Plenoptic Stitching: A Scalable Method for Reconstructing 3D Interactive Walkthroughs

Daniel G. Aliaga
aliaga@bell-labs.com

Ingrid Carlbom
carlbom@bell-labs.com

Lucent Technologies Bell Laboratories

Abstract

Interactive walkthrough applications require detailed 3D models to give users a sense of immersion in an environment. Traditionally these models are built using computer-aided design tools to define geometry and material properties. But creating detailed models is time-consuming and it is also difficult to reproduce all geometric and photometric subtleties of real-world scenes. Computer vision attempts to alleviate this problem by extracting geometry and photogrammetry from images of the real-world scenes. However, these models are still limited in the amount of detail they recover.

Image-based rendering generates novel views by resampling a set of images of the environment without relying upon an explicit geometric model. Current such techniques limit the size and shape of the environment, and they do not lend themselves to walkthrough applications. In this paper, we define a parameterization of the 4D plenoptic function that is particularly suitable for interactive walkthroughs and define a method for its sampling and reconstructing. Our main contributions are: 1) a parameterization of the 4D plenoptic function that supports walkthrough applications in large, arbitrarily shaped environments; 2) a simple and fast capture process for complex environments; and 3) an automatic algorithm for reconstruction of the plenoptic function.

Keywords: Virtual environments, plenoptic function, image-based rendering, interactive walkthroughs, omnidirectional.

1. INTRODUCTION

Computer graphics applications, such as telepresence, virtual reality, and interactive walkthroughs require detailed 3D models of their environments. Traditionally such environments are created using computer-aided design systems to specify the geometry and material properties. Using a lighting model, the environment can then be rendered from any vantage point. However, conventional modeling techniques are generally very time consuming and still fall short of recreating the detailed geometry and subtle lighting effects found in most real-world scenes. Photographs of real-world scenes can help recover geometric [Debevec96, Faugeras98] and photometric properties [Yu98]. Computer vision attempts to create real-world models by automatically deriving the geometry and photogrammetry from images of real-world objects. These techniques are based on stereo matching, which is often noisy and has trouble reliably matching a sufficient number of features to create detailed models of complex scenes.

ACM SIGGRAPH 2001, 12-17 August 2001, Los Angeles, CA, USA

In contrast, image-based rendering (IBR) creates novel views of an environment directly from a set of existing images. It does so by resampling the images to generate the new view, thus avoiding the need for an explicit geometric model. The 7D plenoptic function, as introduced by Adelson and Bergen [Adelson91], describes the light intensity passing through every viewpoint, in every direction, for all time, and for every wavelength. All existing IBR techniques generate lower-dimensional plenoptic functions from a set of images.

In this paper, we present a method to reconstruct a 4D plenoptic function for an observer moving in open areas within a large, complex environment while restricting camera motion to a plane at eye-height (Figure 1). We acquire images of the environment by moving an omnidirectional video camera along several paths forming an irregular grid. We intersect the recorded image paths and stitch together simple, closed *image loops*. At run-time, we generate arbitrary views inside each image loop using the surrounding images. The observer can also move freely from one image loop to the next. Thus, if we tile an environment with image loops, we can capture and reconstruct an environment of arbitrary size and shape using an approximately constant-size memory footprint. The benefits of our approach are:

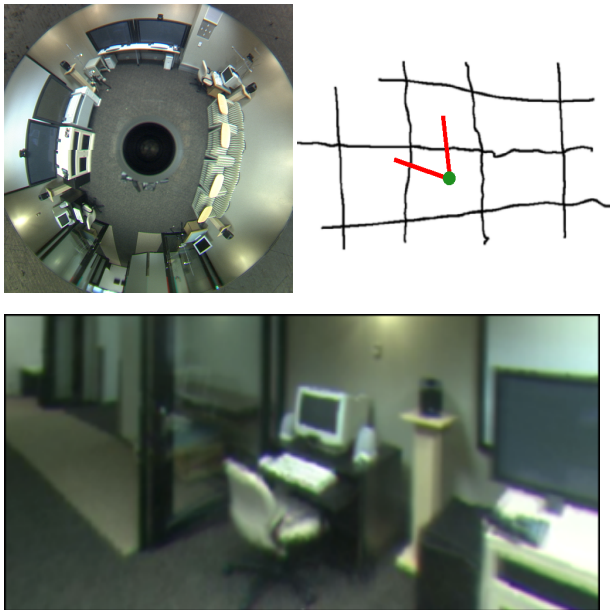

Figure 1. Plenoptic Stitching*. We capture omnidirectional images (upper-left) along several paths through an environment (upper-right). Then, we automatically intersect the paths to create image loops. For a virtual observer (red view frustum) within any image loop, we interactively reconstruct views of the surrounding environment (lower image).*

- **Ease of capture**: We capture complex environments in a manner of minutes. The environments illustrated in this paper were each captured in about 20 minutes.
- **Automated processing**: All processing is automatic, except for the camera pose estimation that requires very minimal user intervention at initialization.
- **Scalability**: Our method can scale to large environments. The reconstruction and interactive display algorithm require only a well-defined local subset of the captured data.
- **Support for arbitrarily shaped environments:** Since the image loops may be of arbitrary shape, we can acquire arbitrarily shaped environments, which may include obstacles.

