Image Warping for Compressing and Spatially Organizing a Dense Collection of Images

Daniel G. Aliaga* Paul Rosen Voicu Popescu Ingrid Carlbom

{aliaga|rosen|popescu}@cs.purdue.edu carlbom@lucent.com

Department of Computer Science *Bell Labs*

Purdue University *Lucent Technologies*

Abstract

Image-based rendering (IBR) systems create photorealistic views of complex 3D environments by resampling large collections of images captured in the environment. The quality of the resampled images increases significantly with image capture density. Thus, a significant challenge in interactive IBR systems is to provide both fast image access along arbitrary viewpoint paths and efficient storage of large image data sets.

We describe a spatial image hierarchy combined with an image compression scheme that meets the requirements of interactive IBR walkthroughs. By using image warping and exploiting image coherence over the image capture plane, we achieve compression performance similar to traditional motion-compensated schema, e.g., MPEG, yet allow image access along arbitrary paths. Furthermore, by exploiting graphics hardware for image resampling, we can achieve interactive rates during IBR walkthroughs.

Keywords: Image-based rendering, spatial hierarchy, compression, hierarchical, random access.

* = contact author: USA – Ph. 765-496-7943, Department of Computer Science, 250 N. University St. West Lafayette, IN 47907

1. Introduction

Computer graphics applications such as telepresence, virtual reality, and interactive walkthroughs require a three-dimensional (3D) model of real-world environments. Students can "visit" famous historical sites, such as museums, temples, castles, battlefields, and entire history rich cities; archeologists can capture excavation sites as they evolve over time; soldiers and fire fighters can train in safe simulated environments; real estate agents can show potential buyers the interiors of homes for sale via the Internet; and, people all over the world can enjoy virtual travel and multi-player 3D games. Thus, a growing desire exists for methods which can efficiently capture important and visually stunning environments.

In recent years, image-based modeling and rendering approaches (IBMR) have addressed this problem [1]. They do so by directly resampling images to generate new views, thus avoiding the need for a detailed geometric model. The 7D plenoptic function describes the light intensity passing through every viewpoint (x, y, z), in every direction (θ, φ), for all time t, and for every wavelength λ [2, 3]. All existing IBMR techniques generate lower-dimensional plenoptic functions from a set of images [4, 5, 6, 7, 8, 9].

IBMR techniques require a large collection of images that must be efficiently stored and accessed. In particular, for interactive walkthroughs image access cannot be limited to coincide with the capture paths, but instead requires access along arbitrary viewpoint paths. Furthermore, disk-to-memory bandwidth limitations require algorithms that reduce both the size of the images on disk and the amount of data that must be transferred to main memory as a virtual observer navigates through a captured 3D environment.

Traditional compression techniques do not provide both access flexibility and full exploitation of data redundancy. Image compression, such as JPEG [10], JPEG2000 [11], and 2D wavelets [12] exploit intra-image redundancy to reduce individual image sizes but do not take advantage of inter-image redundancy. Video compression achieves a significant improvement in overall compression by encoding sparse reference images and using motion-estimation algorithms to determine the coherence for interpolation

between reference images (e.g., MPEG [13]). However, motion-estimation is expensive and is intended for pre-determined linear sequences of images, making it ill-suited for image access along arbitrary viewpoint paths.

We propose a spatial image hierarchy combined with a model-based compression algorithm that provides quick access to images along arbitrary viewpoint paths during interactive walkthroughs and enables efficient compression of high-resolution images whose centers of projection (COPs) irregularly sample a plane (Figure 1). Specifically, our method arranges reference images and residual images into a binary tree. A captured image is extracted via a sequence of image warping operations. Each operation warps a reference image to the viewpoint of a residual image and the two are added together. We eliminate the need for costly motion estimation by using a simple user-supplied geometric proxy of the environment. The proxy, which may consist of only a dozen polygons approximating the environment, helps compensate for the translation between the two viewpoints and provides an inexpensive but accurate mapping from the reference to the residual image. We further reduce the size of the residual images by employing occlusion camera reference images (OCRIs) [14]. These images store samples visible from the reference viewpoint as well as samples likely to become visible from nearby viewpoints (see reference image of Figure 1). Warping OCRIs images instead of regular images produces more compact residuals when disocclusions occur and thus may achieve improved overall compression.

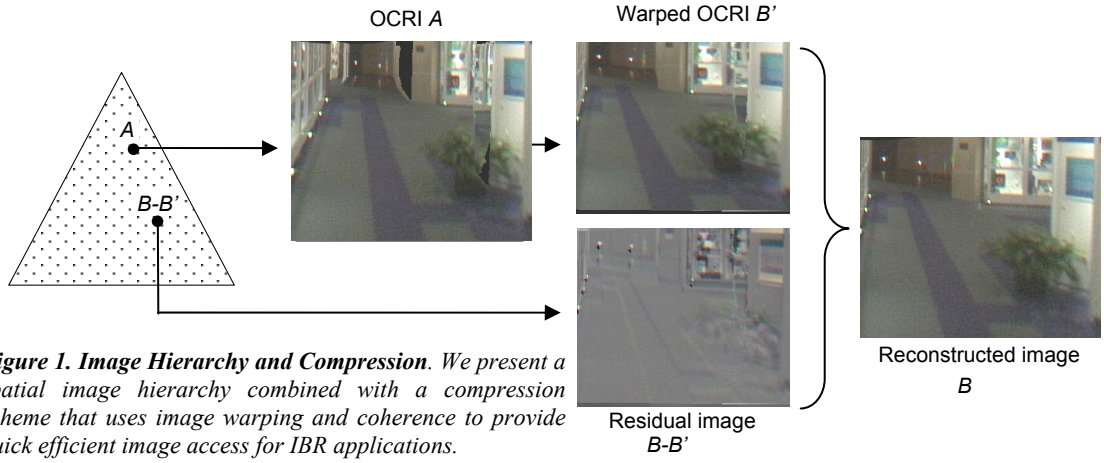

Figure 1. Image Hierarchy and Compression*. We present a spatial image hierarchy combined with a compression scheme that uses image warping and coherence to provide quick efficient image access for IBR applications.*

Residual images primarily account for surfaces that are visible from the viewpoint of the residual but not from the reference viewpoint. In the case of non-trivial scenes, even small viewpoint translations disocclude many surfaces that are not sampled in the reference image. Furthermore, the disocclusions are not grouped in one contiguous region of missing samples, but are rather scattered throughout the scene. Therefore, when regular reference images are used, residual images have to compensate for a substantial number of samples and are difficult to compress.

