a,b)}{\sum_{i=1}^{n} w_i} \quad (3)$$

H_{opt} can be computed by inverting the solution obtained from Eq. (3). In practice we avoid the explicit inversion and directly compute the products $H_{opt}H_i$ from H_{opt}^{-1} by solving the linear equation $H_{opt}^{-1}X = H_i$ which is more numerically stable. This computation is fast enough to be done in real-time on standard PCs. Note that H_{opt} does not induce any translation, i.e., the origin or the center of the mosaic remains stationary. However, it had moved exactly by$(\delta x, \delta y)$ before the application of H^{opt} due to application of $H^{trans}(\delta x, \delta y)$. This is the behavior that the user expects to see on dragging by $(\delta x, \delta y)$, i.e., the center of the mosaic moves by $(\delta x, \delta y)$.

We now describe how we compute importance weights w_i for different images that are used in the optimization step. Distortion of images near the center of the viewport would be more noticeable. Hence we weight such images higher. Let (x_{i0}, y_{i0}) be the location of the center of I_i under transformation H_i. In all of our computations, we normalize the image dimensions to be $[-0.5, 0.5] \times [-0.5, 0.5]$. Our viewport is $[-1, 1] \times [-1, 1]$ with the origin at the center of the viewport. Then we define the weight of I_i as

$$w_i = \begin{cases} 0 & \text{if } max(|x_{i0}|, |y_{i0}|) > 0.5 \\ 0.5 - max(|x_{i0}|, |y_{i0}|) & \text{otherwise} \end{cases} \quad (4)$$

Such a definition of weights ensures that they change smoothly as the user moves around ensuring that the optimal projection also changes smoothly. If the user zooms in very far, it might happen that all weights are zero. In that case, we set the weight of the image that is closest to the origin to be 1. We refer to the image with the highest weight as the *central image*. Any image that does not belong to the neighbor set S of the central image is set to have a weight zero. Finally the weights are normalized such that $\sum_i w_i = 1$.

Application of H_{opt} changes the transforms of the images and consequentially the weights. Ideally one should solve this in an iterative fashion until convergence. However, such an approach is too slow for interactive real time rendering. Hence, we instead amortize the iterations over render cycles, i.e., at each render cycle we compute the weights and solve for an optimal H_{opt}.

Zooming: Dragging interaction works by allowing the user to control the translation of the center of the mosaic while the optimization is invariant to it. Hence, zoom interaction should work similarly – allow the user to zoom in or zoom out with optimization being invariant to uniform scaling. However, the objective in Eq. (1) penalizes uniform scaling of images.

To make the objective invariant to scaling, we first compute the *scale factor* of each image. For that we need to associate a scaling factor $scale(H)$ with a general homography H. Consider a unit square centered at the origin transformed by the homography. We define the scale factor as the square root of the area of the transformed square. Since we normalize the image dimensions to $[-0.5, 0.5] \times [-0.5, 0.5]$, this measures the change in area of the image under that transformation. For our image collection, for every pair of stitchable images (I_i, I_j), we find this pairwise scaling factor $s_{ij} = scale(H_{ij})$ where H_{ij} is the homography that warps I_j to I_i. We then find the scale factor s_i for each image relative to I_1 in a way similar to how we find the exposure gain factors. More precisely, we set $s_1 = 1$ and solve for $s_2, s_3, ..., s_n$ given the pairwise constraints $\frac{s_i}{s_j} = s_{ij}$. It reduces to a least squares system after applying log.

Now, when solving for the best planar projection in our objective, we normalize for the scale factor, i.e., we want the product HH_i^{res} to be close to

$$\begin{bmatrix} s_i & 0 & 0 \\ 0 & s_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5)$$

instead of identity. Correspondingly, we change the objective to $H_{opt} = \arg\min_H \sum_{i=0}^{n} w_i ||H^{-1} - H_i^{res} S_i^{-1}||^2$ where S_i is the matrix defined in Eq. (5) whose inverse is trivial to compute since it is a diagonal matrix. To allow the user to zoom in or zoom out of the mosaic, the viewer maintains a current scale factor s_f which is updated based on user input and we use

$$S_i = \begin{bmatrix} s_i s_f & 0 & 0 \\ 0 & s_i s_f & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6)$$

in our objective.