Our experiments show that 1) our parameterization of the 4D plenoptic function is efficient for walkthrough applications; 2) as with other IBR methods, the computation time and memory requirements of each image loop is independent of model complexity; and 3) with suitable preprocessing, we can generate *interactive* walkthroughs for complex environments.

The rest of the paper is organized as follows. We summarize related work in the following section. Section 3 presents an overview of our method. Sections 4 and 5 detail the capture process and the reconstruction, respectively. We present implementation details in Section 6, and results in Section 7. Finally, we conclude and present future work in the last section.

2. BACKGROUND AND PREVIOUS WORK

The goal of image-based rendering techniques is to reconstruct a continuous representation of the plenoptic function from a set of discrete image samples [Max95, McMillan95]. In practice, all techniques create a subset of the complete, 7D plenoptic function. First by restricting the problem to static scenes captured at discrete wavelengths (e.g. red, green, blue), we can reduce the 7D plenoptic function to 5D.

McMillan and Bishop [McMillan95] use images augmented with depth values to reconstruct the 5D plenoptic function. They formulate an efficient image warping operation that uses a reference image to create images for a small nearby viewing area. For real-world environments, they compute depth by establishing feature correspondences between two cylindrical projections of the environment captured with a small baseline. Expanding to larger environments entails sampling many images from closely-spaced known viewpoints. Other examples of 5D plenoptic functions can be found in [Chen93] and [Kang96].

For unobstructed spaces, we can further reduce the plenoptic function to 4D. Either the scene or the viewpoint is roughly constrained to a box. The Lightfield [Levoy96] and the Lumigraph [Gortler96] capture a large number of images, from known positions, and create a 4D database of light rays. To render the scene from a new viewpoint, they index recorded light rays.

Concentric Mosaics [Shum99] capture an inside-looking-out 3D plenoptic function by mechanically constraining camera motion to a planar concentric circle. Subsequently, they reconstruct novel images from viewpoints restricted to within the circle and with horizontal-only parallax. Takahashi *et al*. [Takahashi00] create another 3D plenoptic function using omnidirectional images captured from a vehicle moving in a straight line, using a global-positioning-system to obtain camera pose.

Finally, if we fix the viewpoint and allow only the viewing direction and the zoom factor to change, we get a 2D plenoptic function. There are many examples in the literature of both cylindrical and spherical panoramas stitched together from multiple images [e.g., Chen95, Szeliski96, Szeliski97]. Another variation is to capture several video streams and show the observer the image captured from the location closest to the current virtual viewpoint [Taylor00]. In addition, there are many other IBR methods for special effects, such as multi-perspective images [Rademacher98].

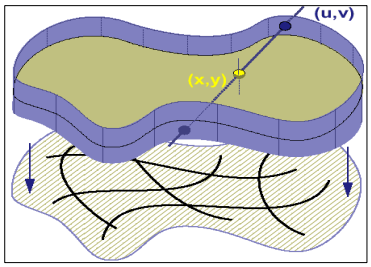

Figure 2. Walkthrough Parameterization. *We parameterize light rays by their intersections (x, y) with the observer plane and (u, v) with the surrounding ruled surface.*

None of the above techniques lend themselves to interactive walkthroughs of large, complex environments. In 2D and 3D plenoptic modeling the viewpoint is severely restricted. With the Lumigraph and Lightfield it would be possible to stitch together large models, but these would be restricted to regular areas and it would be very time-consuming. Finally, the full 5D plenoptic function could, in theory, reconstruct large, complex environments. However, the 5D representation requires the recovery of depth, which is difficult to do accurately and robustly for complex scenes.

3. OVERVIEW OF PLENOPTIC STITCHING

3.1 Walkthrough Parameterization

Our goal is to derive a parameterization of the 4D plenoptic function suitable for walkthrough applications in unobstructed space. One possibility is to parameterize all potential light rays by their intersection with two perpendicular planes: an arbitrarily shaped horizontal plane that represents all viewer positions and a ruled surface, created by sweeping a vertical line around the perimeter of the open space that represents the viewing area.

We parameterize light rays by their intersection with the observer plane (x, y) and with the ruled surface (u, v) creating a 4D plenoptic function (x, y, u, v), as shown in Figure 2. Since we assume that user gaze is kept approximately horizontal, we limit the vertical field-of-view and ignore viewing directions straight-up and straight-down. In order to reconstruct a continuous representation of the 4D plenoptic function, we should ideally sample the entire open space densely.

However, it is not practical to sample the observer plane densely. Thus, we sample the observer plane using an irregular grid of omnidirectional image sequences. (Each image sequence is a 3D plenoptic function.) The grid is adapted to the size, shape, and complexity of the environment. We intersect the paths of the recorded image sequences and form image loops in the observer plane (Figure 2). For each closed loop, we exploit the image loop coherence and warp pixels from the loop boundary to reconstruct views for arbitrary viewpoints inside the loop. The reconstruction also provides a smooth visual transition as the observer moves from one loop to the next.

