IRW: An Incremental Representation for Image-Based Walkthroughs

David Gotz
gotz@cs.unc.edu

Ketan Mayer-Patel
kmp@cs.unc.edu

Dinesh Manocha
dm@cs.unc.edu

University of North Carolina at Chapel Hill
CB #3175, Sitterson Hall
Chapel Hill, NC 27599 USA
http://www.cs.unc.edu/~gotz/projects/irw/

ABSTRACT

We present a new representation for image-based interactive walkthroughs. The target applications reconstruct a scene from novel viewpoints using samples from a spatial image dataset collected from a plane at eye-level. These datasets consist of pose augmented $2D$ images and often have a very large number of samples. Our representation exploits spatial coherence and rearranges the input samples as epipolar images. The base unit corresponds to a column of the original image that can be individually addressed and accessed. The overall representation, IRW, supports incremental updates, efficient encoding, scalable performance, and selective inclusion used by different reconstruction algorithms. We demonstrate the performance of our representation on a synthetic as well as a real-world environment.

1. INTRODUCTION

The ability to represent and visualize large, sampled, $3D$ datasets is becoming increasingly important in a variety of applications. These include virtual environments, computer-aided design, education and virtual tourism. Recent advances in acquisition and modeling technologies have resulted in large databases of real-world and synthetic environments. The resulting datasets may include millions, and in some cases, billions of primitives or samples. Their enormous size poses a number of challenges in terms of efficient representation, manipulation and interactive display.

In this paper, we mainly deal with interactive walkthrough applications which give the users a sense of immersion in an environment. Our goal is to handle large synthetic environments created using modeling systems as well as real-world scenes captured using camera-based acquisition systems. Given the complexity of large synthetic environments, many techniques for interactive display pre-compute a set of images or impostors from different viewpoints in the scene [3, 19, 37]. In both cases, the resulting datasets consist of a large number of images, augmented with pose information, which relates these images to each other spatially. Given the viewer's position at runtime, the visualization system uses IBR (image-based rendering) techniques to generate new images from the existing set of samples. In particular, it generates novel views by resampling a set of images of the environment, without relying upon an explicit geometric model.

Most of the work in image-based rendering has dealt with creating an appropriate subset of the complete $7D$ plenoptic function. These include techniques for $5D$ plenoptic functions [6, 15, 22], $4D$ plenoptic functions [10, 18], inside-looking-out $3D$ plenoptic functions like concentric mosaics [28], other $3D$ plenoptic functions [31], $2D$ plenoptic functions [7, 30] or using a sequence of video streams [32]. Different algorithms and techniques vary based on the restrictions on the environment (e.g. unobstructed spaces) or the underlying acquisition methods or constraints on the viewer's motion. However, none of these representations and reconstruction algorithms are directly applicable to image-based walkthroughs of large and complex environments.

Some of the key issues in generating image samples relate to the sampling density or using portions of a given sample for reconstruction from a novel viewpoint. Moreover, no robust or automatic algorithms are known that can tightly bound the reconstruction error and guarantee the fidelity of the new image. As a result, some techniques generate new samples in an adaptive manner and incrementally add these samples to the database. Furthermore, different samples are treated in a non-uniform manner by the reconstruction algorithm. Some of the commonly used encoding techniques used in JPEG and MPEG standards do not provide capabilities to support this functionality. There is relatively little work on efficiently representing large collection of spatially-augmented image samples for walkthroughs.

1.1 Main Results

We present an efficient and incremental representation of image-samples used for interactive walkthroughs. We call our representation IRW, an 'Incremental Representation for Walkthroughs'. Our formulation takes into account the characteristics of the application, where the user's motion is constrained to a plane at eye-level and allows for translation and yaw-rotation. The application reconstructs a $4D$ plenoptic function from a set of images augmented with camera pose information.

Given $2D$ image samples, we reorganize them based on the camera pose to take advantage of spatial coherence in object space. The base unit in IRW corresponds to a single column in each image that can be individually addressed and accessed. We calculate the *epipolar coordinates* for each column and group them into *epipolar images*. Within this image, each column is represented using a

one-dimensional wavelet. Our representation, IRW, supports:

- **Incremental Updates:** New samples are easily added without any major recoding or reorganization of existing data.
- **Efficient Encoding:** Compression rates are comparable to previous methods like JPEG.
- **Scalable Performance:** Compression rates improve as the database grows in size. At the same time, query times remain relatively constant.
- **Reconstruction Algorithm Independence:** The representation is independent from the choice of reconstruction algorithm.
- **Selective Inclusion:** When coupled with a reconstruction algorithm to measure a sample's utility, our representation stores individual samples only when they improve the quality of reconstruction.

We have implemented a prototype of IRW and demonstrate its performance on both synthetic and real-world spatial image data. Moreover, we compare its performance with JPEG.

1.2 Organization

The rest of the paper is organized in the following manner. In Section 2, we provide background information and related work. Section 3 presents an overview of our approach. Section 4 outlines the epipolar coherence exploited by our representation while Section 5 details the IRW representation itself. In Section 6 we discuss different queries that can be performed based on our representation. In Section 7 we present experimental results obtained from our prototype implementation. We conclude the paper in Section 8 along with a discussion of future work.

2. BACKGROUND AND RELATED WORK

This section provides background on the plenoptic function, image representations as a sampling of the plenoptic function, and a class of applications that use images to reconstruct the plenoptic function for novel viewpoints. These concepts provide a general context for our work which concentrates on the design of an image representation that is congruent with the needs of walkthrough applications. We also review previous work on different IBR applications and representations, and efficient techniques to encode them.

2.1 The Plenoptic Function

The plenoptic function in its most general form describes all light that moves through a scene. The $7D$ plenoptic function, as introduced by Adelson and Bergen [1], describes the light intensity passing through every viewpoint, in every direction, for all time, and for every wavelength. Formally, the plenoptic function is defined as: $p(x, y, z, \theta, \phi, \lambda, t)$, where [$x$, y, z] determine the position within the scene, [θ, ϕ] represent the orientation, λ corresponds to the particular wavelength of light, and t represents the time.
More intuitively, we can consider a form of the plenoptic function where the intensity of all visible wavelengths for a particular position and orientation is mapped into the YUV color space. This formulation can then be expressed as:

$$[y, u, v] = p(x, y, z, \theta, \phi, t)$$

Within the framework of the plenoptic function, an image is simply a discrete sampling of this function at a particular time. Furthermore, a video sequence can be regarded as a sampling of the plenoptic function at regular spaced time intervals along a moving camera path (i.e., x, y, z, θ, and ϕ are functions of t). Compact

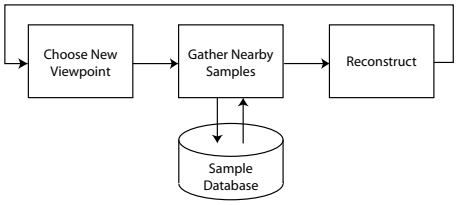

Figure 1: *A typical IBR application starts by selecting a new viewpoint from which to render the scene. This is most often done by user control. The application then queries a sample database for samples of the scene from nearby the new viewpoint. These samples are then used to reconstruct the scene from the novel vantage point.*

(i.e., compressed) representations of still and moving images rely on the coherence of the plenoptic function across its parameters.

