Sea of Images

Daniel G. Aliaga
Lucent Bell Labs
aliaga@bell-labs.com

Thomas Funkhouser
Princeton University
funk@cs.princeton.edu

Dimah Yanovsky
Harvard University
yanovsky@fas.harvard.edu

Ingrid Carlbom
Lucent Bell Labs
carlbom@bell-labs.com

ABSTRACT

A long-standing research problem in computer graphics is to reproduce the visual experience of walking through a large photorealistic environment interactively. On one hand, traditional geometry-based rendering systems fall short of simulating the visual realism of a complex environment. On the other hand, image-based rendering systems have to date been unable to capture and store a sampled representation of a large environment with complex lighting and visibility effects.

In this paper, we present a "Sea of Images," a practical approach to dense sampling, storage, and reconstruction of the plenoptic function in large, complex indoor environments. We use a motorized cart to capture omnidirectional images every few inches on a eye-height plane throughout an environment. The captured images are compressed and stored in a multiresolution hierarchy suitable for real-time prefetching during an interactive walkthrough. Later, novel images are reconstructed for a simulated observer by resampling nearby captured images.

Our system acquires 15,254 images over 1,050 square feet at an average image spacing of 1.5 inches. The average capture and processing time is 7 hours. We demonstrate realistic walkthroughs of real-world environments reproducing specular reflections and occlusion effects while rendering 15-25 frames per second.

CR Categories: I.3.3 [Picture and Image Generation]: Display and viewing algorithms. I.3.7 [Three-dimensional Graphics and Realism]: Virtual Reality.

Keywords: image-based rendering, capture, reconstruction, interactive, walkthrough.

1 INTRODUCTION

Creating an interactive walkthrough of a complex real-world environment remains one of the most challenging problems in computer graphics. While many researchers have tackled parts of the problem, there exist to date no system that can reproduce the photorealistic richness of a large, real-world environment at interactive rates. For example, a virtual visit to the Louvre or Versailles must reproduce the exquisite detail of the paintings and sculptures while at the same time convey the grandeur of the former royal residences. And, as in a real museum visit, the user must be able to walk anywhere, even up close to an interesting work of art.

Image-based rendering (IBR) achieves photorealism by capturing and resampling a set of images. An IBR system usually takes as input photographs of a static scene, and constructs a sample-based representation of the plenoptic function [1]. This function, $P(x, y, z, \phi, \theta, t, \lambda)$, describes the radiance leaving or arriving at any point (x, y, z) from any direction (ϕ, θ) with any wavelength λ at any time t. The plenoptic representation can be quickly resampled to render photorealistic images for novel viewpoints without constructing a detailed 3D model or simulating global illumination,

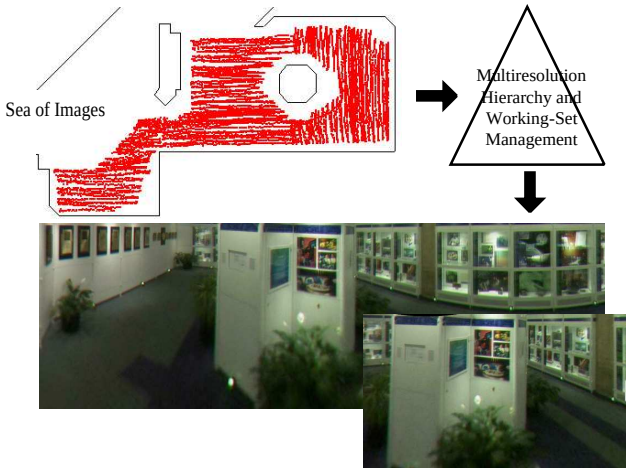

Figure 1: Sea of Images. We capture a dense Sea of Images through a large environment, store them in a multiresolution hierarchy, and generate real-time reconstructions of novel images in an interactive walkthrough system.

and the rendering time for novel images can be independent of a scene's geometric complexity. However, current IBR methods are only able to represent either small scenes (e.g., a statuette) or diffuse environments with low geometric complexity (e.g., a room or a hallway). Our goal is to create an IBR walkthrough system that supports an interactive experience for large and complex real-world scenes.

We create interactive walkthroughs using a "Sea of Images" (SOI) – a collection of images every couple inches throughout a large environment. In our case, we acquire omnidirectional images on an eye-height plane throughout the environment (Figure 1). This representation provides a densely sampled 4D approximation to the plenoptic function parameterized by camera position (x, y) and incoming ray direction (ϕ, θ). We capture a SOI by moving a catadioptric video camera mounted on a motorized cart back and forth in a zigzag pattern through a static environment. We compress the acquired data in a multiresolution hierarchy so that it can be accessed efficiently for continuous sequences of viewpoints. We use time-critical algorithms to prefetch relevant image data and feature-based morphing methods to reconstruct novel images during interactive walkthroughs.

As compared to previous IBR methods for interior environments, our SOI approach replaces the difficult computer vision problems of 3D reconstruction and surface reflectance modeling with the easier problems of motorized cart navigation, data compression, and working set management. Rather than using sophisticated planning and reconstruction algorithms to acquire directly a minimal representation of the plenoptic function, we capture a highly redundant data set and then compress it into a representation that enables real-time working set management for walkthroughs (Figure 2).

The advantages of this approach are four-fold. (1) It enables accurate image reconstructions for novel views in environments with

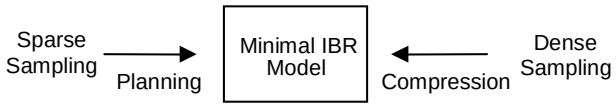

Figure 2: Acquiring Image Samples. Instead of sophisticated planning and vision algorithms to acquire the minimal sample set, we oversample and use compression and working set management to access samples during a real-time walkthrough.

specular surfaces, a great amount of geometrical detail, and complex visibility changes. (2) It does not require an accurate geometric model to produce novel images over a wide range of viewpoints. (3) It provides a method for image-based modeling of a large, concave environment without sophisticated gaze planning. And, (4) it supports rendering of inside-looking-out images in an IBR interactive walkthrough system. We believe no other IBR approach includes all these features.

