Light Field Rendering

Marc Levoy and Pat Hanrahan
Computer Science Department
Stanford University

Abstract

A number of techniques have been proposed for flying through scenes by redisplaying previously rendered or digitized views. Techniques have also been proposed for interpolating between views by warping input images, using depth information or correspondences between multiple images. In this paper, we describe a simple and robust method for generating new views from arbitrary camera positions without depth information or feature matching, simply by combining and resampling the available images. The key to this technique lies in interpreting the input images as 2D slices of a 4D function - the light field. This function completely characterizes the flow of light through unobstructed space in a static scene with fixed illumination.

We describe a sampled representation for light fields that allows for both efficient creation and display of inward and outward looking views. We have created light fields from large arrays of both rendered and digitized images. The latter are acquired using a video camera mounted on a computer-controlled gantry. Once a light field has been created, new views may be constructed in real time by extracting slices in appropriate directions. Since the success of the method depends on having a high sample rate, we describe a compression system that is able to compress the light fields we have generated by more than a factor of 100:1 with very little loss of fidelity. We also address the issues of antialiasing during creation, and resampling during slice extraction.

CR Categories: I.3.2 [Computer Graphics]: Picture/Image Generation — *Digitizing and scanning*, *Viewing algorithms*; I.4.2 [Computer Graphics]: Compression — *Approximate methods*

Additional keywords: image-based rendering, light field, holographic stereogram, vector quantization, epipolar analysis

1. Introduction

Traditionally the input to a 3D graphics system is a scene consisting of geometric primitives composed of different materials and a set of lights. Based on this input specification, the rendering system computes and outputs an image. Recently a new approach to rendering has emerged: *image-based rendering*. Image-based rendering systems generate different views of an environment from a set of pre-acquired imagery. There are several advantages to this approach:

- The display algorithms for image-based rendering require modest computational resources and are thus suitable for real-time implementation on workstations and personal computers.
- The cost of interactively viewing the scene is independent of scene complexity.
- The source of the pre-acquired images can be from a real or virtual environment, i.e. from digitized photographs or from rendered models. In fact, the two can be mixed together.

The forerunner to these techniques is the use of environment maps to capture the incoming light in a texture map [Blinn76, Greene86]. An environment map records the incident light arriving from all directions at a point. The original use of environment maps was to efficiently approximate reflections of the environment on a surface. However, environment maps also may be used to quickly display any outward looking view of the environment from a fixed location but at a variable orientation. This is the basis of the Apple QuickTimeVR system [Chen95]. In this system environment maps are created at key locations in the scene. The user is able to navigate discretely from location to location, and while at each location continuously change the viewing direction.

The major limitation of rendering systems based on environment maps is that the viewpoint is fixed. One way to relax this fixed position constraint is to use view interpolation [Chen93, Greene94, Fuchs94, McMillan95a, McMillan95b, Narayanan95]. Most of these methods require a depth value for each pixel in the environment map, which is easily provided if the environment maps are synthetic images. Given the depth value it is possible to reproject points in the environment map from different vantage points to warp between multiple images. The key challenge in this warping approach is to "fill in the gaps" when previously occluded areas become visible.

Another approach to interpolating between acquired images is to find corresponding points in the two [Laveau94, McMillan95b, Seitz95]. If the positions of the cameras are known, this is equivalent to finding the depth values of the corresponding points. Automatically finding correspondences between pairs of images is the classic problem of stereo vision, and unfortunately although many algorithms exist, these algorithms are fairly fragile and may not always find the correct correspondences.

In this paper we propose a new technique that is robust and allows much more freedom in the range of possible views. The major idea behind the technique is a representation of the *light field*, the radiance as a function of position and direction, in regions of space free of occluders (free space). In free space, the light field is a 4D, not a 5D function. An image is a two dimensional slice of the 4D light field. Creating a light field from a set of images corresponds to inserting each 2D slice into the 4D light field representation. Similarly, generating new views corresponds to extracting and resampling a slice.

Address: Gates Computer Science Building 3B levoy@cs.stanford.edu
Stanford University hanrahan@cs.stanford.edu
Stanford, CA 94305 http://www-graphics.stanford.edu

Generating a new image from a light field is quite different than previous view interpolation approaches. First, the new image is generally formed from many different pieces of the original input images, and need not look like any of them. Second, no model information, such as depth values or image correspondences, is needed to extract the image values. Third, image generation involves only resampling, a simple linear process.