Our approach can be regarded as a generalization of stitching adjacent plenoptic functions. We relax the restrictions of the

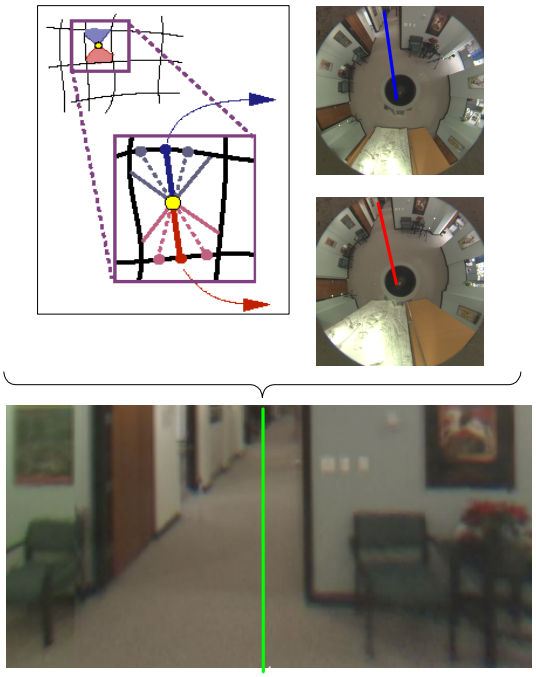

Figure 3. Reconstruction*. For a viewpoint inside an image loop (yellow), we draw several line segments through the viewpoint that intersect the surrounding loop. At each intersection, we extract from the omnidirectional images the radial line of pixels that correspond to the viewing direction. The final reconstructed image is created by warping and combining the extracted pixels.*

camera position to parallel planes (e.g. Lightfield [Levoy95]) or to concentric circles (e.g. Concentric Mosaics [Shum99]) by using an omnidirectional camera pose estimation algorithm and allow an arbitrarily-shaped open space to be captured. Each image loop is analogous to a Lightfield/Lumigraph or a Concentric Mosaic for an observer constrained to an eye-level plane. Thus we call our approach *Plenoptic Stitching*.

3.2 Reconstruction

Given a set of image loops, the goal of our reconstruction algorithm is to create novel planar views of the environment from arbitrary viewpoints inside a loop. We reconstruct the new view (Figure 3) by combining pixels from the omnidirectional images contained in the forward-looking view frustum (blue) with pixels in the omnidirectional images contained in the reverse-looking view frustum (red).

We construct the new image column-by-column. In Figure 3, we highlight the reconstruction process for the middle column of a sample image (green). The line segment that corresponds to the viewing direction of the middle column intersects the surrounding image loop in at least two places. For non-convex loops, there might be more than two intersections. In such cases, we use the two intersections closest to the observer.

Omnidirectional images captured at these intersection points sample the environment outside the loop in the same viewing direction but from two different centers-of-projection (COP). Our omnidirectional camera samples a column at a particular viewing direction using a radial line of pixels. Aside from vertical disocclusions, the environment features sampled by the radial line pair differ only by a radial displacement. If the viewpoint is co-located with one of the COPs, then the new column is a planar re-projection of that radial line. But, in order to reconstruct the column for a viewpoint anywhere along the viewing direction, we need to establish a mapping between the radial lines in the two omnidirectional images. We use this mapping to warp the two radial lines to a planar re-projection for the current viewpoint and blend them together.

4. CAPTURE

4.1 Omnidirectional Capture

We built a camera capture system from off-the-shelf components. The camera is placed on a motorized cart together with a battery, computer, frame grabber, and fast disk system to store the captured images (Figure 4). The setup is completely standalone and radio-controlled, which makes the capture process very simple while also preventing the operator and cables from appearing in the omnidirectional images.

Our walkthrough applications require a 360-degree horizontal field-of-view (FOV) and a large vertical FOV. There are several commercially available omnidirectional cameras, each with different advantages and disadvantages. Two examples are the multi-camera design by Nalwa [Nalwa96] and the paraboloidal catadioptric design by Nayar [Nayar97]. We choose the latter because it has a larger vertical FOV.

This camera uses a convex paraboloidal mirror (i.e. the parabola's focal point is behind the mirror) with an orthographic projection. The full hemisphere of the FOV (360 by 180 degrees) in front of the mirror is reflected onto an image plane that is parallel to and lies in front of the mirror. Since the horizon is located at the mirror border, the vertical FOV spans 0 to -90 degrees. Each omnidirectional image has a single COP, yielding simple transformations to obtain planar re-projections.

4.2 Camera Pose Estimation

To compute camera pose, we developed a camera calibration scheme and a beacon-based pose estimation algorithm for omnidirectional cameras. The details can be found in [Aliaga01]. In two corners of the trackable region of an environment, we place a post equipped with small bright light bulbs and measure the distance between the posts. The calibrated camera is kept at a fixed height and approximately parallel to the ground plane. Before recording, the user initializes pose estimation by identifying the projections of the light bulbs in the first captured image. Then, as the camera moves, the algorithm automatically tracks the light bulbs and, using triangulation, derives the camera position (x, y) and orientation (w) with an average pose error of 0.5% in a typical room-size environment (e.g. 15 ft in diameter).