For example, many still and moving image representations use a block-based discrete cosine transform (DCT) [26]. Typically, an 8×8 block of pixels is transformed using the 2D-DCT. Essentially, these 64 pixels are samples of the plenoptic function from a particular point in space (i.e., the coordinates of the camera position) and a range of viewing angles. The DCT provides compression by exploiting the coherence of the plenoptic function over this viewing angle range. Similarly, video representations that rely on motion compensation and differential encoding exploit the coherence of the plenoptic function across different parameters (i.e., position, viewing angle, and time).

A number of techniques have been developed for exploiting different kinds of coherence within various parameters of the plenoptic function. These include the DCT, wavelets [34], differential encoding, conditional replenishment, and motion compensation. In practice, these techniques constitute a toolbox from which specific representations can be formed. Some specific representations (e.g. JPEG [24], JPEG2000 [14], MPEG [12, 13], H.261 [33], etc.) for still and moving images apply these techniques in combination with more general compression techniques that are not specific to the structure of the plenoptic function (e.g., entropy coding).

2.2 IBR Applications

Image-based rendering applications, or *IBR* applications, use discretized samples or images in conjunction with spatial information in order to construct virtual views of the environment from new camera positions for which no image exists. Typically, the application has access to a database of acquired image samples, which may be acquired using cameras in the real-world or are generated by rasterizing the geometric model in the case of a synthetic environment. These images are associated with spatial information such that the camera viewpoint and orientation are known. The exact precision of the spatial information depends on the details of the acquisition process. The goal of IBR application is to synthesize novel views not present in the database. Figure 1 shows a block diagram model of such an application.

In this paper, we restrict ourselves to databases composed of $2D$ images of real or synthetic worlds that have been augmented with spatial information. Besides $2D$ image samples, other commonly used IBR representations include point samples [11, 25], textured depth meshes [2, 8, 29], multi-mesh impostors [9], range images [21] and layered-depth images [27]. However, these representations have been mainly applied to synthetic datasets only, as there has been relatively little progress towards an automatic technique for capturing depth of large real-world scenes.

Many image-based rendering algorithms are based on $2D$ image samples and the resulting reconstruction algorithms generate a lower dimensional approximation of the plenoptic function from these samples. In [17], a regular array of camera images are used as samples for reconstructing a $4D$ plenoptic function for a scene. They use a "dual-plane" parameterization of the plenoptic function. A similar dual-plane parameterization of a $4D$ plenoptic function is used in [10], and is combined with a low-fidelity geometric representation of surfaces in the scene to select the appropriate depth of focus on a per pixel basis. The concentric mosaics [28] approach captures an inside-looking-out $3D$ plenoptic function by constraining the camera to a planar concentric circle.

Image-based walkthrough applications attempt to synthesize new viewpoints of a real or synthetic environment using images acquired from a single eye-level horizontal plane of the environment. Furthermore, the view planes of the images are perpendicular to this plane. The resulting image samples do not form a regular pattern, and as a result techniques based on the dual-plane parameterization of $4D$ parameterization or restricting the camera's path are not directly applicable. More recently, Aliaga and Carlbom have presented plenoptic stitching [4], a parameterization of the $4D$ plenoptic function for interactive walkthroughs as well as a reconstruction algorithm based on $2D$ image-samples. Our incremental representation, IRW, can be used to represent the resulting samples as well as reconstructions from new viewpoints.

2.3 Representations for IBR Applications

Many techniques have been used for compressing the dual-plane parameterizations of $4D$ plenoptic functions. These include vector quantization and entropy coding for light-fields [18], wavelet basis [16] and block-based DCT encoders [23]. Gortler et al. [10] have treated lumigraphs as arrays of $2D$ images and proposed using JPEG and MPEG for intra-frame and inter-frame compression, respectively. Other algorithms include model-based coders for view-dependent texture mapping [20]. More recently, Wilson et al. have presented techniques to spatially encode impostors for interactive walkthrough applications by using a modified MPEG-2 encoder [36, 37]. In [38], a representation for concentric mosaics is proposed in which image data is realigned to improve image-to-image coherence and thus improve the performance of a $3D$ wavelet coder.

A fundamental drawback associated with all of these previous efforts is that they assume all image samples used for reconstruction are available in advance. In other words, these representations are not designed for incrementally adding new image samples to an existing set. To do so, the entire image database must be reorganized and re-encoded. Furthermore, most of these schemes are unable to distinguish which portions of a new image sample are actually useful for reconstruction and avoid representing those portions which are not. IRW is constructed with these capabilities as primary design goals.

3. DESIGN GOALS AND OVERVIEW

This section outlines design goals for an image representation for IBR applications and provides a high-level overview of our approach. We examine the goals and assumptions of image-based rendering applications in order to distill an appropriate set of representational design goals and show how currently available representations fall short. Finally, we outline our approach which has been specifically designed to achieve these design goals.

3.1 The Importance of Representation

A representation for the plenoptic function reflects assumptions about the nature of the coherence within the function as well as the manner in which an application uses the data. These representational choices are important for ensuring that the data is organized and stored congruently with how the data is accessed and used. For example, JPEG2000 uses a multi-resolutional wavelet to support progressive image refinement for Web applications. Similarly, the MPEG standard exploits motion compensation but limits its use to ensure that errors are not propagated in the presence of loss in a streaming application. Even the 8×8 block size most commonly used for the DCT is the combined result of the likely extent of spatial coherence and the construction of fast algorithms for calculating the transform with those dimensions.

3.2 Image-Based Walkthroughs

Image-based walkthroughs have a number of characteristics that govern their requirements for an image representation.

- **Different portions of different images are required by the application to synthesize a novel viewpoint.** In general, the reconstruction algorithm will not make use of all of the image data within a sample image at the same time. Instead, only a portion of a sample image is used to reconstruct a portion of the novel viewpoint. The required portion is determined by the spatial relationships between the novel viewpoint and the previously acquired image samples.
- **The quality of the reconstruction is generally related to the density of the image sampling.** A dense image sampling reduces the distance between the novel viewpoint being constructed and the samples used for reconstruction. Thus, the walkthrough application improves the reconstruction quality over time by adding more image samples to the available database. In other words, the image sample database may not be static but instead grows as more samples become available.
- **Denser sampling in portions of the environment in which the quality of reconstruction is already high is not helpful.** Even as more samples are acquired, not all of the sampled data are equally useful. Additional samples of the plenoptic function that are already well approximated by the reconstruction algorithm may not be worth the additional storage and management cost incurred by adding the data to the database.