To support images at different scales, we also change our weighing function – give more weight to the image that is at the right resolution for the current scale factor. Hence, we now define the weight as $w_i' = \frac{w_i}{1+|log(s_i s_f)|}$, i.e., amplify the weights of the images for which $s_i s_f$ is close to 1.

4.2. Cross Fades between Seams

The system uses the seams corresponding to the current *central image* (image with the maximum weight), i.e., if the

current central image is I_i it uses the seams corresponding to S_i. As the central image changes, the system smoothly transition between different graph cut seams. The renderer works by keeping an alpha mask A_i [20] corresponding to each image I_i which are updated continuously. We initialize each A_i to be zero everywhere. Let I_m be the current central image. Let M_{mj} denote the mask corresponding to image $I_j \in S_m$. While the original mask computed using graph cuts is binary, it can have real values due to blending across seams (Section 4.4). Then at each render cycle, we update the current masks as

$$A_i(x,y) = \begin{cases} A_i(x,y) - \beta & \text{if } M_{mj}(x,y) < A_i(x,y) - \beta \\ A_i(x,y) + \beta & \text{if } M_{mj}(x,y) > A_i(x,y) + \beta \\ M_{mj}(x,y) & \text{otherwise} \end{cases} \tag{7}$$

M_{mj}'s are loaded as textures and these updates are done in hardware using a custom shader. The parameter β controls the speed of crossfade.

4.3. Dynamic Global Exposure Compensation

We solve for scalar gain factors for each color channel independently. E.g., for the red channel, we use

$$r_0 = \arg\min_r \sum_i w_i (log\frac{r}{r_i})^2 \tag{8}$$

where r_i is the gain factor corresponding to I_i computed in Section 3.4, and similarly for the green and blue channels. The resulting exposure gain that is applied to I_i is $\frac{r_0}{r_i}$. From the objective, one can see that the gain corresponding to images with higher weights be closer to 1. The weights are the same as those used in Eq. (3). The above objective permits a closed form solution of the form $r_0 = \prod_i r_i^{w_i}$. Further, we do not apply the compensation immediately and add a delay in a way similar to Kopf *et al*. [14] to simulate the behavior of human eye which takes some time to adjust to the change in lighting.

4.4. Real-time Multi Band Blending

We also implement real time multi band blending [7] across seams to remove any remaining artifacts. We precompute the Laplacian pyramid corresponding to each of the images and store different levels of the pyramid as individual textures. We add an offset of 255 to the difference images to ensure that all values are positive and then compress them to range $[0, 1]$ before loading them as textures. As higher levels of pyramid are stored at lower resolution, memory requirements are bounded.

At render time, for each I_i we build a Gaussian pyramid of the corresponding alpha mask A_i. This is done on the GPU using OpenGL's texture downsampling. Final rendering is then produced by accumulating the renderings corresponding to each level of the pyramid in accumulation buffer after adding back proper offsets. OpenGL's texture upsampling is used to upscale each rendering to the full resolution. Further, for each level, we also apply exposure compensation factors (Sec. 4.3) while adjusting for offset and compression.

(a) Camera path (b) Captured photos

Figure 6: A collection of 43 photos taken on a staircase capturing paintings on the wall.

Figure 7: Dynamic mosaic corresponding to the staircase collection. The top row shows a sequence of zoom outs corresponding to the part shown on the left. The bottom row shows zoomouts for another part. User can smoothly move between different parts simply by dragging the mouse.

Using too many pyramid levels may lead to quantization artifacts because of the limited resolution of accumulation buffer. Hence we use a two level scheme with a downsampling factor of 7 similar to Brown *et al*. [5]. Further, multi band blending is ineffective if the seams are very close to the image boundary as the blending alpha is truncated by image edges. Hence, we encourage seams that are away from the boundaries by penalizing pixels near the boundary in graph cuts and divide each pixel value in the final composite by the composited alpha value in case there is truncation and alpha value does not sum to 1.

5. Results

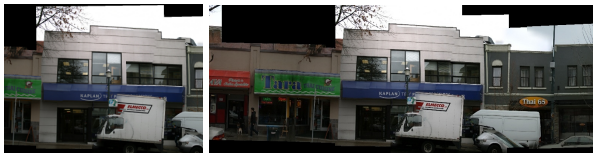
We show some results here. Please view the video [25] as it is difficult to capture the dynamics of the system in still figures. The renderer works at interactive rates on standard PCs.