Figure 4. Omnidirectional Camera Setup*. The motorized cart contains the camera, frame grabber, RAID disks, and battery.*

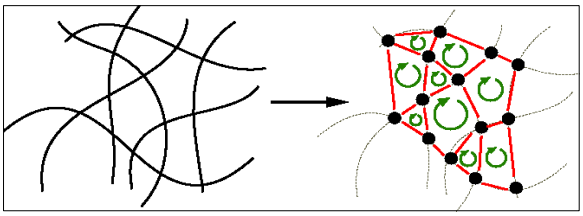

Figure 5. Image Loops*. Given several paths, we intersect them to create image loops.*

Since we know the camera is moving along a generally smooth path at an approximately constant speed and capture frame-rate, we compensate for noise in the pose estimation by fitting B-splines to the camera positions along the recorded paths. Then, we compute local average translation speeds and re-project the camera positions onto the spline curve.

4.3 Image Loops

Once the images are captured over a grid in the environment, we create the image loops. A graph represents the grid with path intersections as vertices and path segments between two intersections as edges. We recursively traverse the graph and determine all image loops (or cycles). See Figure 5.

To compute path intersections, we use oriented bounding box (OBB) trees [Gottschalk96]. A path consists of a sequence of images that are approximately evenly spaced along the path. We derive the OBB tree for each path by dividing it into in two segments, surrounding each segment with an oriented bounding box, and continuing the process until a small number of images remain in each node. To find the path intersections, we intersect path OBB trees top down and then find the intersecting line-segments in the leaf nodes by enumeration.

5. RECONSTRUCTION

Recall that a new view from an arbitrary viewpoint inside an image loop consists of a column-by-column reconstruction of pixels from the omnidirectional images contained in the forward-looking view frustum with pixels in the omnidirectional images in the reverse-looking view frustum. Assuming the viewing direction for a column intersects the COP of two omnidirectional images on opposite sides of the image loop, then that viewing direction identifies two radial lines in the images that represent the same viewing direction but from different COPs.

Figure 6 shows a cross-plot of features in the two corresponding radial lines. The horizontal and vertical axes of the graph represent the parametric position of corresponding image features.

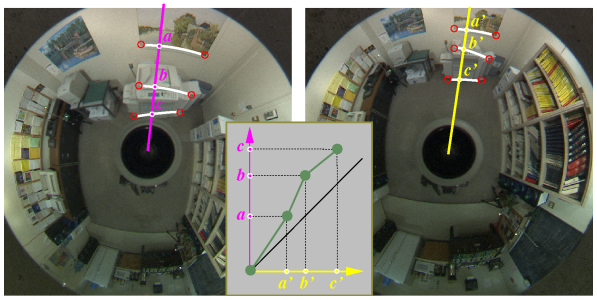

Figure 6. Mapping Function. *We show a cross-plot of corresponding features sampled by a pair of radial lines. The horizontal and vertical axes of the graph represent the parametric position of the intersection between each radial line and each edge of the local feature triangulation.*

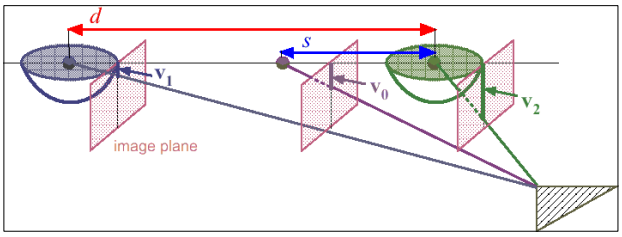

Figure 7. Interpolating Feature Positions. *For each reconstructed column, we compute the interpolated feature position v_0 for a planar re-projection of the environment.*

We warp corresponding segments of each radial line to the column in the reconstructed image.

We need to consider the geometry of our omnidirectional camera when warping the radial line segments. Figure 7 shows a profile of an image loop that depicts the paraboloidal mirrors at opposite ends of the loop, the image column to reconstruct, the viewpoint of a virtual observer, and an example feature outside the loop. For the omnidirectional images, we show the projected positions v_1 and v_2 of the sample feature in the image column. We compute the image column position v_0 of the feature for the virtual observer by using similar triangles. The following expression computes v_0 given v_1, v_2, and the parametric position s as a function of the distance d between the COPs of the omnidirectional images along the viewing direction:

$$v_0 = v_2 \Big/ \left(s\left(\frac{v_2}{v_1} - 1\right) + 1\right) \qquad (1).$$

Using a forward mapping, occlusion compatible ordering of the pixels of a radial line [McMillan95] eliminates the need to explicitly address visibility ordering. Since pixels from the image behind the viewpoint are generally stretched during the warp, we draw pixels using fixed size splats. In our case, splats are actually short line segments. To fill-in vertical disocclusions, we simply use longer than expected splats. This is equivalent to filling the gap with the last sample of the background object. We reduce the remaining reconstruction artifacts by filtering the final image using a 3x3 Gaussian convolution kernel.

Generally, the viewing direction will not intersect the image loop exactly at the center of projection of an omnidirectional image. In this case, we extract two radial lines parallel to the view direction, blend them, and proceed with the reconstruction as previously described.