This representation of the light field is similar to the epipolar volumes used in computer vision [Bolles87] and to horizontal-parallax-only holographic stereograms [Benton83]. An epipolar volume is formed from an array of images created by translating a camera in equal increments in a single direction. Such a representation has recently been used to perform view interpolation [Katayama95]. A holographic stereogram is formed by exposing a piece of film to an array of images captured by a camera moving sideways. Halle has discussed how to set the camera aperture to properly acquire images for holographic stereograms [Halle94], and that theory is applicable to this work. Gavin Miller has also recognized the potential synergy between true 3D display technologies and computer graphics algorithms [Miller95].

There are several major challenges to using the light field approach to view 3D scenes on a graphics workstation. First, there is the choice of parameterization and representation of the light field. Related to this is the choice of sampling pattern for the field. Second, there is the issue of how to generate or acquire the light field. Third, there is the problem of fast generation of different views. This requires that the slice representing rays through a point be easily extracted, and that the slice be properly resampled to avoid artifacts in the final image. Fourth, the obvious disadvantage of this approach is the large amount of data that may be required. Intuitively one suspects that the light field is coherent and that it may be compressed greatly. In the remaining sections we discuss these issues and our proposed solutions.

2. Representation

We define the light field as the radiance at a point in a given direction. Note that our definition is equivalent to the *plenoptic function* introduced by Adelson and Bergen [Adelson91]. The phrase light field was coined by A. Gershun in his classic paper describing the radiometric properties of light in a space [Gershun36]. [1] McMillan and Bishop [McMillan95b] discuss the representation of 5D light fields as a set of panoramic images at different 3D locations.

However, the 5D representation may be reduced to 4D in free space (regions free of occluders). This is a consequence of the fact that the radiance does not change along a line unless blocked. 4D light fields may be interpreted as functions on the space of oriented lines. The redundancy of the 5D representation is undesirable for two reasons: first, redundancy increases the size of the total dataset, and second, redundancy complicates the reconstruction of the radiance function from its samples. This reduction in dimension has been used to simplify the representation of radiance emitted by luminaires [Levin71, Ashdown93]. For the remainder of this paper we will be only concerned with 4D light fields.

[1] For those familiar with Gershun's paper, he actually uses the term light field to mean the irradiance vector as a function of position. For this reason P. Moon in a later book [Moon81] uses the term photic field to denote what we call the light field.

Although restricting the validity of the representation to free space may seem like a limitation, there are two common situations where this assumption is useful. First, most geometric models are bounded. In this case free space is the region outside the convex hull of the object, and hence all views of an object from outside its convex hull may be generated from a 4D light field. Second, if we are moving through an architectural model or an outdoor scene we are usually moving through a region of free space; therefore, any view from inside this region, of objects outside the region, may be generated.

The major issue in choosing a representation of the 4D light field is how to parameterize the space of oriented lines. There are several issues in choosing the parameterization:

Efficient calculation. The computation of the position of a line from its parameters should be fast. More importantly, for the purposes of calculating new views, it should be easy to compute the line parameters given the viewing transformation and a pixel location.

Control over the set of lines. The space of all lines is infinite, but only a finite subset of line space is ever needed. For example, in the case of viewing an object we need only lines intersecting the convex hull of the object. Thus, there should be an intuitive connection between the actual lines in 3-space and line parameters.

Uniform sampling. Given equally spaced samples in line parameter space, the pattern of lines in 3-space should also be uniform. In this sense, a uniform sampling pattern is one where the *number of lines* in intervals between samples is constant everywhere. Note that the correct measure for number of lines is related to the form factor kernel [Sbert93].

The solution we propose is to parameterize lines by their intersections with two planes in arbitrary position (see figure 1). By convention, the coordinate system on the first plane is (u, v) and on the second plane is (s, t). An oriented line is defined by connecting a point on the uv plane to a point on the st plane. In practice we restrict u, v, s, and t to lie between 0 and 1, and thus points on each plane are restricted to lie within a convex quadrilateral. We call this representation a *light slab*. Intuitively, a light slab represents the beam of light entering one quadrilateral and exiting another quadrilateral.

A nice feature of this representation is that one of the planes may be placed at infinity. This is convenient since then lines may be parameterized by a point and a direction. The latter will prove useful for constructing light fields either from orthographic images or images with a fixed field of view. Furthermore, if all calculations are performed using homogeneous coordinates, the two cases may be handled at no additional cost.

