A SPATIAL IMAGE HIERARCHY FOR COMPRESSION IN IMAGE-BASED- RENDERING

Daniel G. Aliaga
Department of Computer Science at Purdue University

Ingrid Carlbom
Lucent Technologies Bell Labs

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 higher 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 compression scheme based on a spatial image hierarchy that meets the requirements of interactive IBR walkthroughs. By exploiting image coherence over the entire 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 achieve interactive display rates during IBR walkthroughs.

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

INTRODUCTION

The ability to capture, store, and reconstruct large 3D environments is increasingly important for interactive walkthrough applications such as telepresence and educational tourism. In recent years, *image-based rendering* (IBR) addresses this problem by capturing large collections of images (e.g., thousands of images) and then resampling these images to create photorealistic novel views without reconstructing a detailed 3D model or simulating global illumination.

For image-based interactive walkthroughs, we must provide both quick image access and significant image compression. Image access cannot be limited to 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.

In this paper, we propose a compression algorithm based on a spatial image hierarchy that both provides quick access to images along arbitrary viewpoint paths and enables efficient compression of high-resolution images irregularly sampled over a plane. Specifically, our method arranges original, reference, and residual images into a binary tree. An image is extracted via a sequence of image warping operations. Each operation warps a reference image so that its center of projection corresponds to that of a residual image and the two are added together. Individual images are compressed using an image compression scheme. But unlike MPEG, our approach takes advantage of the inter-image redundancy over the entire capture plane, and also uses a coarse geometric model of the environment eliminating the need for computationally-expensive motion estimation.

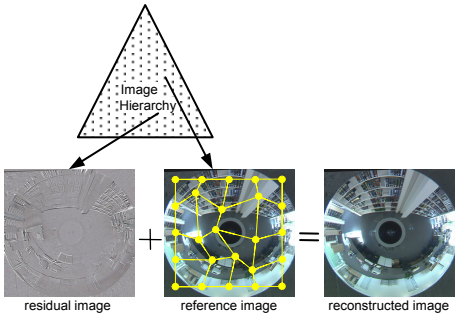

Figure 1. Image Hierarchy and Compression*. We present a compression algorithm based on a spatial image hierarchy for interactive image-based walkthroughs.*

In experiments with environments of 2000 to 10,000 omnidirectional images, 1024x1024 pixels each, our method approaches a factor of one-hundred to one compression without significant loss in quality as well as interactive decoding on today's PC hardware.

The major contributions of this work include (1) a spatial image hierarchy, (2) efficient image compression exploiting image coherence over a plane, and (3) support for quick access to images along arbitrary viewpoint paths for interactive IBR applications.

RELATED WORK AND MOTIVATION

Traditional compression methods, such as JPEG, 2D wavelets, and JPEG2000, exploit intra-image redundancy to reduce image size but they do not take advantage of inter-image redundancy. Video compression (e.g., MPEG) uses motion-estimation algorithms to exploit inter-image coherence, achieving a significant improvement in overall compression performance. However, motion-estimation is intended for linear sequences of images, making it ill-suited for image access along viewpoint paths that do not coincide with the capture paths.

Several compression schemes specific to IBR datasets have been presented in the literature. For example, the Lightfield [4] and Lumigraph [3] use naïve compression schemes based on vector quantization and Lempel-Ziv compression. More elaborate algorithms, such as that of Magnor and Girod [6], use a DCT-based coder for Lightfield datasets containing 1024 images. The images are arranged into a quad-tree and are encoded as either I- or P-frames. For display, all 256x256-pixel I-frames and residuals are decoded and stored in memory. Peter and Strasser [8] describe a wavelet-based compression algorithm for the same Lightfield datasets yielding up to 100:1 compression. A three-layer cache achieves interactive rates (15-20 fps) for 256x256 pixel images but upon too many cache misses or without the cache, performance is reduced to 3 fps. None of these systems support efficient access along arbitrary viewpoint paths.

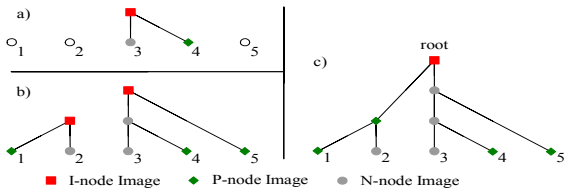

Figure 2. Tree Building. *This figure shows the process of building a small tree. From (a) to (c), image pairs (edges) are collapsed, producing I-nodes, P-nodes, and N-nodes.*

Gotz et al [4] describe an incremental compression scheme for cylindrical projection images. Their representation exploits the coherence amongst nearby images and rearranges the columns of pixels into either encoded index columns or residual columns. 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 [9] create a system for interactive walkthroughs of synthetic models. Their work partitions the model into viewing regions and uses video to represent the geometry outside each region. The scheme exploits redundancy between images of neighboring cells. The average compression ratio for their dataset of 22,000 images of 512x512 pixels is 48.