Since there might be a large displacement between the two omnidirectional images on the opposite sides of the loop, it is difficult to reliably identify corresponding features in the radial lines from only two lines. Instead we rely upon the temporal coherence of the entire image loop to identify the required features. We choose an arbitrary omnidirectional image from the loop and use image processing techniques to identify a set of features (e.g. points, corners) [Shi94]. We then use the Kanade, Lucas, and Tomasi [Tomasi91] algorithm to track features from the original image around the loop, keeping only those features that successfully track all the way around and have similar beginning and ending image positions.

To obtain feature positions for a specific radial line, we triangulate the tracked features in each image and use the intersection points between the triangulated edges and the radial line as the feature points for the mapping.

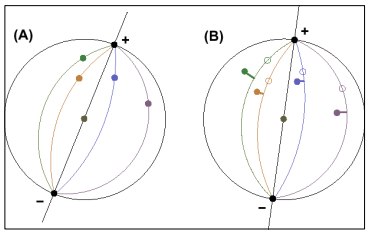

Figure 8. Rotation Correction*. The features of image A move along the arcs of the circles defined by the positive epipole, negative epipole, and the features themselves. On image B, the features should be on the same set of circles. The sum of the squared distances of their deviation from the expected trajectory is the error term used during the optimization.*

In a curved-mirror omnidirectional camera, straight lines project to curves. For a parabolic mirror, these curves are arcs of circles [Geyer98]. Thus, for each image, we compute a Delaunay triangulation of the tracked features and replace the straight (triangle) edges with arcs. Each arc has the property that it passes through the two vertices of the original edge and it intersects the mirror border at two points 180-degrees apart.

To obtain an expression for the circle containing the arc, we start by transforming the two vertices of the original triangulation edge to a canonical space. In this space, the captured image has a radius equal to one and the vertices are rotated so that one of them lies along the x-axis. The vertices are now represented by (x_0, 0) and (x_1, y_1). The following expressions compute the circle at (c_x, c_y) of radius r that passes through the two vertices and intersects the mirror border 180-degrees apart:

$$c_x = \frac{x_0^2 - 1}{2x_0} \quad c_y = \frac{x_1 - x_0 + x_0(x_1^2 + y_1^2) - x_0^2 x_1}{2x_0 y_1} \quad r = \sqrt{1 + c_x^2 + c_y^2} \qquad (2).$$

We intersect the arc-edge with a radial line by solving a quadratic equation. We represent the radial line by a ray from the origin through (r_x, r_y) and to the arc-edge on the border of the circle (c_x, c_y, r). The point of intersection is (tr_x, tr_y) where t satisfies:

$$(r_x{}^2 + r_y{}^2)t^2 - 2(c_x r_x + c_y r_y)t + (c_x^2 + c_y^2 - r^2) = 0 \qquad (3).$$

5.1 Optimization

Under ideal conditions and ignoring vertical disocclusions, the surface samples visible in corresponding radial lines differ only by a radial displacement. As described earlier, the radial line mapping should be able to recover this displacement. In practice, however, both the feature tracking and the camera pose estimation introduce errors into this mapping. In addition we introduce aliasing, since we sample at a discrete set of viewing positions for a discrete set of viewing directions. We perform two types of optimization to compensate for these artifacts.

5.1.1 Rotational Correction

The first type of optimization accounts for the incorrect pairing of radial lines. This is due to camera pose error, in particular, rotation error that dominates any translation error.

Figure 8 illustrates two omnidirectional images intersected by one viewing direction. If we move the viewpoint along the viewing direction from the COP of image A to that of image B, the (instantaneous) positive and negative epipoles lie on that line and at the border of the mirror. Moreover, epipolar geometry tells us that a feature moves along the arc of a circle that intersects the positive epipole, the negative epipole, and the feature itself. If image A and image B were rotationally aligned, corresponding features in the two images move along the same set of circles.

When the images are not aligned, the error can be represented by the sum of the squared distances between the features and their expected trajectories. Expression (2) gives the trajectory for a feature in image A, using the feature and an epipole to define the arc-edge. The distance (in pixels) of a corresponding feature in image B from the perimeter of the computed circle is the error for that feature. To reduce these errors, we search for rotations that better align images in a least-squared sense. For each image, we obtain an array of rotation corrections, one for every image pair.

For each reconstructed column, we add this rotation correction to the omnidirectional image's orientation computed by the pose estimation. We assume the images of neighboring path positions have a similar rotation. Thus, after adding the rotation correction, we perform a smoothing operation over the resulting values.

5.1.2 Column Correlation

The second type of optimization compensates for inaccuracies in the recovered radial displacement between a radial line pair. In the absence of errors, the two warped columns computed from the mapping functions should be vertically aligned. However, feature drifting, lack of sufficient features, and inaccurate distance estimation between the images cause misalignment.

To improve alignment, we scale one column (keeping the other unchanged) prior to blending the columns together. This is equivalent to changing the assumed distance between the COPs. To find the scale factor, we correlate low-pass filtered and downsampled versions of the radial lines [Faugeras93].

6. IMPLEMENTATION

6.1 System Overview

We developed custom C++ software on a 667 MHZ Pentium III PC and on a SGI Onyx2 with 4 250 MHz MIPS processors. The PC captures images to disk, computes camera pose, creates image loops, and tracks features. The SGI creates the column mappings, computes the optimizations, and interactively reconstructs images. The SGI software uses all four processors, with each processor reconstructing ¼ of the columns.