An OCRI is built using a virtual non-pinhole camera model that "sees" around occluders to gather samples that are hidden but are close to the edge of their occluder and therefore are likely to become visible even for small viewpoint translations. For synthetic imagery, a virtual camera model is implemented using graphics hardware. For real-world imagery, the proxy is used in place of the synthetic model to support creating a virtual camera model. In both cases, OCRIs, like regular depth images, have a single layer and therefore share the advantages of bounded number of samples, implicit connectivity, and efficient incremental processing. When disocclusions are present, these additional samples reduce the number of missing samples for which the residual has to compensate, producing residual images with less energy.

In experiments with environments of 2000 to 10,000 omnidirectional images, 1024x1024 pixels each, our method gives nearly a factor of a 100-to-1 compression without significant loss in quality. This article expands the description of methods presented in our conference publication [15] and extends the work to support the use of a model-based compression algorithm based on OCRIs.

The major contributions of our work include:

- a fast technique to access compressed images along arbitrary viewpoint paths for IBMR applications,
- an efficient hierarchical image compression algorithm using reference images and residual images to exploit image coherence over a plane or volume of space, and

- a method for creating and warping reference images for use in inter-image compression that combine into a single layer image the surfaces visible for a neighborhood of viewpoints.

2. Related Work and Motivation

Several image-based rendering systems use schemes for organizing and compressing images. For example, the Lightfield [8] and the Lumigraph [7] form flat hierarchies of images captured in front of an object of interest. The original Lightfield paper exploits image redundancy by using vector quantization (VQ) [16] followed by Lempel-Ziv [17] encoding. For interactive rendering, the entire dataset is first decoded using Lempel-Ziv followed by random image decompression using the VQ codebooks. VQ achieves only about 6:1 to 24:1 compression for their datasets.

Subsequently, more elaborate compression algorithms have been proposed for Lightfields. Magnor and Girod [18] describe a DCT-based coder for regular Lightfields using datasets of 1024 images. The images are arranged into a quad-tree that encodes images as either I- or P-frames. Images of 256x256 pixels are decoded and reconstructed interactively. Optionally, disparity information can be estimated and used to further compress images, although interactive performance is lost. For display, all I-frames and residuals are decoded and stored in memory. This approach achieves 100:1 to 1000:1 compression depending on desired quality and scene characteristics.

Peter and Strasser [19] describe a wavelet-based compression algorithm for the same Lightfield datasets. A three-layer cache system achieves interactive rates (15-20 fps) for 256x256 pixel images – upon too many cache misses or without the cache, performance is reduced to 3 fps. This system demonstrates up to 100:1 compression.

Gotz et al. [20] describe a novel image representation for cylindrical projection images. Their representation exploits spatial coherence and rearranges the columns of pixels. Columns that correspond to the same world-space viewing direction are grouped and stored either as index columns or as residual columns. Each column is coded using a one-dimensional wavelet and the coefficients are quantized and

stored using a Huffman code. Overall they achieve compression performance similar to JPEG (15-20x) but, unlike JPEG, are able to incrementally code new columns and efficiently decode individual columns.

Wilson et al. [21] create a system for interactive walkthroughs of synthetic models and use spatially encoded video to represent image-based impostors [22]. They decompose the model into a regular grid of rectangular viewing cells. For viewpoints inside a cell, they represent the geometry outside the cell by mapping pre-computed images of the far field onto the interior walls of the cell. Their scheme exploits the redundancy between the images of neighboring cells. Using MPEG nomenclature, cells are classified as either I-cells or B-cells. An image-based impostor of a B-cell references two images from the nearest two I-cells. Using depth information obtained from the synthetic model, they compute motion vectors and code the images using MPEG. To access a coded image at runtime in constant time, they store the bit stream offsets to each image in a globally accessible array. The average compression ratio for their dataset of 22,000 images of 512x512 pixels is 48.

Several strategies have attempted to reduce disocclusion artifacts that occur in image warping. Amongst the first methods was to simply warp several images from nearby viewpoints with the hope to obtain the missing samples [3]. A variation is to combine several images into a layered representation as a preprocessing step [23]. In previous work, we have used such layered-depth images for accelerating the rendering of architectural scenes [24]. Unfortunately, having multiple images or layered images requires more storage and work and does not provide the processing benefits of simple single-layered images. Several non-pinhole camera models have also been proposed (e.g, [25, 26]). However, none of them are powerful enough to handle disocclusions in practical scenes.

In contrast to previous work, our goal is a method to compress a large number of high-resolution images of a real-world environment irregularly sampled over a plane in a manner that it yields high compression performance, is highly scalable, uses single layer images, and permits fast image access along arbitrary viewpoint paths. Our OCRI reference images reduce disocclusions and are automatically created using a fine-grain distortion method. Reference and residual images can be processed, stored, and

decoded using graphics hardware already present on most computers. None of the systems described above support all these features.

Our image hierarchy and compression scheme uses the dense datasets captured with the Sea of Images system [5]. This work replaces the difficult computer vision problems of 3D reconstruction and surface reflectance modeling with the easier problems of motorized cart navigation, dense sampling, and working-set management.

3. Image Hierarchy

Our algorithm builds a tree that exploits image coherence. This section describes how we build the image hierarchy using a binary tree, how reference and residual images are created and encoded in the tree, and how the original images are extracted and decoded from the image hierarchy. Figure 2 provides a summary of the entire process.

3.1 Binary Tree Construction

We incrementally build the tree by collapsing the edges of a Delaunay triangulation of the centers-of-projection (COP) of all the original images. First, a node is created for each image. Second, the edges of the Delaunay triangulation are placed into a priority queue. Third, the next highest priority edge is collapsed until no more edges remain.

There are several possible metrics for edge priority. To maximize compression, edges should be processed in an order that maximizes inter-image coherence. One metric is the energy of the image difference between the two images of the edge. Image energy is estimated by calculating the total absolute pixel luminosity of the residual image or by measuring the size of the intra-coded compressed residual image. However, both these measures are computationally expensive.

Binary tree construction
Create a node for each image.
Calculate Delaunay triangulation of the centers-of-projection of the images.
Build the tree using a priority queue and edge collapse operations.

Image encoding
Use image warping to exploit inter-image redundancy.
Use an optimization process to further reduce image energy.

Image extraction and decoding
Extract images in either logarithmic or constant time, at the expense of compression performance.