Our approach to recreating large IBR environments is the *Sea of Images* [1]. The data consists of a large number of high-resolution images of a real-world environment irregularly sampled over a plane parallel to the ground at observer eye-height. Our compression method, described in this paper, yields a high compression ratio, scales easily, and permits fast image access along arbitrary viewpoint paths. Image decoding is interactive and uses commonly available hardware. None of the systems described above support all these features.

IMAGE HIERARCHY

Tree Construction

Our algorithm implements the image hierarchy as a binary tree incrementally built bottom-up and exploits coherence between images and within each image. First, a node (with no children) is created for each image. Second, the centers-of-projection (COP) of the images are connected using Delaunay triangulation. Third, edges in the triangulation are collapsed in edge priority order. Each edge collapse gives rise to a new node in the tree whose children are the adjoining image nodes.

There are many possible metrics for edge priority. To maximize compression, edges should be processed in an order that maximizes inter-image coherence. An effective approximation of image similarity is the Euclidean distance between the images (i.e., edge length). Our experiments show that this very fast metric gives almost as good compression as more costly image energy metrics.

Using each edge and its associated node pair in priority order, we collapse the two vertices of the edge into one vertex and make both nodes children of a common parent node located at the new vertex. This operation produces three types of tree nodes: I-node, P-node, and N-node. The new 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 reference to the parent. The other child node (P-node) becomes a residual image, the difference between the original child and the I-node image. (The I-node image may be optionally warped for better compression – as is described later.) Finally, the nodes are locally re-triangulated and the queue of edges is updated. After all edges have been processed, the result is a single tree for the entire image set (Figure 2).

Image Encoding

The I-node images and residual images in the hierarchy can be coded using any image coding method, such as DCT-based compression (e.g., JPEG) or Wavelet-based compression. For I-node and P-node images, we typically use JPEG and set the quality parameter to determine the amount of loss. For residual images, there are several ways to improve compression beyond compressing simple image differences.

We identify pixel correspondences between images in a node pair and use this information to lower the energy in the residual. Unlike motion estimation, which is computationally expensive, we use a coarse polygonal model (called a "proxy") of the environment to warp one image to the viewpoint of the other prior to image differencing. The algorithm uses the proxy, consisting of a few dozen polygons, to project a common set of 3D point features onto both image planes, establishing a correspondence. Then, the features in one image are triangulated and a projective mapping warps the pixels of each triangle to their positions in the other image by using texturing hardware available on most graphics cards.

For each image differencing operation, we optionally adjust the registration of the proxy to 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 I_A and the COP B of a second image I_B to initialize the vector $V=B-A$, containing the translation and rotation offsets from image I_A to I_B. We minimize the energy of the residual image between image I_A warped to position $A+V$ and image I_B.

Image Extraction and Decoding

In an IBR application we need to extract an arbitrary image sequence from the original set of 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.

We have explored several variants of the tree building process that yields different image-extraction methods. The first and most straightforward image extraction method uses a tree of M nodes where the only I-node is the root. Thus, at most $O(logM)$ image additions (e.g., the height of the tree) are needed to extract an image, but the accumulation of a long sequence of image residuals may result in a large reconstruction error. This large error can be partially mediated by creating residual images top-down instead of bottom-up, ensuring only captured images are used to calculate residuals. Using an edge collapse priority based on Euclidean distance between images and not on image energy allows a tree skeleton to be quickly constructed in order to determine node types and tree structure. Afterwards, the node images can be created top-down.

Dataset	No. Images	Raw	Avg. Img. 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 the datasets used: number of images, raw size, and average image spacing.*

A second extraction method that reduces both decompression time and reconstruction error at the expense of storage, is to force I-nodes to be distributed throughout the tree, in a manner 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 it is determined that 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.

The 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 of a P-node directly relative to the closest I-node up the tree. To build such a tree, we must obtain a tree skeleton as with the first method.

RESULTS

We applied our algorithms to three datasets (Table 1) and performed several experiments. For each dataset, we captured 1024x1024 pixel-resolution images from at eye-height using an omnidirectional camera [7] mounted on a motorized cart driven via remote control. We achieve an average compression ratio of 84-to-1 without significant quality loss. We are able to extract and reconstruct full-resolution images at a rate of 15 to 20 images per second, on a Pentium IV 1.7 GHz processor with an NVidia Graphics card. The average preprocessing time per image is 1-1.5 seconds plus 3 seconds if warping optimization is used. Table 2 breaks down compression performance.

The hierarchy, as mentioned in Section 3.3, supports either residual images relative to their immediate ancestor in the tree or residual images directly relative to the closest I-node up the tree. The former yields better compression but more expensive image extraction. The latter provides constant image extraction cost but reduces compression performance. The tradeoff also depends on the spacing between I-nodes in the tree.

To better understand the aforementioned tradeoffs, we created several trees using I-node spacing ranging from every node (i.e., effectively a JPEG scheme) to one I-node for the entire tree and also used both residual image strategies. Figure 3 summarizes the results from using residual images directly relative to the closest I-node up the tree. Figure 3a shows the normalized size of the entire image dataset using various I-node spacings. Having a large number of I-nodes is expensive in terms of space. Having too few I-nodes causes the residual images to increase in size yet keep similar quality (see Figure 3b). The best tradeoff occurs in between I-node spacings of 4 and 8. Although not shown here, residuals relative to their immediate ancestor tend to have about the same compressed size but their quality degrades with distance to I-node. Thus, the resulting normalized size of the entire image dataset simply decreases monotonically as the I-node spacing increases.