The capture system uses a progressive-scan JVC KY-F70 1360x1024 3-CCD color video camera and a Matrox frame grabber card. We control the recording software remotely using a wireless Ethernet connection (i.e. a WaveLan card) and a small laptop. The motors of the cart are controlled via a radio control unit (from a hobby store). We capture and transfer-to-disk frames at an average rate of 6.5 fps. The motorized cart moves at a speed of 0.2 m/sec, simulating a person walking at about one m/sec and recording video at roughly 30 Hz.

6.2 Reconstruction Acceleration

In order to obtain real-time performance, we divide the reconstruction into a preprocessing and a runtime phase (Figure 9). In the preprocessing step, we track the features for the column mappings, optimize image rotation, and create a data structure to quickly access radial lines and their mapping functions required for the reconstruction. From each radial line in each omnidirectional image in an image loop, we generate potential viewing directions to the opposite side of the image loop and store the derived radial line mapping in the data structure.

At runtime, using the observer position and viewing direction, we find the nearest preprocessed viewing directions and use these to

```
Capture
- Capture multiple image sequences
- Compute camera pose and create image loops
Preprocessing
- Track features around each loop
- For each radial line pair, compute rotation
  corrections and mapping functions
- Compress captured images
Interactive Display
- Use the rotation corrections and camera
  pose information to extract radial line
  pairs and mapping functions for observer
  viewpoint
- Use mapping functions to warp corresponding
  radial lines from the omnidirectional
  images to a planar re-projection
- Compute column correlation optimization
```

Figure 9. Plenoptic Stitching Pipeline.

index the data structure containing the radial lines and mapping functions. In addition, we accelerate reconstruction by (optionally) using fewer mapping functions than columns in the image, effectively warping using lower-resolution information.

6.3 Caches and Compression

In order to maintain an approximately constant-size memory footprint and to reduce overall storage, we compress the source images and computed data structures. At run-time, three caches dynamically load the images and data. For the source images, we use a modified JPEG compression [Wallace91], and for the data structures we use Lempel-Ziv compression [Ziv77].

In order to efficiently use JPEG compression, we re-project the captured images to cylindrical projections using bilinear interpolation. In this projection the radial lines become columns. We divide each path into segments and each cylindrical projection into groups of columns. For each segment, we create a single bitstream. The JPEG comment field stores offsets to every group of columns of every image in the segment. Thus at runtime we can directly extract and decompress the desired column group. Two caches are used for decompression: one cache stores the compressed bitstreams already loaded from disk, the other stores decompressed groups of columns. Both caches use a least-recently-used (LRU) replacement policy.

In a similar fashion, we divide the remaining data structures of each path into segments and compress each segment using Lempel-Ziv coding. At run-time, we dynamically load and decompress each segment into a LRU cache.

7. RESULTS AND OBSERVATIONS

We have captured four environments using plenoptic stitching: Copier Room, Lobby, Multimedia, and Showcase. Table 1 lists several statistics about the acquisition of each environment, as well as the resulting total database sizes and preprocessing times. We do not include time to transport equipment to each location.

As with other IBR methods, interactive reconstruction is accomplished in time proportional to image size. However, reconstruction time does vary slightly depending on the number of features per column. Reconstructions of 320x160 pixels and 640x320 pixels take between 5 to 10 frames per second. Our pixel splat size is usually 1x4 pixels. We note that the capture time is very fast; 25 minutes is the maximum for these environments. Storage requirements are comparable to other IBR methods.

Figure 10 shows the reconstruction quality for several viewpoints near the middle of the image loops in our sample environments, where proper reconstruction is most difficult. As a viewpoint approaches a captured path, reconstruction quality increases to that of the captured images. Figure 11 compares the quality of our reconstructions to the quality of our omnidirectional camera placed at the same observer viewpoint. The upper image in Figure 11 shows a reconstructed image for a viewpoint near the middle of an image loop (worst case) in the Multimedia environment. The bottom image shows an image captured from approximately the same location and re-projected to a planar projection.

We found that the optimizations much improve our reconstruction. We illustrate this in Figure 12, which shows one reconstructed image with and one without optimization. To further illustrate the significance of the optimizations, we recorded the average per-frame corrections for a path in each environment. Figure 13a illustrates the absolute values of the rotation corrections (in degrees), and Figure 13b the average linear-scale values.

The remaining blurriness in our reconstructed images is a consequence of several factors. First, our omnidirectional camera captures more pixels near the horizon than towards the lower part of the vertical FOV, resulting in lower image quality in the lower parts of the reconstructed images. Second, our multiple resampling and interpolation operations yield artifacts. Third, our mapping functions are dependent on feature tracking which in

Figure 10. Example Reconstructions*. We show images reconstructed for viewpoints near the middle of image loops (left column: Multimedia and Copier Room environments, middle column: Showcase environment, right column: Lobby environment).*

Figure 11. Reconstruction Comparison*. The top image is reconstructed by our method for a viewpoint near the middle of an image loop. The bottom image is a planar re-projection of an omnidirectional image captured (approximately) from the same viewpoint.*

turn is dependent on the presence of good features in the environment. Lacking these features, blending of incorrectly warped pixels yields blurriness and ghosting.