Figure 2. Image Hierarchy*. Summary of the process for building an image hierarchy.*

A simpler, but effective, approximation is to use the Euclidean distance between the images (i.e., the edge length) as the metric of image similarity. This is based on the assumptions that nearby images have smaller image differences than those farther apart, and therefore should be processed first. This metric is very fast to evaluate, requiring only a distance calculation, and our experiments prove it gives almost as good a compression performance as the more costly image energy metric.

Using each edge and its associated node pair in priority order, we perform a half-edge collapse operation to convert the node pair into children of a new common parent node. This operation produces three types of tree nodes: I-node, P-node, and N-node. The new reference image parent node (I-node) is placed at the same spatial location as one of its children and contains the image formerly stored with that child. The child node at that location (N-node) simply becomes a pointer to the parent. The other child node (P-node) becomes a residual image, the difference between the original child and the I-node image warped the viewpoint of the child node. Finally, the nodes are locally re-triangulated and the queue is updated with the new edges, while the edges that no longer exist are removed from the queue. After all edges have been processed, the resulting forest of trees joins to form a single tree for the entire original image set (Figure 3).

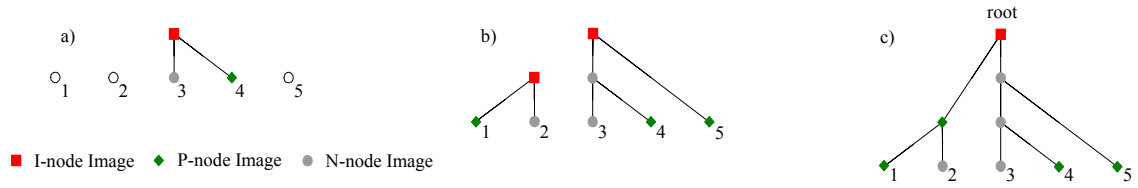

Figure 3. Tree Building. *This figure shows the process of building a tree from an initial set of 5 images. In this didactic example, the images are arranged in a line. From (a) to (c), image pairs (edges) are collapsed, producing I-nodes, P-nodes, and N-nodes.*

3.2 Creating Reference and Residual Images

The tree-building process determines a set of reference and residual images that we must create. Given such single-layer images, our algorithm can use any image coding method, such as DCT-based compression (e.g., JPEG) or Wavelet-based compression for the images at each node in the binary tree. For JPEG, we use the quality parameter to set the amount of quantization, and thus determine the amount of loss. We select a lower quality value for residual images, since they contain less information and thus contribute less to overall quality.

There are several ways to reduce the energy in the residuals beyond using straightforward image differencing. We want to identify pixel correspondences and use this information to lower the energy in the residuals. Loosely, we categorize existing approaches into those that infer pixel correspondences using pixel search algorithms and those that require 3D information of the scene. Motion estimation, typically used in video compression, uses search algorithms to find for a block of pixels in an image A a similar block of pixels in an image B. However, such search algorithms are computationally very expensive.

Proxy-Warped Reference Images

We reduce residual image energy using approximate 3D scene information and warping the reference image to the COP of the residual image prior to image differencing. This approach uses a geometric proxy to project a common set of 3D point features onto both image planes, establishing a feature-based correspondence. Then, the features in one image are triangulated and a projective mapping warps the

pixels of each triangle to their corresponding position in the other image. The warp operation may be performed by software or, as in our implementation, by exploiting texturing hardware that is commonly available on graphics cards.

The algorithm generates 3D features by subdividing the polygons of the proxy and using the vertices as features. In order to obtain an approximately even distribution of features in image space, the algorithm adaptively subdivides proxy polygons according to the size of their screen-space projection. Polygons near the COP are subdivided more than polygons farther away. Pre-computing and storing features for every image pair would increase the amount of data to store, so we choose to generate features on the fly.

It is worth mentioning that a ray casting approach which creates set of 2D features in a first image and finds the corresponding locations in a second image might obtain a better feature distribution, but it also introduces several disadvantages. In particular, features are not necessarily created along edges and corners of the proxy (i.e., places that often correspond to sharp visual change) and ray casting cannot easily be adapted to hardware.

Occlusion Camera Reference Images

To further reduce the size of the residuals, we need to have samples available for the surfaces that may become visible after the image warp. In general, the energy in the residual images comes from three main sources. One is due to the view dependent appearance of scene surfaces. For example, a highly specular surface will look different in images with different COPs, and although the mapping induced by the proxy correctly finds the same surface point in both images, it does not help reducing the color difference. Another source of residual difference is the approximate nature of the proxy; the small geometric features ignored by the proxy do create small inconsistencies between the warped reference image and the image to be compressed. A third and large source is due to disocclusions. The image to be compressed samples some surfaces that are not visible from the reference viewpoint. Simply using the reference image to recreate a second image produces disocclusion errors. Compensating for disocclusion errors in the

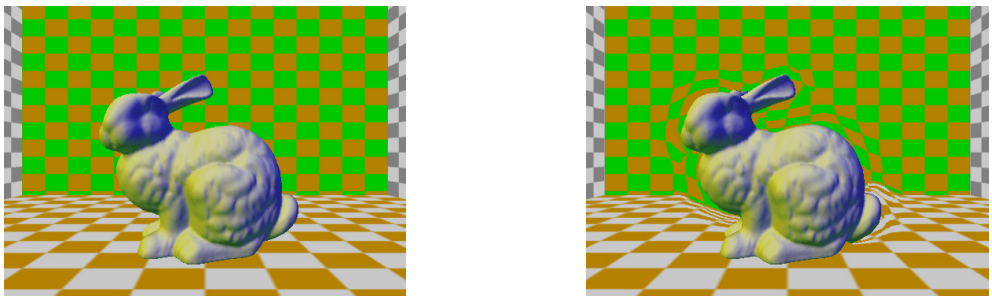

Figure 4a. *Depth image (DI, left) and occlusion camera reference image (OCRI, right). The OCRI stores additional samples of the background around the silhouette of the bunny.*

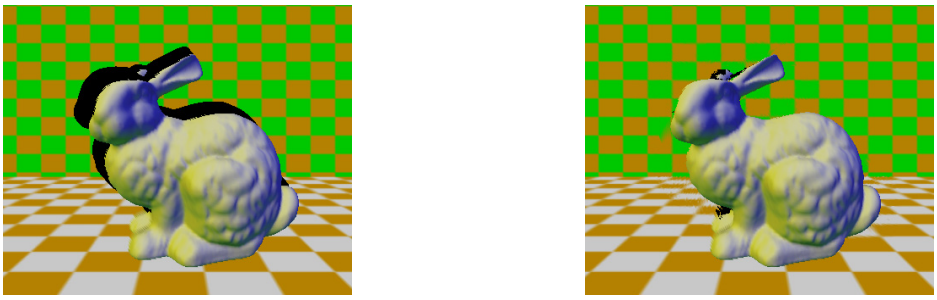

Figure 4b. *Desired image rendered from DI (left) and from OCRI (right). The OCRI image effectively alleviates disocclusion errors.*