Using the best parameter settings just described, we vary the effective compression rate (bits-per-pixel, or bpp) and report in Figure 3c the peak-signal-to-noise-ratio (PSNR) for each of our datasets. The higher PSNR values for the office environment are due to the higher image density in that environment and thus smaller residuals.

In addition, Figure 4 shows sample reconstructed images for several compression ratios, while Figure 5 compares our compression algorithm to a standard MPEG-2 encoding of a linearization of the entire image database. Our algorithm results in better quality images because we are able to capitalize on the 2D-nature of the inter-image redundancy. Furthermore, the proxy model, image warping, and optimization process contribute to our superior quality.

CONCLUSIONS AND FUTURE WORK

We have described a spatial image hierarchy combined with an image compression algorithm that exploits both inter- and intra-image redundancy. This approach allows access of images along arbitrary viewpoint paths and provides significant overall compression. The method can be efficiently implemented on today's computing hardware yielding image access at interactive rates. We have demonstrated our method using up to 10,000 high-resolution omnidirectional images and have achieved average compression ratios approaching one-hundred to one without significant loss of quality.

In the future, we would like to obtain better compression by tracking image features from one image to another and further reducing residual image energy. We look to feature globalization [2] as a potential approach to correspond distant images. Finally, we would like to quantify the tradeoff between the compression schemes available in graphics hardware and better-compression schemes implemented in software.

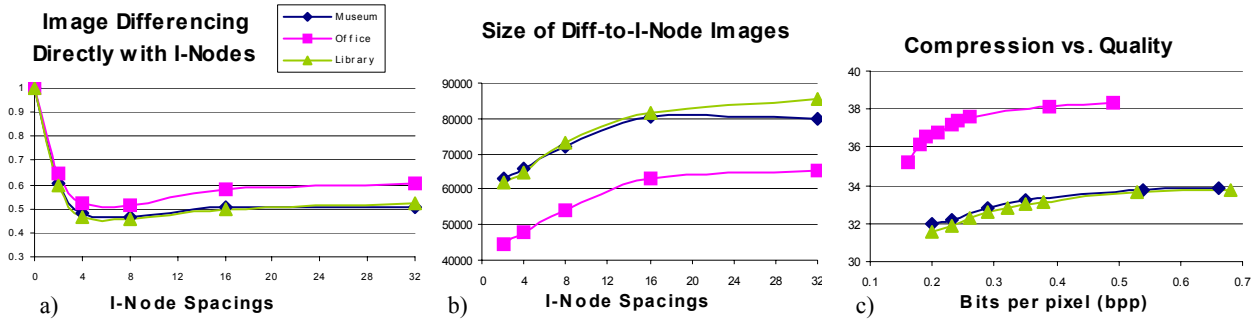

Figure 3. Example Graphs from Experiments. *We show (a) normalized dataset sizes for several I-node spacings using the image differencing directly with I-nodes (optimal setting is an I-node spacing between 4 and 8), (b) size of residual images in bytes (observe the size increases with fewer I-nodes), and (c) quality (PSNR) for several compression ratios (bpp) – original data is 24 bpp.*

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

Table 2. Algorithm Analysis. *We show the improvement in compression as parts of the algorithm are enabled. Results use residual images relative to I-Nodes and I-Node spacings of 4. Table should be read top-down. Sizes are in megabytes and cumulative contributions are percentages, indicating how much of the total compression has been achieved so far.*

Figure 4. Image Compression Examples. *We show images compressed using 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).*

Figure 5. MPEG-2 Comparison. *(a) Portion of an original captured omnidirectional image within the Museum environment. (b) Same frame but reconstructed from MPEG-2 coded images such that the total compression equals 87:1. (c) The same frame reconstructed using the compression algorithm of this paper, yielding 85:1 total compression. Notice the improved quality of our reconstruction as compared to the artifacts visible in the MPEG-2 frame.*

References

[1] Aliaga D., Funkhouser T., Yanovsky D., Carlbom I., "Sea of Images", *IEEE Visualization,* pp. 331-338, 2002.

[2] Aliaga D., Yanovsky D., Funkhouser T., Carlbom I., "Interactive Image-Based Rendering Using Feature Globalization", *ACM Symposium on Interactive 3D Graphics,* pp. 163-170, 2003.

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

[4] Gotz D., Mayer-Patel K., Manocha D., "IRW: An Incremental Representation for Image-Based Walkthroughs", *ACM Multimedia*, 2002.

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

[6] Magnor M., Girod B., "Data Compression for Lightfield Rendering", *IEEE Trans. on Circuits and Systems for Video Technology, pp.* 338-343, 2000.

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

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

[9] Wilson A., Mayer-Patel K., Manocha D., "Spatially Encoded Far-Field Representations for Interactive Walkthroughs", *ACM Multimedia*, 2002.