For a given image loop, we have control over some parameters that affect reconstruction quality. These parameters include the capture frame rate, the image resolution, and the ratio between the approximate diameter of the loop and the distances to the objects. For a given camera, image resolution is fixed. Increasing the capture frame rate improves the sampling of light rays along each path. While this is beneficial for viewpoints near a path, we are more concerned with the reconstruction quality for viewpoints near the middle of an image loop. Since objects at a distance move little in the image plane, it is easier to reconstruct the environment for a small loop sampling distant objects than for a larger loop sampling closer objects. One approach would be to determine, through experimentation, what loop sizes and object distances we require to achieve a desired image quality.

8. CONCLUSIONS AND FUTURE WORK

We have introduced plenoptic stitching, a novel approach to creating interactive walkthroughs of 3D virtual environments for an observer moving through unobstructed space in a plane. We introduce a novel parameterization of a 4D plenoptic function tailored to interactive walkthroughs. A fast and easy capture process samples the environment and then automatically reconstructs the 4D plenoptic function. Our algorithm captures environments of arbitrary size and shape without affecting the local reconstruction quality or storage.

In order to maximize image quality, an interesting avenue of future work is to automatically guide the capture process so as to maintain a certain ratio between the loop size and the distance to the objects in the environment. For example, given a coarse model of an environment, record an optimal path grid [Roberts97].

In order to capture more compelling and multi-room environments, auditoriums, and large exterior spaces, we must extend the camera pose estimation. Our current algorithm limits

Figure 12. Optimizations*. The left image pair shows a reconstructed view of the Copier Room environment with optimizations enabled. The right image pair shows the same view with optimizations disabled.*

us to environments of approximately 120 square feet. We are pursuing extensions to our camera pose estimation algorithm for multi-room environments. Other possible camera pose estimation techniques include the UNC Ceiling Tracker [Ward92].

We would like to improve the quality of omnidirectional capture. In particular, we are interested in multiple-camera configurations that acquire higher resolution images and nearly full spherical projections. For example, one method to increase the FOV is to place two of our hemispherical cameras back-to-back.

Finally, we can exploit the mapping functions to improve compression. Each pair of radial lines exhibits a large amount of coherence. Similar to motion estimation in MPEG compression [LeGall91], we can use the mapping function to predict one radial line from the other, in addition to exploiting the image-to-image coherence around the loop.

Acknowledgments

We thank Sid Ahuja, Multimedia Communications Research VP at Bell Labs, for supporting this research. In addition, we thank Bob Holt, Gopal Pingali, and Nicolas Tsingos for their technical help; Agata Opalach for her assistance with the diagrams; and Tom Funkhouser, Steve Fortune, and Bob Kubli for their advice.

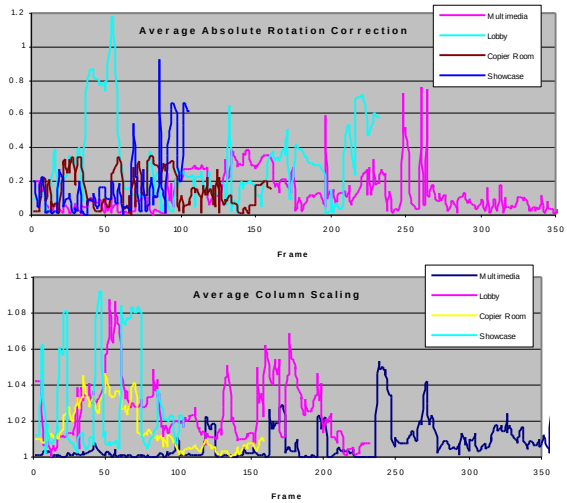

Figure 13. Average Corrections Computed by the Optimizations*. For all environments, we traverse a path and show the average per frame absolute value of the rotation corrections (in degrees) and the average per frame column scale value.*

Name	Setup Time (mins)	Capture Time (mins)	No. of Paths/ Loops	No. of Images	Total Paths (meters)	Raw Images (MB)	Raw Data Structs (MB)	Compressed Images (MB)	Compressed Data Structures (MB)	Total Data (MB)	Prep. Time (hours)
Copier Rm	12	8	4/1	498	15	1494	159	55	59	114	1
Lobby	5	20	7/5	1304	41	3912	586	110	217	327	5
Multimedia	5	15	7/5	1025	32	3075	425	112	158	270	5
Showcase	10	25	8/9	1600	49	4676	528	169	196	365	6

Table 1. Environment Statistics.

References

[Adelson91] Adelson E.H. and Bergen J., In "The Plenoptic Function and the Elements of Early Vision", In *Computational Models of Visual Processing*, MIT Press, Cambridge, MA, 3-20, 1991.

[Aliaga01] Aliaga D., "Accurate Catadioptric Calibration for Real-time Pose Estimation in Room-size Environments", *IEEE International Conference on Computer Vision (ICCV 01)*, July, 2001.

[Chen93] Chen S. E. and Williams L., "View Interpolation for Image Synthesis", *Computer Graphics (SIGGRAPH 93)*, 279-288, 1993.