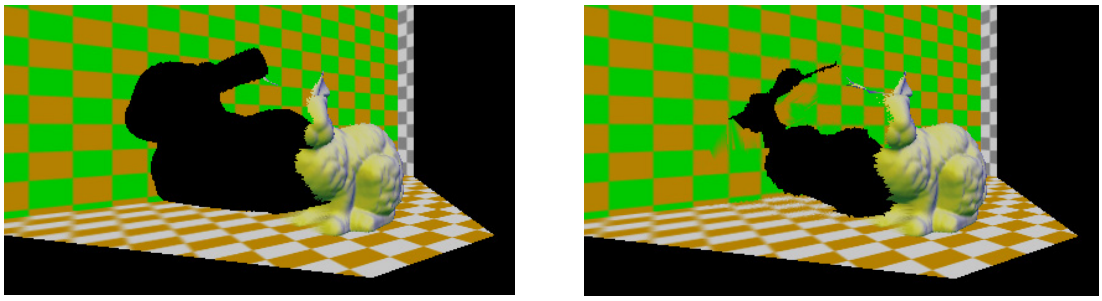

Figure 4c. *The DI "shadow" of the bunny (left) is reduced by the OCRI.*

residual image does not lend itself to good compression. Thus, we reduce the size of the residual images by alleviating disocclusion errors. Instead of regular pinhole camera reference images we use occlusion camera reference images (OCRIs), which store additional samples to prevent disocclusion errors. The OCRI has a single layer. The visible samples are "squeezed" together in the neighborhood of depth discontinuities to make room for barely hidden samples (Figure 4a and 4b). The OCRI does not store

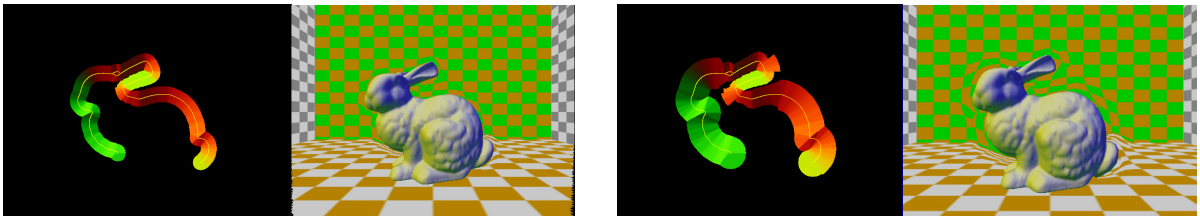

Figure 4d. *Distortion map visualization and resulting OCRI, for two splat sizes. The distortion acts only in the vicinity of depth discontinuities. The size of the splats determines how far the occlusion camera reaches behind the occluder. Larger splats disocclude more, but also reduce the resolution of the visible samples more, as a larger number of samples are squeezed in the same image area.*

samples for all surfaces; samples that are clearly hidden and are unlikely to become visible in nearby views are discarded (Figure 4c).

OCRIs are described in detail in [14]. We give here a brief overview of OCRI construction for completeness. Given a scene *S,* specified as a proxy, and a reference view using a planar-pinhole-camera $PPHC_0$, the OCRI is built by first constructing the occlusion camera OC_0 from *S* and $PPHC_0$, and then by rendering *S* with OC_0. The occlusion camera OC_0 is constructed as follows.

1. Render scene with $PPHC_0$ into z-buffer *ZB*.
2. Find depth discontinuities in *ZB*; we use a robust and efficient method based on thresholding the second-order z difference [27].
3. Splat depth discontinuity pixels into distortion map (Figure 4d).

The distortion map has the same resolution as the reference image. Each distortion map location specifies a distortion for that particular ray. Only rays that pass near a depth discontinuity are distorted. A ray is shifted perpendicularly to the depth discontinuity, and towards the occluder. This way the occlusion camera rays reach behind the occluder and reduce the size of its "shadow" (Figure 4c).

Once the occlusion camera OC_0 is known, the OCRI is built by rendering the proxy triangles with the occlusion camera. A triangle *t* with vertices V_0, V_1, V_2 is rendered in classic feed-forward fashion as follows.

1. Project each vertex V_i with OC_0 in three steps.
    a. Project V_i with $PPHC_0$ to (u_i^u, v_i^u, z_i).

b. Distort (u_i^u, v_i^u) to (u_i^d, v_i^d) using distortion map location (u_i^u, v_i^u) and depth z_i.

c. Compute distorted 3D point V_i^d as point at depth z_i along $PPHC_0$ ray (u_i^d, v_i^d).

2. Render triangle $t^d = (V_0^d, V_1^d, V_2^d)$ in hardware using $PPHC_0$.

Warping Optimization

For each image differencing operation, we optionally adjust the registration of the proxy with the images so as to minimize the energy of the residual image. This optimization attempts to reduce errors introduced by camera pose estimation as well as attempts to compensate for the approximate nature of the proxy. The optimization process uses the COP A of a first image IA and the COP B of a second image IB to initialize the vector V=B-A, containing the translation and rotation offsets from image IA to IB. We minimize the energy of the residual image between image IA warped to position A+V and image IB. A multidimensional minimization method (e.g., Powell's method [28]) is used to find an optimal vector V.

3.3 Extracting and Decoding Original Images

In an IBR application we need to extract an arbitrary image sequence from the original set of M images. We find the corresponding leaf node, and trace the links up to the I-node ancestor. Then starting with the I-node, we reverse the path following the P-node or N-node branches, adding the P-node images to the I-node until we reach the leaf-node. This procedure is applied recursively until the tree is traversed down to the node of the desired image.

There are several variants to the tree building process that yield different original-image extraction methods. We present three such methods here and explore their tradeoffs in the results section.

- Root-based Extraction: A first extraction method always refers to the root node and is the most straightforward one. It requires at most an O(logM) sequence of image additions. For this method, the root of the tree is the only I-node and all other nodes are either P-nodes or N-nodes. Since the root I-node must be accessed, the number of image additions required to extract an arbitrary image is equal to the height of the tree. Unfortunately, the accumulation of a long sequence of

incremental image residuals may result in a large reconstruction error. This problem can be mediated by creating residual images top-down instead of bottom-up, ensuring that only captured images are used to calculate image residuals. In order to create residual images top-down, we must first create a tree skeleton, that is determine the nodes types and the tree connectivity. Such a tree-skeleton is easily constructed if we use a metric for edge collapse priority that depends on the Euclidean distance between the images and not on image energy.