3.3 Representation Design Goals

Given these characteristics of image-based walkthrough applications, we present a set of design goals for an appropriate image representation.

- **Fine-grained access to image data.** Because the application generates reconstructions from small parts of many image samples, the representation should provide efficient fine-grained access to sub-regions of the images.
- **Incremental.** Once an image database is constructed, we should be able to include new samples easily without any major re-encoding or reorganization of existing sample data.
- **Fine-grained selective inclusion.** Just as we need to access only portions of an image sample, we also need the ability to ignore portions of an image sample that may not provide any new reconstructive capability.

In addition to these application-specific design goals, efficiency is a design goal for any image representation and is generally measured in terms of compression ratio for a given quality. However, some representation techniques that result in higher compression ratios may simultaneously work against the application-level goals. A good representation design must find the right balance between these often competing requirements.

Current image and video representations do not represent a good balance between efficiency and the application-specific goals of IBR reconstruction. Most still image representations (e.g., JPEG, JPEG2000, GIF, PNG, etc.) provide little to no support for addressing only a portion of the image. For example, even though JPEG organizes pixels into 8×8 blocks, they cannot be individually decoded. Even if these blocks were individually addressable, the 8×8 block structure is rigid and does not match the needs of walkthrough applications. Although JPEG2000 provides support for organizing picture data in addressable tiles of arbitrary rectilinear dimension, there are no mechanisms for exploiting coherence between the tiles and/or between the images. This capability is necessary for our applications because the dimensions of the image portions used in reconstruction is often quite small. Thus, the representation is less efficient if it only exploits coherence within this addressable unit (as JPEG does with the 8×8 block and JPEG2000 does within the user-specified tiling).

Video representations, on the other hand, include techniques to exploit inter-image coherence, but in a very specific and limited manner. These representations rely on the implicit assumption that the images can be completely ordered such that image coherence is maximized for frames adjacent in the ordering. This ordering is provided by the temporal timestamp associated with each frame. In other words, video representations rely on the fact that two frames close in time are also close in terms of camera position and orientation. Furthermore, these representations assume that access to frames occurs along this temporal dimension and are optimized for forward playback of frames. The image data sets used for walkthrough applications, however, are primarily accessed by their spatial relationships and not temporal ones. Furthermore, access to these frames is driven by the virtual camera path of the reconstruction which is unknown in advance and is not related to any camera path associated with the acquisition of the images.

3.4 Overview of Our Approach

In this section, we describe our design of an image representation for image-based walkthrough applications. In particular, we assume that the reconstruction algorithm makes use of a set of images taken from an environment such that the camera viewpoints all lie on the same plane and that this plane is orthogonal to the image plane of the camera. Furthermore, we assume the spatial location of the camera within the environment is known. This is typically the case in image-based walkthroughs of real and synthetic environments [3, 4, 19].

In our representation, IRW, a single column from a single image is the base unit which can be individually addressed and accessed. For each image in the data set, we calculate the *epipolar coordinates* of each column. The epipolar coordinate system indexes columns first based on the view angle, then on position within the environment along an axis orthogonal to the view angle, and finally on position within the environment congruent with the view angle. The epipolar coordinate system and the mapping of columns is described in more detail in Section 5.1. Columns which share the first two indices but vary in the third index are grouped into *epipolar images*. Within this image, each column is represented using a one-dimensional wavelet. Some columns are encoded as *index columns* where the wavelet is applied to the original pixel values. Other columns are encoded as *differential columns* where the wavelet is applied to the difference between the column and the nearest index column.

Our representation has a number of useful features including:

- **Selective inclusion.** When a new sample image is added to the representation, each column is mapped into the epipolar coordinate system independently and a decision whether to include the sample column in the database can be made on a column-by-column basis. In particular, if the IBR reconstruction algorithm can already reconstruct the column in question within some specified error bound, the column is not included. Thus, view angles in the environment which are not visually complex can be efficiently represented by only a few columns, obviating the need to store additional sample data provided by images acquired later.
- **Selective access.** Each column is individually addressable and accessible. The representation guarantees that any given column will depend on at most only one other column, providing a bounded access time.
- **Highly incremental.** New sample images acquired after a representation for an existing set of images can be added easily with little re-encoding of columns already represented in the database.
- **Epipolar coherence.** Columns are grouped by view angle and then differentially encoded against columns with similar view angles along a slice of the epipolar coordinate system. This ensures that the visual structure of adjacent columns is highly coherent. Furthermore, a simple linear stretch and offset can be used to map the index column (i.e., the predictive base) to a nearby column, increasing the efficacy of the differential encoding.

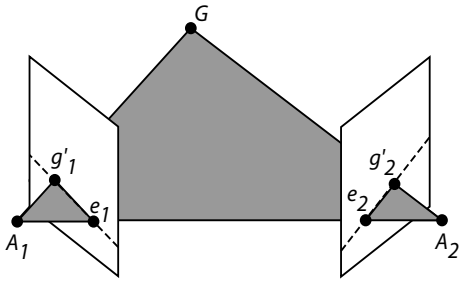

Figure 2: *Epipolar geometry describes the relationship between a pair of perspective cameras, labeled as A_1 and A_2, that both observe a common point G. The epipolar plane contains these three points and is shaded gray. The baseline connects A_1 and A_2. The Epipolar e_1 and e_2 are located where the baseline intersects the image plane for each camera. Points g_1' and g_2' are the projections of point G into the two image planes. $g_1'e_1$ and $g_2'e_2$ are epipolar lines.*

4. EPIPOLAR COHERENCE

Before presenting the detailed design of the IRW representation, we discuss *Epipolar Plane Images* (EPIs) [5] and their implications with respect to our work. EPIs have been extensively used in the computer vision community in their analysis of structure from motion. Before defining EPIs, we first present a brief overview of *epipolar geometry*.