2 Related Work

Our "Sea of Images" is related to several previously described IBR methods. In this section, we review the most closely related representations and discuss the advantages of our representation for walkthrough applications.

Movie maps [21] and panoramas [6, 23, 32, 7] sample the plenoptic function in 2D for a small, sparse set of reference viewpoints. While this representation is simple to capture, it grossly undersamples the spatial dimensions of the plenoptic function, and thus it is not able to capture specular highlights or complex occlusion effects common in interior scenes. Also, it requires either constraining the user to view the scene only from reference viewpoints [6, 21, 32, 16] or acquiring accurate geometry to enable image reconstructions for viewpoints far from any captured view [23, 7]. This is difficult for complex environments, even when the environment is modeled using laser range scanners [26]. In contrast, a Sea of Images samples both spatial and angular dimensions densely and interpolates nearby reference images when reconstructing novel views. Thus, it is better able to capture and reproduce subtle lighting and occlusion changes. Furthermore, only approximate geometry or depth information is required for reconstructing novel views since the viewpoints of captured images are usually very close to novel viewpoints typical in a walkthrough.

Light Fields [13, 20] provide a 4D approximation to the plenoptic function parameterized by a point on a "camera plane" (s, t) and a point on an "object plane" (u, v). In contrast, our 4D representation is parameterized by a point on a camera plane (x, y) and a view direction (ϕ, θ). This difference is significant for interior walkthrough applications for three reasons. First, it is much easier to capture a Sea of Images than it is to acquire a Light Field. Special gantries can acquire the (s, t, u, v) Light Fields for small objects, but seems difficult to scale to large, complex environments. In contrast, capturing a (x, y, ϕ, θ) representation simply requires moving an omnidirectional video camera in a single plane. Second, the (x, y, ϕ, θ) parameterization simplifies coverage of a large, concave space. Since each Light Field provides samples for a limited range of novel viewpoints, it is difficult to cover a large complex environment. In contrast, we capture reference images where we expect the user to explore, and then the system can synthesize new images anywhere within the triangulated mesh of reference viewpoints. Finally, our (x, y, ϕ, θ) approach provides a pre-filtered multiresolution representation of light radiance arriving at a point, which is important for walkthroughs where surfaces in the same image can appear at significantly different distances. In contrast, Light Fields with fixed uniform sampling of the (u, v) plane will either be undersampled for close-up viewpoints or oversampled for distant viewpoints.

Surface Light Fields [36] are similar to our representation as they also parameterize the plenoptic function by a point and a direction. However, they describe the radiance leaving a surface point rather than arriving at a camera location. This difference has two significant implications for walkthrough applications. First, the Surface Light Field requires an accurate geometric description of the scene in order to aggregate and store samples on surfaces, where as our approach works well with little or no geometry. Second, and more importantly, the data access patterns of a walkthrough application have much more coherence for nearby viewpoints and incoming ray directions than for nearby surface points and outgoing directions. Hence, the Surface Light Field parameterization may enable greater overall compression of samples on disk (if the surfaces are mainly diffuse), but the Sea of Images approach is better suited for efficient working set management during a walkthrough.

Recent computer vision work has provided methods for reconstructing 3D geometry and capturing textures of environments for interactive walkthroughs. Representative examples include MIT's City Scanning Project [33], Pollefey et al.'s work on 3D reconstruction from hand-held cameras [18, 28], Zisserman et al.'s work on obtaining graphical models from video sequences [37], and Kang et al.'s omnidirectional multibaseline stereo algorithms [17]. This research is largely orthogonal to ours, as we could use their methods to acquire more accurate geometric models to improve the quality of our reconstructed images. However, an important feature of our approach is that it does not require a very accurate 3D model to produce novel images because we capture images at very high density and nearby reference images are available for almost any novel viewpoint. Our approach avoids the difficult problems of 3D scene reconstruction (e.g., dense correspondence, depth estimation) and replaces them with a simpler data management problem, which is addressed in this paper.

Delta Trees [8] and LDI Trees [5] are related to our approach in that they store multiresolution representations of the radiance field. Pre-filtered image data is available for every surface at multiple resolutions (e.g., like mip-maps), and thus the amount of real-time filtering required to reconstruct images from novel viewpoints is significantly reduced. Our SOI approach also stores image data for each surface at multiple resolutions (pre-filtered by the camera according to the location of each captured image). However, our multiresolution representation is viewpoint-centric, rather than object-centric, which greatly improves cache coherence during an interactive walkthrough.

Plenoptic Stitching [3] is most closely related to our work. But, it samples the plenoptic function densely in only one spatial dimension, capturing images in a cross-hatch pattern of lines separated by meters. This results in a 3.5D approximation to the plenoptic function, which does not capture most visibility or lighting changes in the vertical dimension. In addition, the reconstruction algorithm warps samples only along one degree of freedom (radial lines), which generally produce lower quality reconstructions than our system. For instance, pixels are reconstructed by combining samples from a captured image significantly nearer to and another one farther from the sampled environment, thereby combining samples at different resolutions and causing blurring and ghosting in the rendered images. Our system avoids this problem by warping three nearby reference images. Additionally, we use multiresolution data compression and real-time working set management algorithms.

In summary, previous IBR methods can be classified by how many image samples are collected and how much a priori 3D scene geometry and reflectance information is required to produce realistic novel views. Some methods capture images densely, thus requiring little geometric or reflectance information (e.g., [20, 13]). However, so far, they are limited to small scenes and small ranges of novel viewpoints. Other methods sparsely sample the space of possible viewpoints (e.g., [23]). However, they must acquire de-

tailed geometric information to warp images to novel views. Hybrid methods capture images from a semi-dense set of viewpoints and utilize approximate geometric information (e.g., [3, 30]). These methods usually produce good results only for scenes without complex surface reflections or visibility effects.