Figure 6 shows a collection of 43 photos. Traditional image stitching software is unable to stitch them into a single mosaic. However, in our system the user can browse the scene by using simple drag and zoom interaction (Fig. 7).

The system can also handle photos at different scales. The user is able to smoothly zoom in with the system choosing the images at the right resolution. The weighing function takes into account scale of the image and hence

(a) (b) (c)

Figure 8: (a) A collection of 31 photos of a street side. (b) In presence of parallax, stitching artifacts are unavoidable at image edges. (c) However, seam selection is able to mask such artifacts.



(a) (b)

(c)

Figure 9: A sequence of zoomouts from our viewer.

(a) (b)

Figure 10: Two renderings corresponding to the left end of Fig. 9 bottom. As the user drags to this part of the mosaic, the system smoothly updates the rotation to show as horizontal as possible view.

gives higher weight to images which are at the current scale of the viewer. Please view the video [25] to see the results. The system is also capable of handling walking forward/backward motion provided the adjacent images can be aligned via a homography.

Figure 8a shows photos of a street side shot using a hand held camera. Because of significant parallax between images, we solve for a similarity transform between images instead of a full homography to reduce the degrees of freedom. Misalignments due to restricted matching model and parallax are minimal and are compensated by seam selection and blending (Fig. 8c). Fig. 9 shows the collection being browsed in our viewer. Due to extreme parallax in the scene, the stitching artifacts are more evident here, e.g., the vehicles on the road. Unlike the system of Agarwala *et al*. [1], the camera motion is not restricted to translation; some of the taller buildings have been captured by tilting the camera up and down. Further, dynamic seams emphasize parallax similar to the Street slide system of Kopf *et al*. [13].

Stitching such long facades can often lead to curved mosaics if the camera is not horizontal or not looking head on at the facade as can be seen on the left of Fig. 9 bottom. However, because we penalize in-plane rotation in the objective in Eq. (1), the viewer smoothly rotates the mosaic as the user navigates making the current view as horizontal as possible (Fig. 10).

(a) (b) (c)

Figure 11: (a) 6 out of 18 input photos. (b) A rendering where the interior is well exposed while the window on the right is saturated. (c) As the windows come into view, a shorter exposure is selected; view outside the window is visible.

(a) (b) (c)

Figure 12: (a) 2 out of 18 photos from a rotating camera; the person moves between shots. (b),(c) Renderings from the viewer which stitches the images into a larger field of view. Dynamic seams recreate the motion without cutting through the person as the user pans from right to left.

Figure 11 shows a rotational mosaic that adjusts dynamically to the varying brightness in the scene.

5.1. Dynamic Motion Mosaics

Dynamic seams allow us to recreate the motion in the scene (Fig. 12). While there exists specialized approaches [3, 22] for rotational mosaics, our framework easily allows for recreating motion in the scene for more general camera motion. Our formulation automatically chooses seams to minimize artifacts, we allow the user to add additional constraints to achieve more targeted effects like the stroboscopic effect shown in Fig. 13c where the dynamic mosaic shows the history of the jump up to the current time with time increasing as the user pans from left to right. This mosaic is created from a panning video where the location of the jumper was marked in each frame. This can be achieved by adding a constraint in local graph cuts. Assuming that the images are indexed by the frame number, then while computing the seams corresponding to S_i, we add a constraint to choose pixels corresponding to the jumper from I_j if $j \leq i$ and not choose such pixels if $j > i$. The method can also be seen as video summarization analogous to [8].

(a) Frames from a panning video (b) Masked location of jumper in frames

(c) Dynamic mosaic; jumper moves forward as the user pans from left to right.

Figure 13: Dynamic motion mosaic from a panning video.

6. Discussion

We proposed an approach for creating dynamic mosaics of scenes from photos. They provide intuitive navigation while allowing for more general camera motion and being dynamic in nature. Using a more versatile exposure compensation model and determining pairwise homographies in a globally optimal way that allows loop closing are some of the directions we will like to explore. Another desirable feature is to be able to show the summary of the entire collection if the user is zoomed out all the way. Image collages [26, 18] can layout a collection on a planar surface at the expense of stitching artifacts. Peleg *et al*. [19] relax perspective-correctness and find adaptive manifolds for stitching mosaics from video sequences. However, it can result in arbitrary bending of scene lines and does not work for photos. Ultimately, the approach should allow intuitive capture and navigation of arbitrary scenes, e.g., a house with multiple rooms, where dynamic mosaics for individual rooms are connected to each other in an intuitive fashion.