- Indirect I-node-based Extraction: A second extraction method reduces decompression time, at the expense of storage, by decreasing the maximum number of image additions. This option forces I-nodes to be distributed throughout the tree, akin to forced intra-coding in MPEG. This distribution may be proportional to the residual image energy (adaptive) or set by a pre-defined constant. For example, if during an edge collapse the residual image energy is greater than a pre-defined threshold or the distance between a newly created I-node and the farthest P-node descendant is greater than a pre-defined constant, we change the newly created P-node to an I-node.
- Direct I-node-based Extraction: A third image extraction method further reduces the number of image additions to exactly one in all cases (i.e. constant time) by defining the residual image of a P-node directly relative to the closest I-node up the tree. To build such a tree, we either have knowledge of which nodes will be I-nodes or make residuals after creating the tree skeleton.

4. Implementation

We have implemented our algorithms on a Pentium IV 3.0 Ghz system equipped with a modern graphics card. We use JPEG compression both for the I- and P-nodes, adjusting the quality setting to achieve the desired quantization. We use an OpenGL programming interface and the graphics subsystem to achieve image differencing and image addition at interactive rates. Both operations are performed using multiple-pass texture mapping operations.

Proxy-based reference image warping, as described in Section 3.2, is implemented using OpenGL. To determine the set of features for reconstructing an image using an I-node and P-node image pair, we recursively subdivide proxy polygons on the main CPU for the COP of either of the images. Using a low-resolution frame buffer, we render the proxy polygons into the z-buffer and then render the proxy vertices into the color buffer encoding unique vertex IDs in the color channels. Next, we read back the frame buffer and determine which vertices are visible. We compute the location of the same features in the other image. Next, we load the two images into texture memory. We render the triangular mesh for the I-node image using its feature positions in image space as texture coordinates, but render the mesh using the vertex coordinates of the P-node mesh to create the warped image. Finally, the reconstructed image is obtained by adding the signed P-node residual image to the warped I-node image.

Like in the case of a regular reference image, each pixel of an OCRI defines a 3D point with color and the regular 2D array of samples implicitly defines a triangle mesh. The OCRI is efficiently warped by rendering the OCRI mesh (or proxy) from the novel view in hardware.

The complexity of the image warp operation is proportional to the size of the images. In order to reduce the complexity, we can optionally reduce the size of the warped images. This reduction in image resolution improves runtime performance but shifts more energy into the residual image, resulting in diminished compression performance.

5. Results

We have applied our algorithms to three datasets (Table 1). For each dataset, we captured images at 1024x1024 pixels from an eye-height plane using an omnidirectional camera [29] mounted on a motorized cart driven through the environment via remote control. The first dataset, Museum, consists of 9832 images covering 900 square feet. The second dataset, Office, consists of 3475 images covering 30 square feet. And finally, the third dataset, Library, contains 1947 images covering 120 square feet. To create the proxy for each environment, we obtain a coarse floor plan of each environment and extrude the

Dataset	No. Images	Raw	Avg COP Spacing
Museum	9832	30.9 GB	2.2 inches
Office	3475	10.9 GB	0.7 inches
Library	1947	6.1 GB	1.6 inches

Table 1. Dataset Summary*. This table summarizes our datasets: number of images, raw size, and average COP spacing.*

walls to construct a 3D model of about a dozen polygons each. Floor and ceiling surfaces are approximated with two large planes. On average, we are able to extract and reconstruct from proxy-warped images 1024x1024 pixel-resolution images at a rate of 15 to 20 images per second.

5.1 Hierarchy Alternatives

The hierarchy, as described in Section 3.3, supports either residual images that are relative to their immediate ancestor in the tree (e.g, indirect I-node-based coding) or residual images that are directly relative to the closest I-node up the tree (e.g., direct I-node-based coding). The former yields better compression but more expensive image extraction. The latter provides constant image extraction cost but less compression performance. The tradeoff also depends on the spacing between I-nodes in the tree.

To better understand the aforementioned tradeoff, Figures 5(a-d) show the compression performance for various maximum I-node spacings (i.e., maximum levels of the tree between I-nodes) using proxy-warped reference images and both residual-image strategies. Since none of the compression trees has more than 32 levels, the examples at an I-node spacing of 32 are those having a single I-node for the entire dataset. The examples at I-node spacing of 0 represent trees where every node is an I-node (this configuration is effectively the same as compressing each image as an independent JPEG image). The average preprocessing time for datasets, such as those in Figure 5, is approximately 1 second per image. This includes building the tree, computing residual images, and intra-coding images.

Figures 5a and 5b show the compressed dataset sizes plotted against I-node spacings. Other compression parameters are kept constant so that the variance observed is solely due to changing I-node