Our system captures a dense sampling of images over a large area. This allows it to reproduce specular highlights and complex visibility effects during walkthroughs of large environments over a wide range of novel viewpoints, without requiring detailed geometric information.

3 Research Challenges

In this paper, we investigate a dense sampling of the plenoptic function for interactive walkthroughs of interior environments. We capture closely spaced omnidirectional images, compress them, and resample them to form novel views. This approach allows reconstruction of novel views with subtle lighting and visibility effects. However, it requires that we address the following three questions:

- *"How can we obtain a dense set of calibrated images over a large area in a practical manner?"* - We have built a capture system from off-the-shelf components that uses a motorized cart to move an omnidirectional video camera in a zigzag pattern at eye height through a large environment. We have also developed camera calibration and pose estimation algorithms for large, multi-room environments.
- *"How do we compress the massive amounts of captured data?"* - We have developed a multiresolution IBR representation that enables us to exploit coherence in nearby images resulting in significant compression.
- *"How can we access a large out-of-core data set for real-time walkthroughs?"* - We manage the working set through predictive prefetching and caching algorithms that load images from disk based on estimated "benefits" and "costs" as a user moves through an environment interactively.

In the following three sections we investigate answers to these questions.

4 Capture & Calibration

The first problem is the capture and calibration of a dense Sea of Images in a manner both practical and automatic. The method should work reliably in large, multi-room environments without invasive hardware and without per-image manual processing. For example, capturing an image every few inches inside a non-trivial environment (e.g., a small museum) should take no more than an afternoon.

Most previous IBR capture methods rely either upon special-purpose hardware, specific environment content, or upon careful path and gaze planning. Levoy and Hanrahan [20] use a gantry to capture a dense set of images uniformly over a plane. Similarly, Shum and He [30] use special-purpose hardware to capture images on a circle. Large-scale (optical) trackers could be used for image capture but require a significant hardware installation. Teller et al. [33] capture and calibrate thousands of outdoor images spaced by several to tens of meters and depend on initial measurements from a global positioning system. These methods are practical only for small objects, single rooms, or outdoor environments. They seem difficult to extend to a wide range of dense viewpoints in complex interior environments.

Vision-based approaches often depend on structure-from-motion and on scene content [28, 18, 11]. Jung and Taylor [16] describe a vision-based approach augmented with inertial sensors. To compensate for drift inherent in these devices, global features or landmarks must be identified. While this strategy could in principle work for large interior environments, it is very dependent on the content of the environment. Moreover, the results so far are not sufficiently robust to reconstruct detailed geometry and reflectance effects (e.g., visibility changes, specular highlights).

Gaze planning algorithms [29, 31] can be used in larger environments, but require a priori knowledge of the location of the interesting detail and also what is of interest to the user. Unless an accurate model already exists, a system based on these approaches may miss significant image samples that are difficult to capture at a later date.

In our approach, we capture a dense Sea of Images by moving an omnidirectional camera on a motorized cart throughout the environment, continuously capturing image data to disk. The capture system is built entirely from off-the-shelf components. The system is small enough to move anywhere in the environment where a person may walk. We replace path planning with image redundancy. We may capture an environment more in areas of potential occlusion or where a user may want to study objects in detail – this does not require planning prior to capture, but can be decided on the spot. Redundant data is automatically discarded at a later stage.

Prior to capture, we calibrate the intrinsic camera parameters [12]. The camera manufacturer provides a calibration method, but it does not yield sufficiently accurate results. The manufacturer assumes that the lens is perfectly telecentric and produces an orthographic projection onto the film plane. However, relaxing the assumption of telecentricity, we obtain more accurate intrinsic parameters [2], which we fix for the entire capture session.

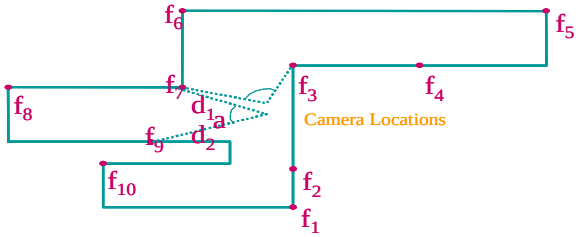

Figure 3: Capture and Calibration. We place small portable fiducials f_k that track robustly in the environment. Using triangulation (distances d_i and angle a), we obtain an estimated camera position and orientation from every pair of visible fiducials. A coarse floor plan predicts the location of fiducials in the captured image. Bundle adjustment optimization refines the global location of the fiducials and the camera pose for each captured image.

Our camera pose estimation algorithm tracks fiducials in the captured images and triangulates position and orientation. The process begins by an operator creating an approximate floor plan of the environment (e.g., using a tape measure). Then, the algorithm uses the floor plan to suggest fiducial locations so as to ensure a sufficient number of fiducials are visible at all times. To obtain high reliability and accuracy, the fiducials are small battery-powered light bulbs placed in the environment by the operator. After providing the initial camera position, the algorithm uses the current position estimate and the floor plan to determine which fiducials may be visible in the current image (Figure 3), tracks these from image to image, and triangulates camera position and orientation.

The fiducials, used for capture, appear in the captured images. Although they do not seem to interfere with the visualization, we could later use image-processing techniques to remove the fiducials and replace them with an estimate of the local background color.

To determine a globally consistent set of camera poses and fiducial locations, we use bundle adjustment [34], a non-linear least squares optimization method. We alternate between using the estimated fiducial locations to compute the camera pose for each captured image and using a sparse subset of the camera pose estimates to refine the global position of the fiducials (i.e., about 10% of the

pose estimates uniformly distributed through the data set).