4.1 Epipolar Geometry

Epipolar geometry describes the basic geometric relationship between a pair of perspective cameras. Suppose we have two cameras A_1 and A_2, both of which observe a common point in space G, as shown in Figure 2. The projection of point G onto the image planes is shown as g_1' and g_2' respectively. The line connecting A_1 and A_2 is the *baseline*. The points e_1 and e_2, where the baseline intersects the image planes for the two cameras, are the *Epipolar*. The plane

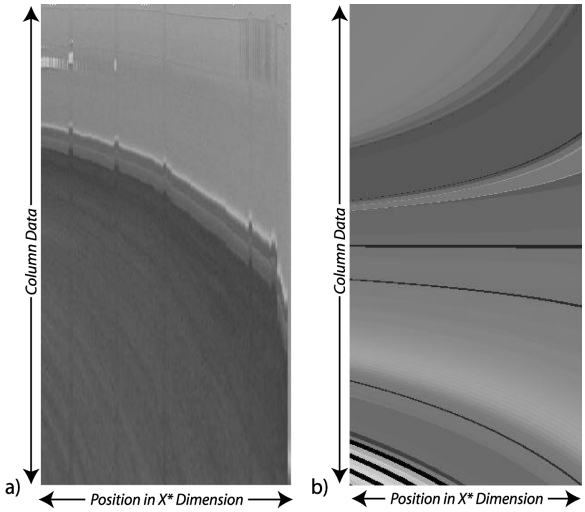

Figure 3: *These two images are EPIs created by extracting one column for each image in a sequence captured by moving the camera in the view direction and arranging them in sorted order. (a) is from a real office scene and (b) is from a synthetic model.*

defined by the three points A_1, A_2, and G is known as the *epipolar plane*. The intersection of the epipolar plane with the image plane forms the *epipolar line*. For a point G and camera A_i, the epipolar line is also defined by the two points g_i' and e_i.

As the point G moves through space, the epipolar plane rotates about the baseline, forming a family of planes known as the *epipolar pencil*. Note that when A_1, A_2, and G are collinear, there are an infinite number of valid epipolar planes, all containing the camera positions and the point G.

4.2 Epipolar Plane Images

Consider the case where a camera moves linearly along the baseline, recording images at various camera positions A_i along the path. We assume that the camera always observes point G. We also assume that G is not collinear with the camera positions A_i. We form an EPI by extracting the epipolar line from each camera A_i, and arranging them in sorted order to form a new image. Because the EPI is formed of epipolar lines that contain the point g_i', the paths traced out by points in the EPI provide insight into the structure of the scene and camera motion. For example, if the camera motion is perpendicular to the view direction, the point g_i' will trace out a line in the EPI whose slope is proportional to the depth of the point.

In this paper, we restrict ourselves to the case where the camera motion is in the same direction as the view direction. In this case, if we take the center column of each image, we can form an EPI as shown in Figure 3. The shapes of the paths seen in the EPI correspond to the perspective effects on features in the scene. Feature paths flow upwards or downwards as the camera moves closer to the scene objects. The feature paths approach a vanishing point as the camera moves further away. Figure 4 illustrates this effect with three cameras. The EPIs exhibit a very high degree of coherence from column to column due to spatial coherence in object space. Regions in the EPI disappear and appear due to occlusion and disocclusion events, respectively.

4.3 Reorganization of Image Data

The IRW representation takes advantage of the inherent coherence of spatial data sets by rearranging the image data, column by column, into EPIs. Because we assume that each image is augmented with pose information, the original images can be viewed as a collection of columns, all captured from the same position in world space, but with different camera orientations. We regroup the columns into EPIs, collections of columns captured along the same baseline with the same orientation. This reorganization transforms our dataset from a collection of traditional images to a collection of EPIs.

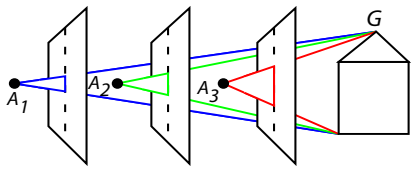

Figure 4: *This figure shows three camera positions (A_1, A_2 and A_3) arranged linearly in the view direction. As the camera position approaches G, the projection of the house onto the image plane increases in size. As the camera moves toward the house, the projection of G will move upwards until it leaves the image plane. As the camera moves away from G, the projection of G will approach a vanishing point. Arranging the epipolar lines shown by dotted lines together into a single image forms an EPI.*

5. IRW REPRESENTATION

This section presents the detailed design of the IRW representation. We start by describing the algorithm that groups the columns into EPIs. We detail the encoding algorithm and an present an important encoding invariant. Finally, we describe how our representation supports incremental updates.

5.1 Epipolar Coordinate System

We assume that we know the position and orientation of the camera used to capture each column. As mentioned in Section 3.4, the camera position is restricted to a plane at eye-level and the orientation has one degree of freedom: the angle of rotation about the Z axis. We can therefore represent the pose for column C_j as a three dimensional quantity $P_j = (x_j, y_j, \theta_j)$.

Another way to reference each column is to store the angle of rotation θ_j, as well as the (x_j, y_j) values rotated about the origin by $-\theta_j$, which we note by (x_j^*, y_j^*).

We refer to this alternate pose value as $P_j^* = (\theta_j, x_j^*, y_j^*)$. This form is useful because it allows us to easily group columns together to form a *slice*. A slice is the set of columns that share the same (θ_j, y_j^*) values. See Figure 5 for an illustration of this process. Columns within a slice are sorted along the X^* dimension. When the columns in a slice are viewed as an image, they are equivalent to an EPI formed by a camera moving in the view direction. By encoding columns grouped into slices, we can exploit the coherence found in the EPI images discussed in Section 4.2.

All three dimensions of the epipolar coordinate system are continuous. In order to group columns into slices, we discretize the space. The θ dimension, ranging from 0 to 2π, is split into n_θ discrete steps. Similarly, the Y^* dimension is split into n_{Y^*} discrete steps. Because this dimension is a rotation of the traditional y-axis, its range depends on the radius r of a bounding circle around all camera positions and centered at the origin. The range varies from $-r$ to r. The X^* dimension is bounded by the same range and can be represented at any fixed precision.

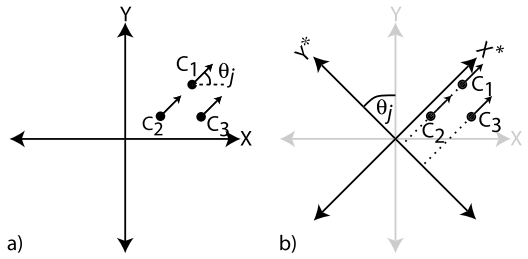

Figure 5: *Three columns, C_1, C_2, and C_3, are taken from different camera positions. Part (a) depicts a traditional coordinate system. The pose for each column C_j is (x_j, y_j, θ_j). Part (b) shows the same columns in an epipolar coordinate system. The old coordinate system is drawn in light gray. In this new coordinate system, C_1 and C_2 have the same value along the Y^* axis, placing them in the same slice. C_3 has a different Y^* value and belongs to a different slice.*