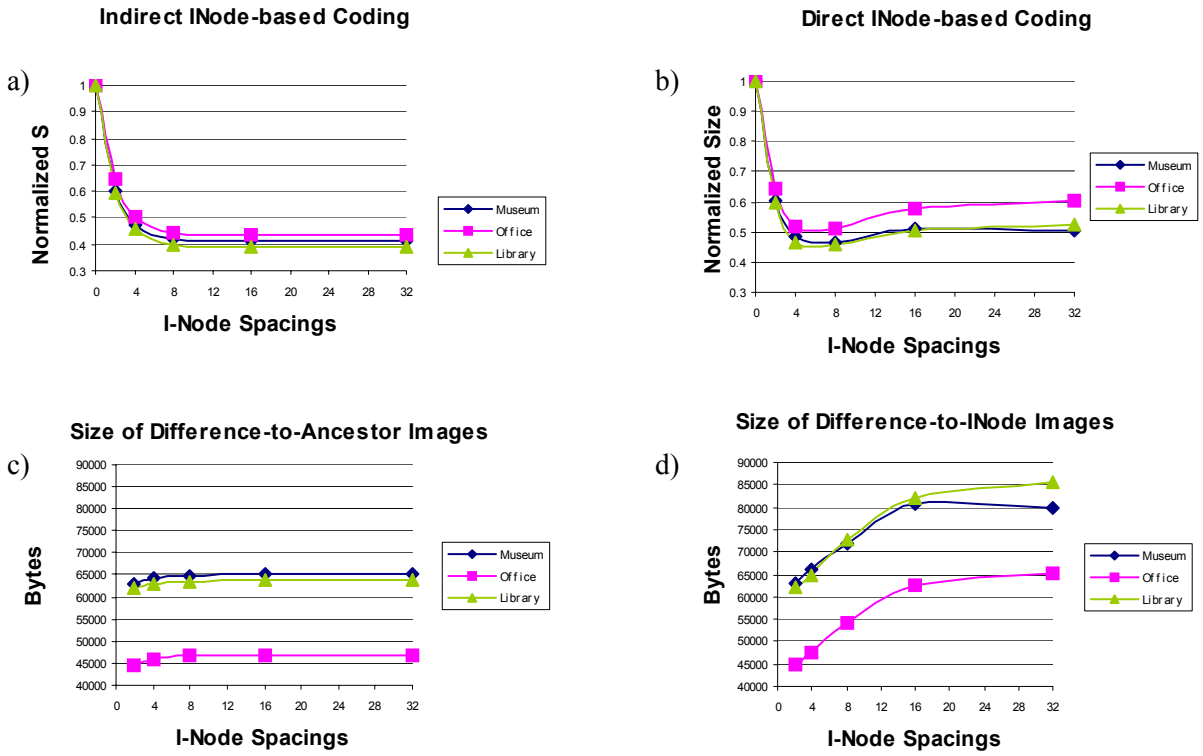

Figure 5. Comparison between residual image strategies at varying I-Node spacings. *We show the compression results for several I-node spacings. An I-node spacing of 0 corresponds to only intra-coded images. An I-node spacing of 32 corresponds to a single I-node for the entire dataset. In order to show all dataset sizes in the same graph, normalized size values are used. These values, used in (a,b), are computed by dividing the compressed dataset sizes by the maximum compressed size of the dataset in each graph. (a,c) Results using residual images computed relative to immediate ancestors (indirect I-node-based coding). (b,d) Results using residual images computed from the closest I-node up the tree (direct I-node-based coding).*

spacings. Fewer I-nodes generally yields better compression but at the expense of either quality or image extraction performance.

Figures 5c and 5d show the average compressed residual image size at each of the sampled I-node spacings. Residuals relative to their immediate ancestor tend to have about the same compressed size (Figure 5c). The image quality is inversely proportional to distance to the I-node. This is because a larger number of residuals must be used to reconstruct images, introducing more error.

However, residual images directly relative to their I-node tend to vary in size proportional to their distance from the I-node (Figure 5d). They also tend to maintain similar image quality although at potentially greater cost in terms of space. As evidenced in Figure 5b, the best tradeoff between residual image size and number of I-nodes occurs between I-node spacings of 4 and 8.

Operation	Museum		Office		Library		Average	
	Contrib	MB	Contrib	MB	Contrib	MB	**Contrib**	**Ratio**
Raw Data	0%	30929	0%	10931	0%	6128	**0%**	**1:1**
Intra-coding	35%	1559.7	35%	376.4	34%	317.9	**35%**	**23:1**
Proxy image-warping	72%	754.8	67%	195.3	72%	148.8	**70%**	**46:1**
Adaptive settings	77%	709.6	77%	169.2	78%	137.8	**77%**	**51:1**
Optimization	100%	538	100%	129.7	100%	106.6	**100%**	**66:1**

Table 2. Algorithm Analysis*. This table shows the incremental improvement in compression as different parts of the algorithm are enabled. Results in this table use residual images relative to I-Nodes and I-Node spacings of 4. The table should be read top-down. For each environment, we show compressed dataset sizes in megabytes and cumulative contributions as percentages and ratios. The percentage/ratio indicates how much of the total compression has been achieved thus far.*

In summary, I-node spacings between 4 and 8 yield a reasonable tradeoff between reconstruction time, reconstruction quality, and compression performance. Using residual images directly relative to their I-node produces a more consistent image quality although at the expense of compression. Thus, in rest of the results section we use an I-node spacing of 4 and direct I-node-based coding.

5.2 Compression Analysis

Table 2 breaks down how each part of the algorithm contributes to the total compression. As can be observed in the table, approximately one third of the total compression comes from each of intra-coding (35%), proxy-based image warping (35%), and image-warping optimization (23%). Another source of compression (approximately 7%) is adaptive quality settings for the residual images based on image energy. When intra-coding images without adaptive settings, all images are coded at fixed quality settings. In particular, we fix I-node images to be at quality setting 75 (default for JPEG) and residual images to be at quality setting 65.

To adaptively choose JPEG quality settings, we calculate the image energy range for the residuals and use interpolation to obtain the quality setting (this is similar to rate-control in video coding). The energy

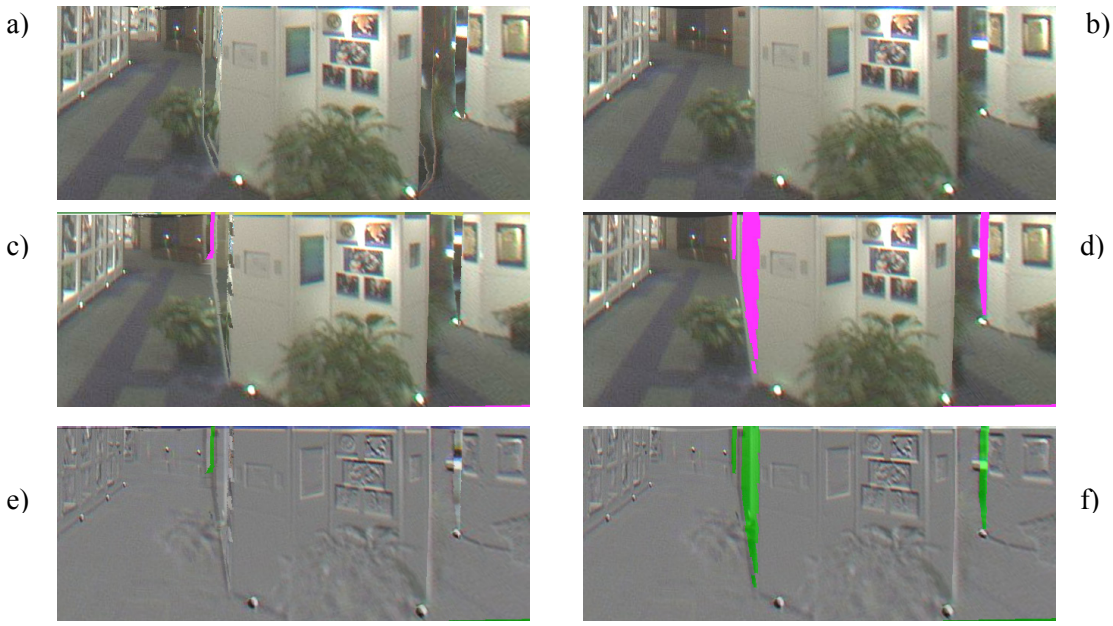

Figure 6. OCRI usage*. (a) Planar re-projection of an OCRI for the Museum environment. (b) Desired image to reconstruct. (c) OCRI warped to desired image viewpoint with the missing samples highlighted in purple. (d) Standard reference image warped to desired viewpoint with more missing samples. (e) Residual to add to warped OCRI (missing samples are highlighted in green). (f) Residual to add to proxy-warped reference image. Notice the larger quantity of missing samples that the residual image has to store.*

range is computed as the mean energy plus/minus its standard deviation. Through experimentation, we found that compressing low-energy residuals 30% more aggressively yields good results.