The goal of the bundle adjustment procedure is to find the 3D fiducial locations (X_f, Y_f, Z_f) and camera poses (x_i, y_i, ω_i) that minimize the difference between the observed fiducial locations T_{if} and the projection of the estimated fiducial locations. The function $P_{if}(X_f, Y_f, Z_f, x_i, y_i, \omega_i)$ encapsulates the projection from 3-space onto our omnidirectional images [25]. If the observed error is zero-mean Gaussian, then bundle adjustment corresponds to the maximum likelihood estimator. The error term used for bundle adjustment is given below (the Cronecker delta term δ_{if} is 1 when fiducial f was tracked on image i):

$$e = \sum_i \sum_f \vec{\delta}_{if} || P_{if}(X_f, Y_f, Z_f, x_i, y_i, \omega_i) - \vec{T}_{if} ||$$

When this process has converged, we obtain a dense set of omnidirectional images with calibrated camera parameters on a plane at eye height.

5 Compression

The second problem is the compression of the massive amounts of acquired data. Our captured data sets usually contain thousands of images, requiring gigabytes of storage. However, only a small portion of the data is needed to reconstruct an image for a novel viewpoint, and there is a great deal of coherence among the images from adjacent viewpoints. Thus, as long as our disk storage system is large enough to hold the captured data, the main problem is working set management.

Our focus is quite different from previous work on data management for IBR representations. For instance, the original Light Field paper [20] describes a method that uses vector quantization and Lempel-Ziv coding. Follow-up work has investigated other compression schemes [27, 15, 22]. However, these methods focus on reducing the overall storage requirements of the IBR representation. They assume that the entire data set (or large subsets) will be decompressed and stored in memory before any novel image is rendered. Of course, this assumption is unrealistic in situations such as ours where the size of the IBR representation exceeds the capacity of host memory.

We make the assumption that disk storage is sufficient to hold the entire data set and that the goal is to compress the data into a form suitable for efficient access during an interactive walkthrough. The challenge is to take advantage of the redundancy in the data while optimizing the data layout for cache coherence.

The first and most obvious option is to compress and store every captured image independently, e.g., in a separate JPEG file. (In fact, we JPEG compress the data during capture.) This approach is very straightforward, but does not take advantage of inter-viewpoint redundancy, which can improve both storage and bandwidth utilization.

A second option is to utilize prediction and replace some images with the residual, or difference, between the original image and the predicted image. With a good prediction strategy, the residual has less energy than the original image, resulting in significant compression gain. This technique is used in video codecs based on motion compensation, such as MPEG [1]. A similar strategy has been used for encoding far-field representations in a walkthrough system for synthetic environments [35].

One difficulty in predictive coding is finding the proper spacing of the I-frames. Frequent I-frames yield little compression gain. Infrequent I-frames make non-linear image access inefficient, and also yield poor cache utilization for our walkthrough application.

[1] MPEG uses the terms "I-frames" for the original frames and "P-frames" for the predicted frames; we use the same notation.

We optimize both compression and cache utilization by storing images in a multiresolution hierarchy. Instead of the MPEG strategy of I-frames followed by a number of P-frames followed by an I-frame, etc., we store images in nested trees, where the root of each tree is an I-frame and all other nodes are P-frames. Since the system caches image data associated with interior nodes of the hierarchy, only P-frames must be read for nearby viewpoints lower in the hierarchy, improving the disk-to-memory bandwidth utilization.

Our multiresolution data structure is a binary tree built bottom up using half edge-collapse operations [14]. The tree is initialized by creating a set of nodes representing each captured image at its viewpoint. Using a (Delaunay) triangulation of the nodes, we compute a priority for every edge and place the edges in a heap. We collapse the highest priority edge and replace the nodes of the edge endpoints with a new common parent node (Figure 4a). To avoid introducing resampled images in the hierarchy, we place the parent node at the same viewpoint location as one of its children and replace the original image with an image reference. We then locally retriangulate the current node-set, update the heap, and repeat the collapse process for the next highest priority edge (Figure 4b) until no edges remain in the heap, storing a full image with the remaining (root) node (Figure 4c).

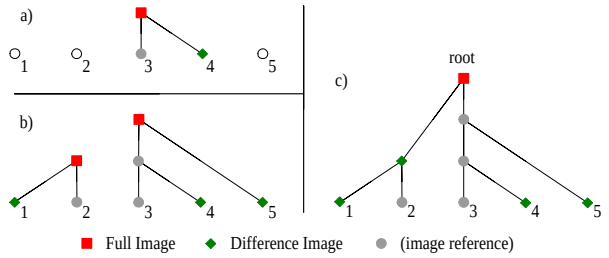

Figure 4: Multiresolution Tree. We show a three-step sequence (a-c) of building a small tree for a set of five captured images.

We have considered several measures to compute edge collapse priority. Ideally, we would like to collapse edges between the most similar images first. One measure is to compute the image energy of the residual. A faster approximation is to collapse the shortest edge, which tends to work well since nearby images are often the most similar.

In order to optimize the compression gain, working set size, and decompression time, we keep I-frames at every L levels in the tree ($L = 5$ for our environments), creating subtrees of P-frames, with each P-frame predicted from the root of its subtree. While prediction relative to the root instead of another P-frame does not give optimal compression, it improves the working set size and decompression time.

Our motion prediction utilizes a simple 3D geometric proxy of the environment to warp the image at the root of the subtree to the location of its child. The residual is the difference of the predicted child and the root image. To improve compression performance, we optionally use an optimization process for each difference image computation. The optimization searches for the best set of translation and rotation offsets to register the proxy to the images so as to minimize image energy and improve compression.

6 Real-Time Data Management

The third problem is the management of the out-of-core data in a real-time walkthrough system. Our goal is to pre-load and cache images from disk for rendering novel views as a user navigates interactively through the IBR environment. Given the amount of available storage, bandwidth, and processing capacity of each hardware component, our task is to develop a time-critical algorithm

that loads and caches data in a manner that produces the highest quality images while maintaining an interactive frame rate [10].

Ideally, the algorithm guarantees that a set of images is always surrounding the viewpoint allowing for good reconstruction and smooth transitions from one set of images to another. Moreover, upon failure, the algorithm should exhibit graceful degradation and eventually return to maximum quality. To this end, we maintain a cut through the multiresolution tree. This cut guarantees that at any location within the walk-able space we always have a set of surrounding images. These images will be used for reconstruction, as is described in the following section.