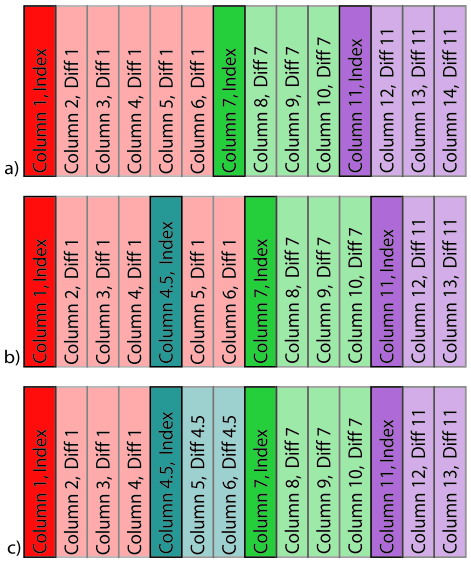

Figure 6: *(a) The database before an insertion. It complies with the encoding invariant. (b) The database after a new index column has been inserted at position 4.5. Note that columns 5 and 6 are now in violation of the invariant. (c) The system re-encodes all columns between the new index column and the following index column, Column 7. The database again complies with the encoding invariant.*

5.2 Encoding

Once columns have been grouped into slices, each slice is encoded independently. The columns are sorted by their position along the X^* axis in the epipolar coordinate system. Each column is then stored as either an index column or a difference column. Index columns encode the full color data for the column. Difference columns are predicted from nearby index columns and encode the difference between the actual color data and the predicted values. A pointer to the predictor for each column is stored in a separate structure, along with other data needed for decoding.

5.2.1 Index Columns

For index columns, we transform the raw image data using a one dimensional wavelet. For our prototype we used the irreversible $1D$ filter outlined in Appendix F of the JPEG 2000 standard [14]. The resulting coefficients are then quantized to a fixed number of bits and coefficients below a threshold are clamped to zero. Then, the quantized coefficients and run lengths of zeros are encoded using a Huffman code. The number of quantization bits and the threshold are both used to increase the compression rate at the cost of image quality.

5.2.2 Difference Columns

Consider an index column C_i and a difference column C_j. Because of the structure found in EPIs, we can assume that if the two columns are nearby, they are very similar. Even more, we can assume that a scaled and shifted version of C_i can be found that is nearly identical to C_j. In fact, if the scene being observed was a flat surface, an identical match could be found. We denote the scale factor as α, the offset as δ, and the transformed column as $W(C_i, \alpha, \delta)$, where W is a linear warping function. We need to search for the values for α and δ that best predict C_j. Note that if we enforce that $x_i^* < x_j^*$ we can limit the search by ensuring that $\alpha > 1$. This is because as the camera nears the scene objects, the features move upwards or downwards in the EPI, just as objects in an image get larger as a camera moves forward.

The encoding process starts by finding the index column C_i that is closest to C_j and that satisfies $x_i^* < x_j^*$. A search is performed to find the α and δ that minimize the error between $W(C_i, \alpha, \delta)$ and C_j. Any error metric may be used, however our prototype measures the error by calculating the sum of the absolute difference between the columns (i.e., L_1 norm).

The warped index column is then subtracted from C_j, and the difference is transformed using the same $1D$ wavelet used for index columns. Next, coefficients below a threshold are clamped to zero. We call this threshold the *zero-cutoff threshold*. The remaining non-zero coefficients are quantized to a fixed number of bits. The quantized coefficients and zero run lengths are then encoded using a Huffman code. Column metadata (α, δ, and a pointer to the index column) is stored in a separate structure.

5.3 Encoding Invariant

Our encoding algorithm maintains an invariant that is enforced after each change to the database. At all times, columns are stored in a sorted order along the X^* axis of the epipolar coordinate system. The column with the least x_j^* value is always encoded as an index column. Subsequent columns all refer to the same index column until the next index column is reached. In order to support incremental updates to our representation, there is no fixed distance between the index columns. Index columns are added as needed as described in Section 5.4. Once a new index column has been reached, all difference columns are encoded using the new index column and none are encoded using any earlier index columns. This prevents *index fragmentation* (fragmentation of the index/difference relationship) and limits search times during incremental updates to the database (see Section 5.4). It also ensures that nearby columns reference the same index column, reducing the amount of work needed to decode the data. Figure 6 shows a graphic illustration of the invariant.

5.4 Incremental Updates

The IRW representation is incrementally updated as new samples become available. This is possible because the addition of a new column requires changes to only a small neighborhood of samples. When a column is added to the database, the first step is to compute a *candidate index column*. The candidate is used to encode the new

column as a difference column. The system begins by finding the closest column already in the database whose x_j^* value is lower than that of the new column. If this closest column is an index column, it is used as the candidate. If it is a difference column, we load its associated index column and use that as the candidate index column.

A search is performed to find values for α and δ that best match the new column with the candidate index column. The candidate index column is then warped by the best α and δ values and subtracted from the new column. The difference data is transformed by a wavelet and the coefficients are quantized to a fixed number of bits. Low coefficients are rounded to zero. At this point, we perform an evaluation to determine whether the difference data is sufficient to store in the database or whether the new column should be saved as an index column. For our prototype implementation, we performed this evaluation by counting the number of zero coefficients. If the number is above a threshold, the difference is encoded using a Huffman code and stored in the database along with the new α and δ values.

If the number of zeros is below the threshold, the new column is fully encoded as an index column. However, simply adding the new index column violates the invariant presented in Section 5.3. To reinforce the invariant, the system must alter all difference columns encoded using the candidate index column whose position along the X^* axis is greater than that of the newly inserted index column. Each of these columns must be re-encoded using the newly inserted index column. Once these rows have been fixed, the invariant will again be true.

Figure 6 illustrates this process. Part (a) shows the initial database state. Note that the first column is an index column, along with columns 7 and 11. The encoding invariant holds and there is no index fragmentation. Part (b) shows the database after a new index column has been inserted at position 4.5. This insertion places the difference columns 5 and 6 in violation of the invariant because they are encoded using column 1. The algorithm must re-encode all difference columns between the new index column (4.5) and the following index column (7). Part (c) shows the database after columns 5 and 6 have been re-encoded using the new index column. The index fragmentation has been repaired and the encoding invariant is reinforced.

5.5 Independence from Reconstruction

The basic IRW representation is independent from the reconstruction algorithm used by the target application. Any reconstruction algorithm from the appropriate sub-class of IBR applications is compatible with IRW. For example, an application may simply render the nearest sample or linearly interpolate between the two nearest samples. More sophisticated algorithms, such as Plenoptic Stitching [4], can also be used. Each of these algorithms begins reconstruction by querying the IRW database for the appropriate image samples. The reconstruction then proceeds independent of the representation.

6. QUERYING THE COLUMN DATA

An important factor in the design of the IRW representation is the need to support very fast queries. Many applications that could take advantage of our representation are interactive and require fast access to needed image data.