[Chen95] Chen S. E, "QuickTime VR - An Image-Based Approach to Virtual Environment Navigation", *Computer Graphics (SIGGRAPH 95)*, 29-38, 1995.

[Debevec96] Debevec, P.E., Taylor C.J., and Malik, J., "Modeling and Rendering Architecture from Photographs", *Computer Graphics (SIGGRAPH 96)*, 11-20, 1996.

[Faugeras93] Faugeras O.D**,** *Three-Dimensional Computer Vision: A Geometric Viewpoint*, MIT Press, Cambridge, MA, 1993.

[Faugeras98] Faugeras O.D., Laveau S., Robert L., Czurka G., and Zeller C., "3D Reconstruction of Urban Scenes from Sequences of Images*"*, *Computer Vision and Image Understanding*, Vol. 69(3), 292-309, 1998.

[Geyer98] C. Geyer and K. Daniliidis, "Catadioptric Camera Calibration", *IEEE International Conference on Computer Vision (ICCV 98)*, pp. 398-404, 1998.

[Gortler96] Gortler S., Grzeszczuk R., Szeliski R., and Cohen M., "The Lumigraph", *Computer Graphics (SIGGRAPH 96)*, 43-54, 1996.

[Gottschalk96], Gottschalk S., Lin M., and Manocha D., "OBBTree: A Hierarchical Structure for Rapid Interference Detection", *Computer Graphics (SIGGRAPH 96)*, 171-180 (1996).

[Kang96] Kang S.B. and Szeliski R., "3D Scene Data Recovery Using Omnidirectional Baseline Stereo", *IEEE Computer Vision and Pattern Recognition (CVPR 96)*, 364-370, 1996.

[Levoy96] Levoy M. and Hanrahan P., "Light Field Rendering", *Computer Graphics (SIGGRAPH 96)*, 31-42, 1996.

[LeGall91] Le Gall D., "MPEG: A Video Compression Standard for Multimedia Applications", *Communications of the ACM (CACM)*, Vol. 34(4), 46-58, 1991.

[Max95] Max N. and Ohsaki K., "Rendering Trees from Precomputed Z-Buffer Views", *Rendering Techniques '95*: *Proceedings of the 6th Eurographics Workshop on Rendering*, 45-54, 1995.

[McMillan95] McMillan L. and Bishop G., "Plenoptic Modeling: An Image-Based Rendering System", *Computer Graphics (SIGGRAPH 95)*, 39-46, 1995.

[Nalwa96] Nalwa V.S., *A True Omnidirectional Viewer*, Technical Report, Bell Laboratories, Holmdel, NJ, 1996.

[Nayar97] S. Nayar, "Catadioptric Omnidirectional Camera", *IEEE Computer Vision and Pattern Recognition (CVPR 97)*, 482-488, 1997.

[Rademacher98] Rademacher P. and Bishop, G., "Multiple-Center-of-Projection Images", *Computer Graphics (SIGGRAPH 99)*, 199-206, 1999.

[Roberts97] Roberts D.R. and Marshall A.D., "A Review of Viewpoint Planning", Technical Report 97008, University of Wales, College of Cardiff, Department of Computer Science, 1997.

[Shi94] Shi J. and Tomasi C., "Good Features to Track", *IEEE Computer Vision and Pattern Recognition (CVPR 94)*, 593-600, 1994.

[Shum99] Shum H. and He L., "Rendering with Concentric Mosaics", *Computer Graphics (SIGGRAPH 99)*, 299-306, 1999.

[Szeliski96] Szeliski R., "Video mosaics for virtual environments", *IEEE Computer Graphics and Applications*, 22-30, 1996.

[Szeliski97] Szeliski R. and Shum H., "Creating full view panoramic image mosaics and texture-mapped models", *Computer Graphics (SIGGRAPH 97)*, 251-258, 1997.

[Takahasi00] Takahashi T., Kawasaki H., Ikeuchi K., and Sakauchi M., "Arbitrary View Position and Direction Rendering for Large-Scale Scenes**",** *IEEE Computer Vision and Pattern Recognition (CVPR 00)*, 296-303, 2000.

[Taylor00] Taylor C., "Video Plus**",** *IEEE Workshop on Omnidirectional Vision*, 3-10, 2000.

[Tomasi91] Tomasi C. and Kanade T., "Detection and Tracking of Point Features**",** Carnegie Mellon University Technical Report CMU-CS-91-132, 1991.

[Wallace91] Wallace G., "The JPEG Still Picture Compression Standard", *Communications of the ACM (CACM)*, Vol. 34(4), 30-44, 1991.

[Ward92] Ward M., Azuma R., Bennett R., Gottschalk S., and Fuchs H., "A Demonstrated Optical Tracker with Scalable Work Area for Head-Mounted Display Systems.", *ACM Symposium on Interactive 3D Graphics (I3D 92)*, 43-52, 1992.

[Yu98] Yu Y. and Malik J., "Recovering photometric properties of architectural scenes from photographs", *Computer Graphics (SIGGRAPH 96)*, 207-218, 1996.

[Ziv77] Ziv J. and Lempel A., "A universal algorithm for sequential data compression", *IEEE Transactions on Information Theory*, IT-23, 337-343, 1977.