We use an asynchronous prefetching process to maintain a cache of images in main memory. The user's velocity predicts which images are needed next. The prefetcher maintains two linked lists of tree nodes (Figure 5). The evict list defines a cut through the tree such that all nodes on and above this cut are in cache. The fetch list consists of all the immediate children of the evict list nodes. The fetch list and the evict lists are sorted by priority; the first image in the fetch list has the highest probability of being needed next and the last image in the evict list has the highest priority of being needed next. Initially, the evict list consists of just the root node and the fetch list are the immediate children of the root node. To display each frame at run-time, the renderer uses the current cut to determine which images to use for reconstructing the current view.

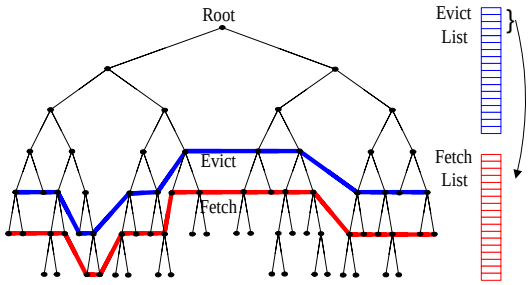

Figure 5: Evict and Fetch Lists. We maintain two linked lists of tree-nodes that define a cut through the tree: the evict list and the fetch list. The prefetching process updates the likelihood of needing the nodes in both lists. As long as there is a fetch node that is more likely to be needed than the least likely evict node, and cache space can be made available, the nodes are swapped.

To update the cut, we swap nodes between the two lists. We use the observer's latest viewpoint and predicted velocity to compute for all nodes in the evict list and fetch list the probability that its image is needed next. The probability of a node increases as the observer gets closer to the node and as the direction from the observer to the node approaches the viewing direction. Since in our hierarchy we force nodes higher in the tree to be loaded first, the distance to a node is determined by the distance to the bounding box surrounding the parent and all its children. If the cache is not yet full, we swap the node in the evict list that has the lowest probability with the node with the highest probability in the fetch list and then load the image data. If the cache is full, we must swap and flush the image data of a sufficient number of evict nodes to fit the image data for the next fetch node. The prefetching process recomputes the probabilities and sorts the fetch and evict lists at regular intervals (e.g., 10 to 30 times a second).

If desired, we could optimize the prefetcher for narrow FOV reconstructions (e.g., 60 degrees). For instance, by subdividing each omnidirectional image into several tiles, the prefetcher could use the current predicted viewing direction to only load the subsets of the omnidirectional images necessary for the FOV to reconstruct.

7 Reconstruction

The final stage of our process is to reconstruct synthetic images for novel viewpoints. The goal is to produce high quality images with little blurring or popping during an interactive sequence.

Since we are interested in first-person walkthroughs, we optimize our system for reconstruction from viewpoints near the eye-height plane in which the images were captured. We project the user's viewpoint onto the triangular mesh of images and extract the three closest images, which we warp to the viewpoint, and blend using the barycentric coordinates of the viewpoint [9].

Since the images to blend have different centers of projection, we warp them using a set of 3D "feature points" distributed on the surface of a coarse polygonal proxy. We project the visible feature points onto the image planes of the novel and captured viewpoints. Then, we use the features to morph each captured image to the novel viewpoint [7, 19], obtaining a reconstructed omnidirectional image. In addition, we can create arbitrary projections, including planar and cylindrical (Figure 6).

An alternate approach is to map the captured omnidirectional images onto the proxy directly and blend them using texture mapping. But, this approach may produce artifacts, such as the incorrect mapping of portions of the images to the wrong surfaces, especially along silhouette edges of the proxy.

The accuracy of the proxy becomes increasingly important as the observer viewpoint approaches an object. Our reconstruction method may start exhibiting ghosting artifacts when the viewer gets too close to a small object, occupying the entire field-of-view (FOV). To remedy this, we could either sample more densely, create a more accurate 3D proxy [33, 18, 28, 37], or use more complex reconstruction algorithms, including ones that combine image data from multiple reference images [3, 4, 13, 20]. Even without these extensions, a novel viewpoint is almost always near three captured views, which are pre-filtered versions of images very similar to the desired image, so we can usually produce resampled images with quality almost equal to the captured images.

Figure 6: Reconstruction Examples. We show several reconstructed images for the library environment. The left image is the reconstructed omnidirectional image. The upper right is a cylindrical projection and the bottom right is a planar projection.

8 Implementation Details

Our software system is implemented in C/C++ and OpenGL on a SGI Onyx2 with 4 195MHz R10000 processors using an Infinite-Reality2 graphics subsystem.

Our omnidirectional video camera is based on a commercial Cyclovision/Remote Reality S1 unit [25], which acquires 1024x1024 images over a hemispherical FOV. The camera is placed on top of a motorized cart carrying a battery and a small PC, and it is moved using radio remote control. The cart moves at an average speed of 7 inches/sec and our RAID disk holds approximately 50GB of

Env.	Setup	Capture	Fiducial Tracking	Pose Optimization	Area	Avg. Dist to Images	No. Images
Library	15 min	17 min	60 min	20 min	120 sq. ft	1.6 in.	1,947
Office	5 min	10 min	50 min	25 min	30 sq. ft	0.7 in.	3,475
Museum	30 min	82 min	160 min	30 min	900 sq. ft	2.2 in.	9,832

Table 1: Capture and Image Statistics. We captured three Sea of Images, covering a total of 1050 square feet with 15254 images at an average image spacing of 1.5 inches, and average capture and calibration time of 2.8 hours per environment.