An application initiates a query for column C_j by providing the epipolar coordinate P_j^*. As discussed in Section 5.1, the space is discretized at a fixed resolution. We can therefore determine which slice contains the desired column from P_j^* in $O(1)$ time. We then load up the column metadata for all columns in the slice.

The column metadata is stored in sorted order and is loaded into a Red-Black tree structure that allows the appropriate column metadata to be found in $O(\log n)$ time where n is the number of columns in the slice. The metadata contains the exact location on disk of the encoded image data. The data is read from the disk, Huffman codes are decoded, and the inverse wavelet transform is performed. If the metadata indicates that this is an index column, the query is finished and the data is returned.

If the column is a difference column, then the metadata contains the x_j^* coordinate of the associated index column. Finding the index column entry requires another search through the slice metadata at a cost of $O(\log n)$. The index column is read from disk, decoded, and transformed with the inverse wavelet transformation. The index data is then warped by the α and δ values associated with the difference column. The warped index data is added to the decoded difference column and the result is returned.

A query requires at most two columns to be decoded. If the column is an index column, only one column must be decoded. Otherwise, both the difference column and index column need to be decoded.

To avoid the cost of decoding a long chain of columns, our representation does not allow a column to depend on more than one other column. For the same reason, our representation does not exploit inter-EPI coherence. In order to take advantage of the similarity between neighboring EPIs, the representation would be forced to chain multiple columns together during the encoding process. We sacrifice the potential compression benefits of inter-EPI coherence to keep the number of computations required during the decoding stage at a minimum.

7. RESULTS

We have applied IRW to two benchmark environments. The first is a well-sampled synthetic model of a conference room. The room is circular in nature with a ring of desks. There are additional open areas at two ends of the room. The second environment is a real world scene courtesy of Lucent Technologies. The images are taken from inside a computer demonstration lab. The distribution of samples in the real world space is relatively sparse. Cylindrical panoramas captured from the two environments are shown in Figure 7.

7.1 Compression

When tested on both of our benchmark environments, IRW yields compression ratios similar to standard JPEG compression. A direct comparison with JPEG for both environments is shown in Figure 8. Our representation also provides mechanisms for trading im-

Figure 7: *Cylindrical panoramic images captured from the (a) synthetic and (b) real scenes analyzed in this paper. The synthetic model is a circular conference room. The real world panorama is an image captured from a computer demonstration lab at Lucent Technologies.*

Real Scene Using JPEG						
Quality	80	84	88	92	96	100
Size (KB)	11.5	13.9	17.2	21.6	34.0	63.7
PSNR	44.99	45.52	46.23	47.05	48.40	50.34
Real Scene Using IRW						
Quality	7	6	5	4	3	2
Size (KB)	17.8	21.4	26.6	33.9	45.6	61.4
PSNR	37.29	38.05	39.05	40.10	41.18	43.15
Synthetic Scene Using JPEG						
Quality	80	94	88	92	96	100
SIZE	12.9	14.8	17.5	21.6	31.8	56.0
PSNR	36.29	35.83	36.21	36.64	36.58	36.61
Synthetic Scene Using IRW						
Quality	7	6	5	4	3	2
Size (KB)	45.5	50.0	52.2	54.2	57.6	84.2
PSNR	38.59	39.39	40.26	41.40	43.17	45.01

Figure 8: *These tables compare the compression performance of IRW with that of JPEG. The first table shows JPEG applied to an EPI from a real scene. The second table shows IRW applied to the very same EPI. The third and fourth tables make the same comparisons for a synthetic EPI. For the JPEG trials, quality is measured using the JPEG quality value. For the IRW trials, quality is measured using the zero-cutoff threshold.*

age quality for increased compression rates, analogous to the JPEG quality value. One such mechanism is the zero-cutoff threshold. As shown in Figure 9, increasing this threshold increases compression at the expense of image quality as measured by peak signal-to-noise ratio (PSNR).

Finer level control can be achieved by adjusting the quantization resolution. Fewer bits per coefficient result in higher compression rates. A final quality control that can be used is to clamp higher frequency wavelet coefficients to zero. This allows a lower resolution version of the column being encoded on disk, resulting in higher compression rates.

7.2 Fast Queries

IRW requires at most two columns to be decoded for a single query. This is less work than required by other methods. For example, JPEG uses 8×8 blocks of pixels to encode images. Therefore, at least eight full columns must be decoded to retrieve a single column of image data. In practice, the costs with JPEG are higher

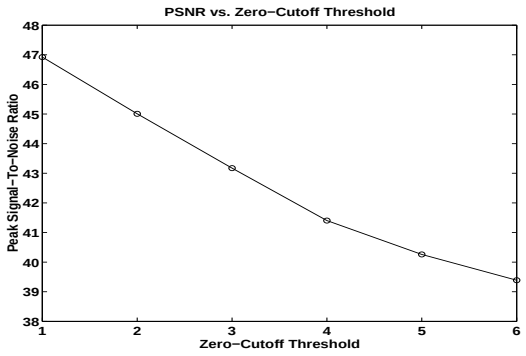

Figure 9: *The zero-cutoff threshold effects both compression size and image quality. This plot shows that when the threshold is raised, the peak signal-to-noise ratio is reduced.*

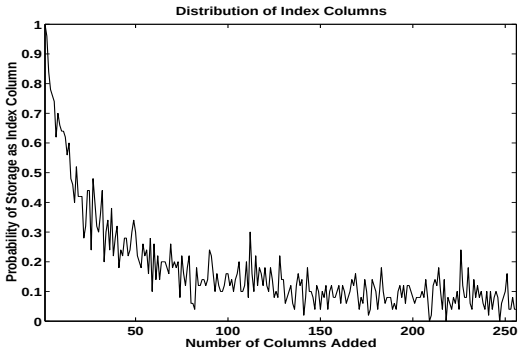

Figure 10: *As more column samples are stored in the database, the probability that a new sample must be stored as an index column decreases.*

than that because blocks are not individually addressable. Video compression methods that use differential coding, such as MPEG, have even worse query properties. To decode a single column from an MPEG frame, all pixel blocks containing the column must be decoded along with the blocks used for the differential coding.

Furthermore, many reconstruction algorithms combine samples from multiple images. For JPEG-based representations, the full query time needs to be spent for each sample taken from a unique image. For MPEG, the full query time can only be reduced if the unique images are differentially encoded using a common reference image. However, the IRW representation stores images in EPIs which group columns that observe the same scene objects. Two columns that are neighbors in an EPI can be retrieved by decoding at most three columns.

7.3 Scalability

The IRW representation scales well as the data set grows. When slices are poorly sampled, more columns tend to be encoded as index columns. This results in a substantial storage cost per column. However, as the sampling density increases, a larger percentage of columns are encoded as difference columns, reducing the marginal cost per column. Figure 10 shows the probability that a column is encoded as an index column as a function of the number of columns present in a slice.