Acknowledgment: This work was supported in part by National Science Foundation grants IIS-0811878, IIS-0963657, the University of Washington Animation Research Labs, Intel, Microsoft, and Google.

References

[1] A. Agarwala, M. Agrawala, M. Cohen, D. Salesin, and R. Szeliski. Photographing long scenes with multi-viewpoint panoramas. In *Proc. SIGGRAPH*, pages 853–861, 2006.

[2] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen. Interactive digital photomontage. In *Proc. SIGGRAPH*, pages 294–302, 2004.

[3] A. Agarwala, K. C. Zheng, C. Pal, M. Agrawala, M. Cohen, B. Curless, D. Salesin, and R. Szeliski. Panoramic video textures. *ACM Trans. Graph.*, 24(3):821–827, 2005.

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11):1222 –1239, nov 2001.

[5] M. Brown and D. Lowe. Recognising panoramas. In *Proc. ICCV*, pages 1218 –1225, 2003.

[6] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *IJCV*, 74:59–73, 2007.

[7] P. J. Burt and E. H. Adelson. A multiresolution spline with application to image mosaics. *ACM Trans. on Graphics*, 2:217–236, October 1983.

[8] Y. Caspi, A. Axelrod, Y. Matsushita, and A. Gamliel. Dynamic stills and clip trailers. *Vis. Comput.*, 22(9):642–652, 2006.

[9] J. Davis. Mosaics of scenes with moving objects. In *Proc. CVPR*, pages 354–1361, 1998.

[10] G. Finlayson, M. Drew, and B. Funt. Diagonal transforms suffice for color constancy. In *Proc. ICCV*, pages 164–171, 1993.

[11] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In *Proc. SIGGRAPH*, pages 43–54, 1996.

[12] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[13] J. Kopf, B. Chen, R. Szeliski, and M. Cohen. Street slide: browsing street level imagery. *ACM Trans. Graph.*, 29:96:1–96:8, July 2010.

[14] J. Kopf, M. Uyttendaele, O. Deussen, and M. F. Cohen. Capturing and viewing gigapixel images. *ACM Trans. on Graphics*, 26, 2007.

[15] M. Levoy and P. Hanrahan. Light field rendering. Proc. SIGGRAPH, pages 31–42, 1996.

[16] W. Y. Lin, S. Liu, Y. Matsushita, T. T. Ng, and L. F. Cheong. Smoothly varying affine stitching. In *Proc. CVPR*, pages 345–352, 2011.

[17] L. McMillan and G. Bishop. Plenoptic modeling: an image-based rendering system. In *Proc. SIGGRAPH*, pages 39–46, 1995.

[18] Y. Nomura, L. Zhang, and S. Nayar. Scene Collages and Flexible Camera Arrays. In *Proc. of Euro. Symp. on Rendering*, Jun 2007.

[19] S. Peleg, B. Rousso, A. Rav-Acha, and A. Zomet. Mosaicing on adaptive manifolds. *PAMI*, 22:1144–1154, 2000.

[20] T. Porter and T. Duff. Compositing digital images. *Proc. SIGGRAPH*, 18(3):253–259, 1984.

[21] P. Rademacher and G. Bishop. Multiple-center-of-projection images. Proc. SIGGRAPH, pages 199–206, 1998.

[22] A. Rav-Acha, Y. Pritch, D. Lischinski, and S. Peleg. Dynamosaics: video mosaics with non-chronological time. In *Proc. CVPR*, 2005.

[23] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3D. *Proc. SIGGRAPH*, pages 835–846, 2006.

[24] M. Uyttendaele, A. Eden, and R. Szeliski. Eliminating ghosting and exposure artifacts in image mosaics. In *Proc. CVPR*, pages 509–516, 2001.

[25] Video Results. http://youtu.be/qfEiARm9Dmg.

[26] L. Zelnik-Manor and P. Perona. Automating joiners. In *Symp. on Non-Photorealistic Animation and Rendering (NPAR)*, pages 121–131, 2007.

[27] L. Zelnik-Manor, G. Peters, and P. Perona. Squaring the circle in panoramas. In *Proc. ICCV*, pages 1292 –1299 Vol. 2, 2005.

[28] D. Zorin and A. H. Barr. Correction of geometric perceptual distortions in pictures. In *Proc. SIGGRAPH*, pages 257–264, 1995.