Env.	Raw (MB)	Flat (MB)	Diff-Image Hierarchy (MB)	Optimized Hierarchy (MB)	Tree Building	Diff Images	Optimized Diff Images
Library	6000	318	150 (40:1)	113 (54:1)	4 min	60 min	3 hrs
Office	11000	376	198 (55:1)	139 (79:1)	6 min	35 min	5 hrs
Museum	31000	1,560	740 (42:1)	564 (55:1)	50 min	160 min	14 hrs

Table 2: Compression Statistics: We show the raw, JPEG-compressed, and multiresolution hierarchy sizes of our Sea of Images. The multiresolution hierarchy sizes and times are shown for both unoptimized and per-difference-image optimized compression. The average compression time is 4.7 hours per environment.

data. We compress the images on the PC and write them to disk in JPEG format at a rate of 2 frames per second, which yields an image every couple of inches for nearly 60 hours. We move the cart back and forth along closely spaced paths (separated by a couple of inches) to acquire a dense Sea of Images over a plane at eye height throughout the entire environment.

We use graphics hardware to compute the difference images for the multiresolution tree. Current graphics hardware does not support an efficient way to compute image energy, so we use the CPU. To accelerate the precomputation, we typically compute image energy values at 128x128 pixel resolution, noting that at full resolution we should expect similar relative values.

Our reconstruction algorithm also takes advantage of graphics hardware. While features are interpolated on the CPU, all image data is blended using either multiple-pass texture blending or the accumulation buffer. We rely upon a NetLib package for Delaunay triangulation of features and the graphics subsystem for paging of texture memory on demand.

9 RESULTS & OBSERVATIONS

In this section, we evaluate how well the Sea of Images approach achieves the goal of walking through a complex environment with photorealistic images. Specifically, we ask the following questions: (1) is our capture process practical?, (2) are our data management strategies effective?, (3) can the system maintain interactive frame rates?, and (4) does it in fact reconstruct images with complex illumination and visibility effects during interactive walkthroughs?

So far, we have experimented with three environments. The first one ("Office") is a small room, around the size commonly used for IBR demonstrations. The second one ("Library") is the lobby area of a large public library. Finally, the third environment ("Museum") is a small museum. It provides our most challenging test case because it has a concave floor-plan, it contains many specular surfaces (glass cases and brass plaques on the wall), and it contains complex geometry (plants on the floor and museum pieces in the cases). No previous walkthrough system could produce photorealistic images of such a difficult environment at interactive rates.

Table 1 presents statistics about our capture process. Overall, the capture process for each of the three environments took only a couple of hours. The setup time includes making lighting changes, camera setup (e.g. aperture adjustments) and placing and measuring the fiducial setup in the environment. The actual capture time during which the motorized cart moved through the environment was relatively short (e.g., 10 minutes per 1,000 images). The proxy for each environment was created using a measuring tape and a simple text-based polygon editor. The entire capture process yielded a Sea of Images cumulatively covering more than 1,000 square feet of walkable area at a density such that the average distance from a random point on the eye height plane to its nearest image is 1.5 inches. We conjecture that few other capture processes would be able to both cover such a large space and sample fine details within the viewpoint space typical of an interactive walkthrough.

Table 2 lists the size of the data before and after compression, as well as compression times. Overall, the three environments contain 15,254 images and require 48GB of disk space in their raw form. JPEG compression of the images reduced the size of the data sets to 2.25GB (at quality factor 75). Our multiresolution compression strategy achieved another 2-3X compression factor. This extra compression is determined mainly by the efficiency of our coder for difference images and by the order in which we collapse nodes in our multiresolution tree structure. By computing per-difference-image optimized translation and rotation offsets for registering the proxy to the images, we obtain a compression performance of 54-79X of the original data, at the expense of 5 seconds of optimization time per difference image (totalling about 3 hours, 5 hours, and 14 hours for the Library, Office, and Museum environments, respectively). Without this optimization, we obtain 40-55X compression of the original data (totalling about 1 hour, 0.5 hours, and 2.5 hours). The images and sequences we show use the optimized offsets.

Resol.	0.25 m/s	0.5 m/s	1.0 m/s
256	1.00 ($\sigma = 0.10$)	1.04 ($\sigma = 0.20$)	1.98 ($\sigma = 1.87$)
512	1.03 ($\sigma = 0.25$)	1.51 ($\sigma = 1.02$)	3.24 ($\sigma = 2.68$)
1024	1.38 ($\sigma = 0.83$)	2.72 ($\sigma = 2.13$)	5.11 ($\sigma = 4.78$)

Table 3: Prefetching Performance. We report the effectiveness of the prefetcher as a ratio of two distances: the average distance to a fetched image and the average distance to any captured image.

The prefetcher uses the multiresolution hierarchy to ensure that the reconstruction algorithm always has some set of images in memory surrounding the novel viewpoint. To measure the performance of the prefetcher, we collected statistics as the user walked through the Museum environment at different speeds along a typical path. We also used different image resolutions to test the sensitivity of our prefetcher to increasing data sizes. During these tests, we estimated how well the prefetcher was working by taking the average distance to the surrounding closest trio of cached images and dividing it by the average distance to the closest among all images - a value of 1.0 is a perfect score (Table 3). Note how the distance to the closest loaded image increases as the user walks faster or asks for higher resolution images. This result corresponds to graceful degradation in image quality in our walkthrough system.

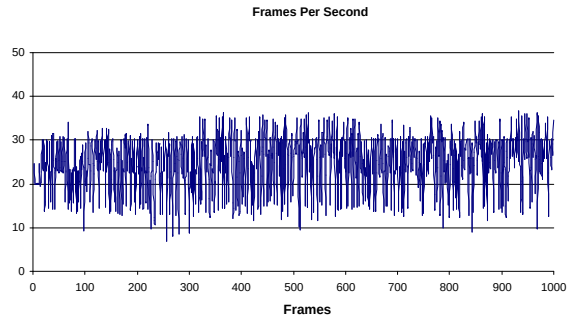

Figure 7: Frame Rate. We show the frame rate for a pre-recorded path through the museum environment at 512x512 pixels. The average frame rate is 25 frames per second.