Over a number of experiments, we found that the storage cost of an index column is on average 10.9 times greater than the cost of a difference column. This ratio depends on the encoding quality parameters. While the cost of an index column remains relatively constant, the cost of a difference column decreases as the quality decreases.

As shown in Figure 10, when new columns are added, there is a lower probability that they will be encoded as index columns. The result of this trend is a sub-linear growth rate for the IRW representation. The results of an experiment showing this behavior on synthetic images are shown in Figure 11.

7.4 Incremental Updates

As stated in the previous section, sub-linear growth implies that as more columns are added, an increasing percentage of them are encoded as difference columns. This has significant implications regarding incremental updates to the IRW representation. As stated in Section 5.4, adding difference columns requires relatively little work. None of the existing structure needs to be altered except for the sorted list of slice metadata. The encoded image data is unchanged. Additional work is performed to reinforce the en-

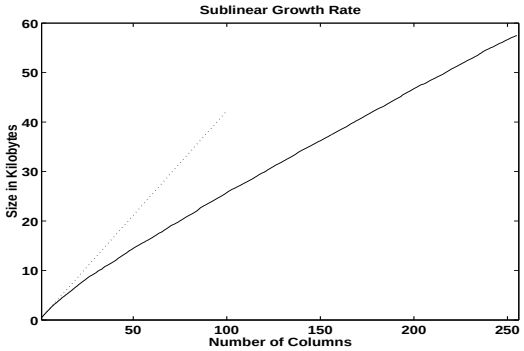

Figure 11: *The IRW representation exhibits sub-linear growth. As additional columns are added to the database, the marginal storage cost decreases. The data shown here was computed from images of the synthetic scene.*

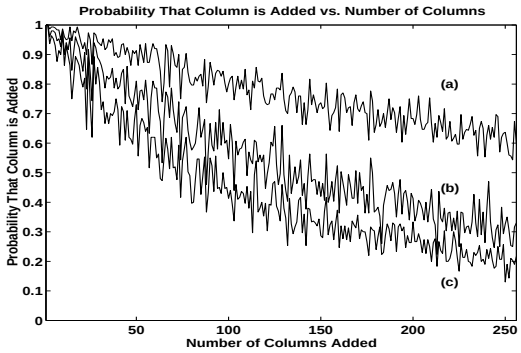

Figure 12: *The table shows the probability of a row being included for three different error tolerances. (a) shows the behavior for a low error tolerance while (c) shows a high error tolerance. The tolerance for (b) is in the middle. While the drop in probability is directly related to the allowed error, the general shape is the same for all three trials.*

coding invariant only when new index columns are added. This means that the cost of adding new columns to the representation decreases as the representation gets larger. As more columns are added to the representation, adding an additional column becomes easier and easier. Furthermore, because of the encoding invariant, any changes that may be needed are local in scope and only effect data located between the new index column and the index column that follows.

However, there is a price to pay for the incremental nature of IRW. The encoding invariant implies that the order of insertion impacts the compression rate. For example, in the worst case where columns are added in order of decreasing X^* values, every column is encoded as an index column. The best case occurs when the columns are added in increasing X^* order. Adding columns in random order increases the storage cost by an average of 44.8% over the best case storage size. This cost can be limited by occasionally re-packing a slice by re-encoding it in sorted order.

7.5 Overhead Costs

In addition to the image data, IRW stores indexing and metadata for each column. We have a storage overhead of 14 bytes per column. For a typical compressed image size (256 columns) of about 60KB, the overhead for this data is about 3.5KB, or 5.8%. We note however that this overhead is not unique to our representation. Any other method must store extra information to augment the raw image data such as camera pose estimation.

7.6 Selective Inclusion

One of the design goals for IRW is selective inclusion. The performance of this feature depends upon the reconstruction algorithm's ability to use existing samples to predict the new sample under consideration. In general, a poor reconstruction algorithm requires a large number of samples while a better reconstruction algorithm can produce a reconstruction of equal quality with fewer samples in the database. IRW is flexible and can work with any reconstruction method, but the degree to which individual samples can be omitted from the database depends on which algorithm is used.

For our experiments, we used an extremely simple reconstruction algorithm: comparing a new sample with the nearest neighbor already in the database. We then subtracted the two columns and if the difference was small enough, we discarded the new column. Figure 12 shows that the probability of including a column decreases as the size of the database grows. The exact probability

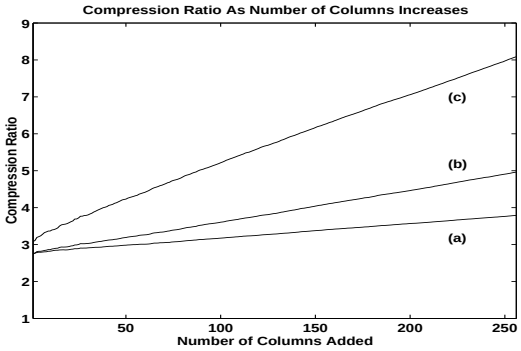

Figure 13: *The compression ratio is plotted as a function of the number of rows added to the representation for the same three error tolerances shown in Figure 12. The three trials are labeled with the same letters for comparison. A higher error tolerance results in higher compression rates. Even at low error tolerances, the compression rate steadily increases as columns are added.*

depends heavily on both the reconstruction performance and the acceptable error, but the general trend should hold for most reconstruction algorithms.

Selective inclusion includes columns where the scene is hard to reconstruct, but rejects columns where the reconstruction algorithm already has enough samples. The ability to discard columns that don't improve reconstruction quality allows us to greatly improve our compression rate. Figure 13 shows the compression rate as columns are sent to the database for inclusion. The rate increases as columns are processed because an increasing percentage of columns can be rejected.

In the limit, the database will reach the point where the scene is adequately sampled in all dimensions. At this point, all additional columns can be safely rejected. This implies that there is some point at which the database can recognize that it has been sampled at the rate needed by a given reconstruction algorithm to reconstruct the entire scene within a given error tolerance.

8. CONCLUSIONS AND FUTURE WORK

We have presented an image database representation for a subclass of IBR applications that use images taken from an environ-

ment such that all camera positions lie upon a plane that is orthogonal to the camera's image plane. We exploit camera pose information associated with each image, allowing us to reorganize the image data into highly coherent epipolar plane images.

The IRW representation provides fine grained access to image data by supporting queries for individual columns of data. IRW also supports selective inclusion which allows us to include only useful portions of a given image in our representation. In addition, the IRW representation is incremental and allows new samples to be easily and efficiently integrated with an existing dataset.

Along with these novel features, IRW yields compression rates similar to JPEG and provides controls over the encoding process that trades compression rates for image quality.

As part of future work, we plan to integrate existing reconstruction algorithms with our representation as well as explore new methods of reconstruction from image-based datasets.