The error function for the image warping optimization requires computing residual images and their average energy value. Since the error function is called many times per optimization and one optimization is performed per residual, this step is computationally expensive. To reduce the preprocessing time, we use 256x256 resolution residuals during the optimization. The translation and rotation offsets are similar to those obtained using full resolution images but at reduced cost. On average, optimization using proxy-based warping adds 3 seconds to the per-image preprocessing time. Using the results of the optimization does not, however, affect the runtime decoding performance.

5.3 OCRI Benefit

Using occlusion camera reference images yields us improved compression performance over warping standard reference images whenever disocclusion errors are present. For each I-node of the tree, we create

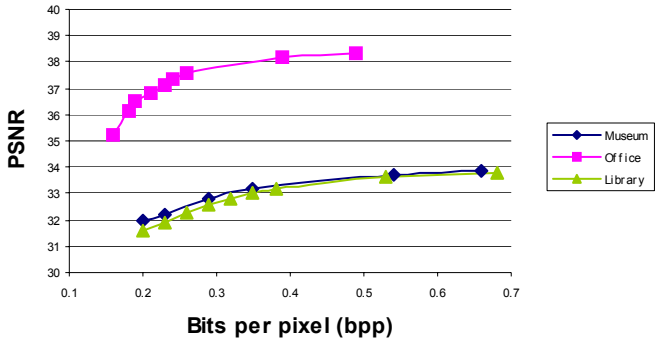

Figure 7. Compression vs. Quality*. This graphs shows the tradeoff between compression (in terms of bits-per-pixel) and quality (in terms of peak-signal-to-noise-ratio) using proxy-based warping. Original images have 24 bpp; thus, 0.68 bpp is equivalent to 35:1 compression and 0.16 bpp is equivalent to 149:1 compression.*

an OCRI by computing the distortion map using the proxy and obtain color samples from a neighborhood of images surrounding the OCRI. It takes approximate an additional 3 to 5 seconds to create each OCRI. Figure 6a shows a planar re-projection of a subset of an OCRI for the Museum environment. (Our current OCRI implementation handles only planar projections, so we tile the omnidirectional image into planar re-projections and then optionally re-assemble them). Figure 6b contains the desired destination image. Figure 6c shows how the OCRI is warped to the viewpoint of the desired image and is able to fill-in disocclusions. Figure 6d contains an image from the same viewpoint but warped using regular reference images. Figure 6e shows the residual image to be added to the OCRI in order to complete the image in Figure 6b, while Figure 6f is the residual image when warping a standard reference image. The reduced number of samples and energy in the OCRI residuals allows us to obtain improved compression performance in this example.

5.4 Coding Performance

Using the best parameter settings, as previously described, we vary the effective compression rate (bits-per-pixel, or bpp) and report in Figure 7 the peak-signal-to-noise-ratio (PSNR) for each of our datasets when using proxy-based warping. The higher PSNR values for the office environment are due to the higher image density in that environment (see Table 1). This increase in image density creates lower energy residuals.

Figure 8. Image Compression Examples*. We show images compressed by several ratios. Images (a-c) are of the Museum environment. Images (d-f) are of the office environment. Images (g-i) are of the library environment. The original omnidirectional images are shown in (a,d,g). For our datasets, we able to compress to 83:1 (b), 100:1 (e), and 69:1 (h) without significant loss in quality. Further compression slowly exhibits artifacts such as those visible at 121:1 (c), 149:1 (f) or 120:1 (i). Image (j) was reconstructed using OCRIs and the resulting more compact difference images. Image (k) highlights the area of the difference image for proxy-warping which needs to store the samples of disoccluded surfaces in image (k). Image (l) shows the OCRI difference image lacking those samples.*

a) b) c)

Figure 9. MPEG-2 Comparison*. (a) A portion of an original captured omnidirectional image within the Museum environment. (b) The same frame but reconstructed from MPEG-2 coded images such that the total compression equals 85:1. (c) The same frame reconstructed using the compression algorithm of this paper with standard reference images. Reconstruction uses difference-from-I-nodes, I-node spacing of 5, proxy-based image warping, and image warping optimization, yielding 85:1 total compression. Notice the improved quality of our reconstruction as compared to the artifacts visible in the MPEG-2 frame.*

For our datasets, we show example reconstructed-images for several compression ratios, ranging from 35:1 to 149:1, in Figure 8. On average, our algorithm can compress images without significant artifacts by a factor of 84-to-1 (average compression ratio of Figures 8b, 8e, and 8h).

For the museum environment, our largest dataset, we used OCRIs to yield improved compression. The compression performance, by definition, will be the same or better than proxy-based warping depending on the presence of disocclusions in the image sequence. For our simple proxy model, disocclusions mostly occur near the kiosk located in the middle of the museum. For a sequence of planar re-projections containing views of this subset of the model (similar to Figure 6), OCRIs are able to reduce disocclusions and image energy. Using a sequence of 2000 images facing the kiosk, compressed at low to medium compression ratios (35:1 to 83:1), we observed individual OCRI difference images that were up to 3.3% smaller and up to 4.4% more compact at higher compression rations (121:1). Figures 8(j-l) show an image reconstructed using OCRIs and a comparison of difference images. The overall average improvement we saw in images containing disocclusions was only about one percent. In our particular environment, the

missing samples were often of a single color (e.g., white colored surfaces) and thus compress well in the difference image – this offsets the maximum gains to be obtained from OCRIs in this case.