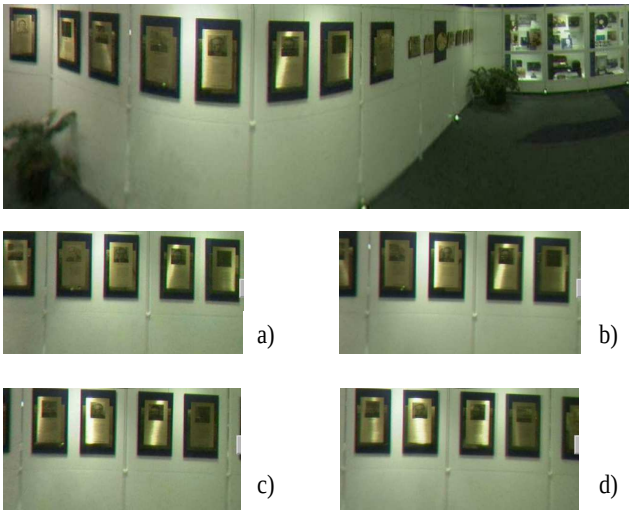

Figure 8: Specular Highlights. This figure demonstrates specular highlights moving over the surface of a shiny bronze plaque in the museum environment. The top snapshot shows a cylindrical projection of a subset of the reconstructed image. The four-image sequence is a zoomed-in planar projection of the reconstructed image. Observe the highlight moving across the plaques as the viewpoint moves laterally (a-d).

Figure 9: Captured vs. Reconstructed Comparison. The top omnidirectional image is one of the nearby captured images. The bottom omnidirectional image is reconstructed for a novel viewpoint that is as far as possible from the surrounding images.

Our current run-time system reconstructs novel views at an average of 15-25 frames per second. Reconstructions at 512x512 or 256x256 pixels can be performed at an average of 25 frames per second, while higher resolution views at 1024x1024 pixels are produced at about 15 frames per second. Figure 7 plots the frames per second for an example path through the museum environment using 512x512 images. The frame rate is near constant because our system always warps three images into the frame buffer. We believe the variations in frame times are due to texture swapping.

Finally, Figures 8-10 show the quality of novel views reconstructed by our Sea of Images approach. We can reproduce specular highlights moving over a surface without modeling material properties. For instance, Figure 8 shows an example specular highlight moving over the surface of a shiny bronze plaque. Because the density of sampled viewpoints is so large, we can almost always achieve reconstructions of similar quality to the captured omnidirectional images. Figure 9 shows a captured image (top) and a synthetic image (bottom) reconstructed from a novel viewpoint at the middle of a Delaunay triangle of captured viewpoints. Differences in the images are almost imperceptible. Also, we can reproduce images of objects at multiple resolutions. Figure 10 demonstrates renderings of a single object at multiple distances.

10 Conclusion & Future Work

We have introduced a novel approach for creating interactive walkthroughs using image-based rendering techniques. What distinguishes our approach from other IBR approaches is that we are not limited to small environments with restricted visibility changes such as with the Lightfield/Lumigraph [13, 20]. We are also not constrained to sparse sampling, which in turn limits rendering to diffuse environments and often requires viewpoint planning and a priori information about the geometry of the scene. We remove these constraints by mapping the IBR problem onto a compression and working-set management problem.

However, the system is not without limitations. First, our current capture device (a catadioptric omnidirectional camera) has limited resolution. While this affects the resolution of images currently generated, the methods described in this paper are independent of a particular capture device and would work for any omnidirectional video camera arrangement. We have purchased a FullView camera [24] and begun to investigate other multiple-camera configurations that acquire higher resolution images.

Second, we currently acquire image data sampling the lower hemisphere of directions from viewpoints at some density on an eye-height plane. The resulting representation provides enough information for reconstruction of any downward-looking view for which the eye-height plane is unoccluded. Expanding our capture procedure to acquire images containing a wider range of view directions and/or multiple viewpoint heights would expand the space of allowable novel views. We are investigating use of spherical cameras to partially address this issue. In addition, we would like to be able to determine a conservative estimate of what average image capture spacing is ”dense enough” for a particular environment.

Third, we are looking into other (multiresolution) reconstruction methods for a Sea of Images. In particular, tracking visible features from image-to-image in order to generate more accurate correspondences between the images surrounding the novel viewpoint.

Finally, our capture process currently requires some manual effort. An operator estimates the locations of fiducial, builds an approximate proxy model, and drives a motorized cart back and forth via remote control. While these activities are not too burdensome (they take 15 to 112 minutes for our three environments), the system could be made easier to use by further automating the capture process. This is possible by estimating camera pose in real-time from the fiducials or from automatically detected features, and letting the

Figure 10: Far-to-Near Image Sequence. We show the museum environment and focus on a particular item from three distances.

PC control the wheel motors to drive the cart autonomously. This is a topic for future work.

In conclusion, we believe the Sea of Images approach to be a fertile bed for future work. Never before have researchers had access to such a large and dense sampling of an environment. We believe it could lead to new 3D reconstruction algorithms, novel compression strategies, and new walkthrough applications.

ACKNOWLEDGMENTS

We are grateful to Sid Ahuja, Multimedia Communications Research VP at Bell Labs, for supporting this research. We also would like to extend our gratitude to Bob Holt for his mathematical help and to the Bell Labs and Princeton University staff that allowed us to capture their environments. Thomas Funkhouser is partially funded by a NSF CAREER grant (CCR-0093343).

REFERENCES

[1] E.H. Adelson and J. Bergen. The plenoptic function and the elements of early vision. In *Computational Models of Visual Processing*, pages 3–20, Cambridge, MA, 1991. MIT Press.

[2] Daniel G. Aliaga. Accurate catadioptric calibration for real-time pose estimation in room-size environments. In *IEEE International Conference on Computer Vision*, pages 127–134, July 2001.

[3] Daniel G. Aliaga and Ingrid Carlbom. Plenoptic Stitching: A scalable method for reconstructing 3D interactive walkthroughs. In *Proceedings of ACM SIGGRAPH 2001*, pages 443–450, August 2001.

[4] Chris Buehler, Michael Bosse, Leonard McMillan, Steven J. Gortler, and Michael F. Cohen. Unstructured Lumigraph Rendering. In *Proceedings of ACM SIGGRAPH 2001*, pages 425–432, August 2001.