There is also a need for further exploration of the behavior of the IRW representation in the limit as the number of samples grows. We expect that this property can be used to help address the next-best-view problem with regards to the automatic capture of an unknown scene [35]. Slices that suffer from poor compression ratios are in need of additional samples. Conversely, highly compressed slices are sampled well enough for the environment's complexity.

There are also many short-term improvements to be made. Portions of our prototype can be optimized for both speed and compression. We also plan to encode a real-world dataset with more uniform spatial properties than the dataset analyzed in this paper. Finally, we would like to extend the IRW representation to support application-level requirements of various reconstruction algorithms, such as image feature correspondences.

9. ACKNOWLEDGMENTS

We would like to thank Daniel Aliaga at Lucent Technologies Bell Laboratories for allowing us to experiment using his real-world image data collection. This work has been supported in part by ARO Contract DAAD19-99-1-0162, NSF awards ACI 9876914 and ACI 0118743, ONR Contract N00014-01-1-0067, a DOE ASCI grant, and by Intel Corporation.

10. REFERENCES

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

[2] D. Aliaga, J. Cohen, A. Wilson, H. Zhang, C. Erikson, K. Hoff, T. Hudson, W. Stuerzlinger, E. Baker, R. Bastos, M. Whitton, F. Brooks, and D. Manocha. Mmr: An integrated massive model rendering system using geometric and image-based acceleration. In *Proc. of ACM Symposium on Interactive 3D Graphics*, 1999.

[3] D. Aliaga and A. Lastra. Automatic image placement to provide a guaranteed frame rate. In *Proc. of ACM SIGGRAPH*, 1999.

[4] D. G. Aliaga and I. Carlbom. Plenoptic stitching: A scalable method for reconstructing interactive walkthroughs. In *ACM SIGGRAPH*, pages 443–450, 2001.

[5] R. C. Bolles, H. H. Baker, and D. H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *International Journal of Computer Vision*, 1:7–55, 1987.

[6] S. Chen and L. Williams. View interpolation for image synthesis. In *Proc. of ACM SIGGRAPH*, volume 27, pages 279–288, 1993.

[7] S. E. Chen. Quicktime VR - an image-based approach to virtual environment navigation. In R. Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 29–38. ACM SIGGRAPH, Addison Wesley, Aug. 1995. held in Los Angeles, California, 06-11 August 1995.

[8] L. Darsa, B. Costa, and A. Varshney. Walkthroughs of complex environments using image-based simplification. *Computer and Graphics*, 22(1):55–69, 1998.

[9] X. Decoret, G. Schaufler, F. Sillion, and J. Dorsey. Multi-layered impostors for accelerated rendering. *Computer Graphics Forum*, 18(3), 1999.

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

[11] J. Grossman and W. J. Dally. Point sample rendering. *Eurographics Workshop on Rendering*, pages 181–192, 1998.

[12] ISO/IEC. *ISO/IEC International Standard 11172*, August 1993.

[13] ISO/IEC. *ISO/IEC International Standard 13818*, July 1995.

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

[15] S. B. Kang and R. Szeliski. 3d scene data recovery using omnidirectional baseline stereo. *IEEE CVPR*, pages 364–270, 1996.

[16] P. Lalonde and A. Fournier. Interactive rendering of wavelet projected light fields. *Proc. of Graphics Interface*, pages 107–114, 1999.

[17] M. Levoy. Polygon-assisted JPEG and MPEG compression of synthetic images. In *SIGGRAPH 95 Conference Proceedings*, pages 21–28, 1995.

[18] M. Levoy and P. Hanrahan. Light field rendering. In *SIGGRAPH 96 Conference Proceedings*, pages 31–42, 1996.

[19] P. Maciel and P. Shirley. Visual navigation of large environments using textured clusters. In *ACM Symposium on Interactive 3D Graphics*, pages 95–102, 1995.

[20] M. Magnor and B. Girod. Model-based coding of multi-viewpoint imagery. *SPIE Conference on Visual Communications and Image Processing*, pages 14–22, 2000.

[21] N. Max and K. Ohsaki. Rendering trees from precomputed Z-buffer views. In *Eurographics Rendering Workshop 1995*, 1995.

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

[23] G. Miller, S. Rubin, and D. Poncelen. Lazy decompression of surface light fields for pre-computer global illumination. *Proc. of Eurographics Workshop on Rendering*, pages 281–292, 1998.

[24] W. Pennebaker and J. Mitchell. *JPEG: Still Image Data Compression Standard*. Van Nostrand Reinhold, 1993.

[25] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. *Proc. of ACM SIGGRAPH*, 2000.

[26] K. R. Rao and P. Yip. *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Academic Press, Inc., 1990.

[27] J. Shade, S. Gortler, L. wei He, and R. Szeliski. Layered depth images. *Proc. of ACM SIGGRAPH*, pages 231–242, 1998.

[28] H. Shum and L. He. Rendering with concentric mosaics. *Proc. of ACM SIGGRAPH*, pages 299–306, 1999.

[29] F. Sillion, G. Drettakis, and B. Bodelet. Efficient impostor manipulation for real-time visualization of urban scenery. In *Computer Graphics Forum*, volume 16, 1997.

[30] R. Szeliski and H. Shum. Creating full view panoramic image mosaics and texture-mapped models. In *Proc. of ACM SIGGRAPH*, pages 251–258, July 1997.

[31] T. Takahashi, H. Kawasaki, K. Ikeuchi, and M. Sakauchi. Arbitrary view position and direction rendering for large-scale scenes. *IEEE CVPR*, pages 296–303, 2000.

[32] C. Taylor. Video plus. In *IEEE Workshop on Omnidirectional Vision*, pages 3–10, 2000.

[33] I. T. Union. *ITU-T H.261: Video codec for audiovisual services at p x 64 kbits.*, March 1993.

[34] M. Vetterli and J. Kavacevic. *Waveletes and Subband Coding*. Prentice Hall PTR, 1995.

[35] A. Wilson. *Spatial Representations of Image-Based Impostors for Interactive Walkthroughs*. PhD thesis, Department of Computer Science, University of North Carolina at Chapel Hill, 2002.

[36] A. Wilson, M. Lin, D. Manocha, B. Yeo, and M. Yeung. Video-based rendering acceleration algorithms for interactive walkthroughs. *Proc. of ACM Multimedia*, pages 75–84, 2000.

[37] A. Wilson, K. Mayer-Patel, and D. Manocha. Spatially-encoded far-field representations for interactive walkthroughs. *Proc. of ACM Multimedia*, 2001.

[38] Y. Wu, C. Zhang, J. Li, and J. Xu. Smart rebinning for compression of concentric mosaics. In *ACM Multimedia*, pages 201–209, 2000.