In Figure 9, we compare our compression algorithm to a standard MPEG-2 encoding of a linearization of the entire image database. First, we select an approximate target compression ratio for our algorithm (e.g., 85:1) and then select a bpp setting for an MPEG-2 encoder that yields approximately the same overall compression. Even though we are only using proxy-based warping in this example, on average, our algorithm results in better quality images because we are able to capitalize on the 2D-nature of the inter-image redundancy, as opposed to the linear (1D) inter-image redundancy in MPEG-2. Furthermore, the proxy model, image warping, and optimization process contribute to our superior quality.

6. Conclusions and Future Work

We have described a spatial image hierarchy combined with an image compression algorithm that uses image warping and exploits both inter-image and intra-image redundancy. This approach allows access of images along arbitrary viewpoint paths and provides significant overall compression. Warping using proxies and OCRI's yields us more compression, on average, than standard approaches (e.g., Figure 9). Our method can be efficiently implemented on today's graphics hardware. We have demonstrated our method using up to 10,000 high-resolution omnidirectional images and have achieved compression ratios of nearly 84-to-1 without significant quality loss.

As future work, we are investigating how to obtain better compression by tracking image features from one image to another. This correspondence information could be used to more accurately warp one image to another, thus reducing further the residual image energy and improving compression.

In addition, we could store images at varying resolutions and create a multi-resolution pixel hierarchy [30]. This would allow a runtime system to decide on-the-fly which resolution it desires. Each resolution level could be encoded independently or differentially from the previous resolution level. The latter

approach would achieve the greatest overall compression but would require a larger memory footprint for decoding.

Finally, recent NVidia graphics hardware supports hardware decoding of textures (e.g., S3TC format). This format does not provide a large amount of compression but can be decoded efficiently. It would be interesting to quantify the tradeoff between decoding speed and compression performance.

References

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

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

[3] McMillan L. and Bishop G., "Plenoptic Modeling: An Image-Based Rendering System", *Proceedings of ACM SIGGRAPH*, pp. 39-46, 1995.

[4] Aliaga D., Carlbom I., "Plenoptic Stitching: A Scalable Method for Reconstructing Interactive Walkthroughs", *Proceedings of ACM SIGGRAPH*, pp. 443-450, 2001.

[5] Aliaga D., Funkhouser T., Yanovsky D., Carlbom I., "Sea of Images: A Dense Sampling Approach for Rendering Large Indoor Environments", *IEEE Computer Graphics & Applications, Special Issue on 3D Reconstruction and Visualization*, 23(6), pp. 22-30, 2003.

[6] Buehler C., Boose M., McMillan L., Gortler S., Cohen M., "Unstructured Lumigraph Rendering", *Proceedings of ACM SIGGRAPH*, pp. 425-432, 2001.

[7] Gortler S., Grzeszczuk R., Szeliski R., and Cohen M., "The Lumigraph", *Proceedings of ACM SIGGRAPH*, pp. 43-54, 1996.

[8] Levoy M. and Hanrahan P., "Light Field Rendering", *Proceedings of ACM SIGGRAPH*, pp. 31-42, 1996.

[9] Shum H., He L., "Concentric Mosaics", *Proceedings of ACM SIGGRAPH*, pp. 299-306, 1999.

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

[11] ISO/IEC. *ISO/IEC International Standard 15444*, Final Committee Draft, March 2000.

[12] Vetterli M., Kovacevic J., "Wavelets and Subband Coding", Prentice Hall PTR, 1995.

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

[14] Popescu V., Aliaga D., "Depth Discontinuity Occlusion Camera", Proceedings of *ACM Symposium on Interactive 3D Graphics*, pp. 139-143, 2006.

[15] Aliaga D., Carlbom I, "A Spatial Hierarchy for Compression in Image-Based Rendering", *Proceedings of IEEE International Conference on Image Processing*, pp. 609-612, 2005.

[16] Gersho A., Gray R.M., *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1992.

[17] Ziv J. and Lempel A., "A universal algorithm for Sequential Data Compression", *IEEE Transactions on Information Theory*, IT-23, pp. 337-343, 1977.

[18] Magnor M., Girod B., "Data Compression for Lightfield Rendering", *IEEE Transactions on Circuits and Systems for Video Technology,* 10(3), pp. 338-343, 2000.

[19] Peter I., Strasser W., "The Wavelet Stream: Interactive Multiresolution Lightfield Rendering", *Rendering Techniques 2001: Proceedings of 12th Eurographics Workshop on Rendering*, pp. 127-138, 2001.

[20] Gotz D., Mayer-Patel K., Manocha D., "IRW: An Incremental Representation for Image-Based Walkthroughs", *Proceedings of ACM Multimedia*, pp. 67-76, 2002.

[21] Wilson A., Mayer-Patel K., Manocha D., "Spatially Encoded Far-Field Representations for Interactive Walkthroughs", *Proceedings of ACM Multimedia*, pp. 348-357, 2002.

[22] Aliaga D., Cohen J., Wilson A., Baker E., Zhang H., Erikson C., Hoff K., Hudson T., Stuerzlinger W., Bastos R., Whitton M., Brooks F., Manocha D., "MMR: An Interactive Massive Model Rendering System Using Geometric and Image-based Acceleration", *Proceedings of ACM Symposium on Interactive 3D Graphics,* pp. 199-206, 1999.

[23] Shade J., Gortler S., He L., Szeliski R., "Layered Depth Images", *Proceedings of ACM SIGGRAPH*, pp. 231-242, 1998.

[24] Popescu V., Lastra A., Aliaga D., Oliveira M., "Efficient Warping for Architectural Walkthroughs using Layered Depth Images.", *Proceedings of IEEE Visualization*, pp. 211-215, 1998.

[25] Mei C., Popescu V., Sacks E., "The Occlusion Camera", Proceedings of Eurographics 2005, *Computer Graphics Forum*, 24(3), pp. 139-143, 2005.

[26] Yu J., McMillan L., "General Linear Cameras", Proceeding of 8th European Conference on Computer Vision (ECCV), Vol. 2, pp. 14-27, 2004.

[27] Popescu V., Eyles J., Lastra A., Steinhurst J., England N., and Nyland L, *Proceedings of ACM SIGGRAPH,* pp. 433-442, 2000.

[28] Press W., Teukolsky S., Vetterling W., Flannery B., "Numerical Recipes in C", *Cambridge University Press*, Reprinted 1999.

[29] Nayar S., "Catadioptric Omnidirectional Camera", *Proceedings of IEEE Computer Vision and Pattern Recognition*, pp. 482-488, 1997.

[30] Chang C., Bishop G., and Lastra A., "LDI Tree: A Hierarchical Representation for Image-based Rendering", *Proceedings of ACM SIGGRAPH*, pp. 291-298, 1999.