[5] Chun-Fa Chang, Gary Bishop, and Anselmo Lastra. LDI Tree: A hierarchical representation for image-based rendering. In *Proceedings of ACM SIGGRAPH 1999*, pages 291–298, August 1999.

[6] Shenchang Eric Chen. Quicktime VR - an image-based approach to virtual environment navigation. In *Proceedings of ACM SIGGRAPH 1995*, pages 29–38, August 1995.

[7] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proceedings of ACM SIGGRAPH 1993*, pages 279–288, August 1993.

[8] William Dally, Leonard McMillan, Gary Bishop, and Henry Fuchs. The Delta Tree: An object-centered approach to image-based rendering. In *MIT AI Lab Technical Memo 1604*, May 1996.

[9] P. E. Debevec, Y. Yu, and G. D. Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In *Eurographics Rendering Workshop 1998*, pages 105–116, June 1998.

[10] Thomas A. Funkhouser. Database management for interactive display of large architectural models. In *Graphics Interface '96*, pages 1–8. Canadian Information Processing Society / Canadian Human-Computer Communications Society, May 1996.

[11] A. Zisserman G. Simon, A. Fitzgibbon. Markerless tracking using planar structures in the scene. In *IEEE and ACM International Symposium on Augmented Reality*, 2000.

[12] C. Geyer and K. Daniilidis. Catadioptric camera calibration. In *IEEE International Conference on Computer Vision*, pages 398–404, 1999.

[13] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The Lumigraph. In *Proceedings of ACM SIGGRAPH 1996*, pages 43–54, August 1996.

[14] Hugues Hoppe. Progressive meshes. In *Proceedings of ACM SIGGRAPH 1996*, pages 99–108, August 1996.

[15] I. Ihm and R. Lee. On enhancing the speed of splatting with indexing. In *IEEE Visualization*, pages 69–76, Atlanta, GA, 1995. IEEE.

[16] S.H. Jung and C.J. Taylor. Camera trajectory estimation using inertial sensor measurements and structure from motion results. In *IEEE Computer Vision and Pattern Recognition*, pages 732–737, 2001.

[17] S.B. Kang and R. Szeliski. 3D scene data recovery using omnidirectional baseline stereo. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 364–370, 1996.

[18] Reinhard Koch, Marc Pollefeys, and Luc Van Gool. Realistic surface reconstruction of 3D scenes from uncalibrated image sequences. *The Journal of Visualization and Computer Animation*, 11(3):115–127, July 2000.

[19] Seungyong Lee, George Wolberg, and Sung Yong Shin. Polymorph: Morphing among multiple images. *IEEE Computer Graphics and Applications*, 18(1):58–71, January/February 1998.

[20] Marc Levoy and Patrick M. Hanrahan. Light Field Rendering. In *Proceedings of SIGGRAPH 96*, pages 31–42, August 1996.

[21] A. Lippman. Movie-maps: An application of the optical videodisc to computer graphics. In *Proceedings of ACM SIGGRAPH 1980*, pages 32–42, July 1980.

[22] M. Magnor, A. Endmann, and B. Girod. Progressive compression and rendering of light fields. *Proc. Vision, Modeling, and Visualization (VMV-2000)*, Saarbrücken, Germany, pages 199–203, 2000.

[23] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *Proceedings of SIGGRAPH 95*, pages 39–46, August 1995.

[24] V. Nalwa. A true omnidirectional viewer. In *Technical Report*, Bell Laboratories, Holmdel, NJ, 1996.

[25] S. Nayar. Catadioptric omnidirectional camera. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 482–488, 1997.

[26] L. Nyland, A. Lastra, D. McAllister, V. Popescu, and C. McCue. Capturing, processing, and rendering real-world scenes. In *Videometrics and Optical Methods for 3D Shape Measurement, Electronic Imaging, Photonics West*, volume 4309, 2001.

[27] Ingmar Peter and Wolfgang Straßer. The Wavelet Stream: Interactive Multi Resolution Light Field Rendering. In *Rendering Techniques 2001: 12th Eurographics Workshop on Rendering*, pages 127–138. Eurographics, June 2001.

[28] M. Pollefeys, M. Vergauwen, K. Cornelis, J. Tops, F. Verbiest, L. Van Gool. Structure and motion from image sequences. In *Proc. Conf. on Optical 3-D Measurement Techniques*, pages 251–258, Oct. 2001.

[29] D.R. Roberts and A.D. Marshall. A review of viewpoint planning. In *Technical Report 97008*, University of Wales, College of Cardiff, Department of Computer Science, 1997.

[30] Heung-Yeung Shum and Li-Wei He. Rendering with concentric mosaics. In *Proceedings of SIGGRAPH 99*, pages 299–306, August 1999.

[31] Wolfgang Stuerzlinger. Imaging all visible surfaces. In *Graphics Interface '99*, pages 115–122, June 1999.

[32] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic mosaics and environment maps. In *Proceedings of ACM SIGGRAPH 1997*, pages 251–258, August 1997.

[33] Seth Teller, Matthew Antone, Zachary Bodnar, Michael Bosse, Satyan Coorg, Manish Jethwa, and Neel Master. Calibrated, Registered Images of an Extended Urban Area. In *IEEE Computer Vision and Pattern Recognition*, December 2001.

[34] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Vision Algoritms: Theory and Practice*. Springer-Verlag, 2000.

[35] A. Wilson, D. Manocha, and K. Mayer-Patel. Spatially encoded far-field representations for interactive walkthroughs. In *ACM Multimedia*, pages 348–357, 2001.

[36] Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. Surface Light Fields for 3D Photography. In *Proceedings of ACM SIGGRAPH 2000*, pages 287–296, July 2000.

[37] Andrew Zisserman, Andrew W. Fitzgibbon, and Geoff Cross. VHS to VRML: 3D graphical models from video sequences. In *ICMCS, Vol. 1*, pages 51–57, 1999.