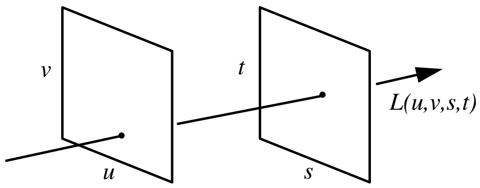


Figure 1: The light slab representation.

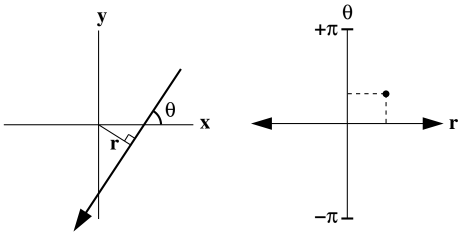


Figure 2: Definition of the line space we use to visualize sets of light rays. Each oriented line in Cartesian space (at left) is represented in line space (at right) by a point. To simplify the visualizations, we show only lines in 2D; the extension to 3D is straightforward.

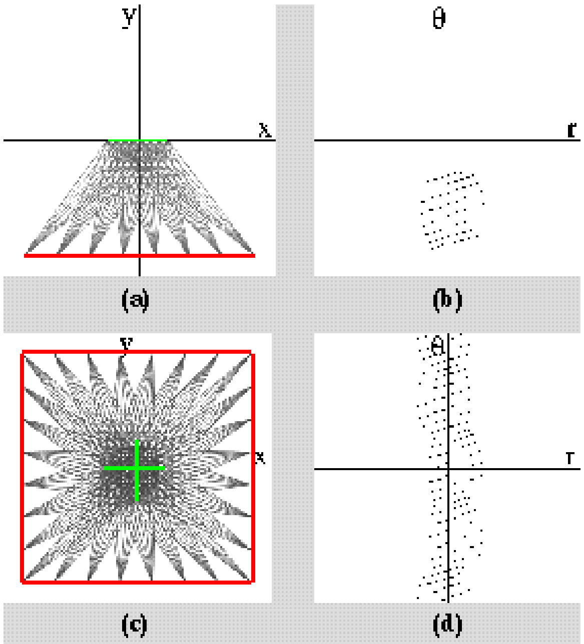


Figure 3: Using line space to visualize ray coverage. (a) shows a single light slab. Light rays (drawn in gray) connect points on two defining lines (drawn in red and green). (c) shows an arrangement of four rotated copies of (a). (b) and (d) show the corresponding line space visualizations. For any set of lines in Cartesian space, the envelope formed by the corresponding points in line space indicates our coverage of position and direction; ideally the coverage should be complete in θ and as wide as possible in r. As these figures show, the single slab in (a) does not provide full coverage in θ, but the four-slab arrangement in (c) does. (c) is, however, narrow in r. Such an arrangement is suitable for inward-looking views of a small object placed at the origin. It was used to generate the lion light field in figure 14d.

A big advantage of this representation is the efficiency of geometric calculations. Mapping from (u, v) to points on the plane is a projective map and involves only linear algebra (multiplying by a 3x3 matrix). More importantly, as will be discussed in section 5, the inverse mapping from an image pixel (x, y) to (u, v, s, t) is also a projective map. Methods using spherical or cylindrical coordinates require substantially more computation.

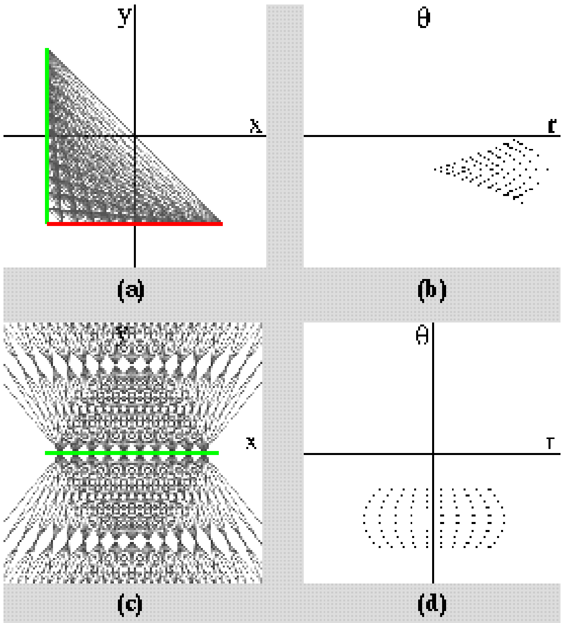


Figure 4: Using line space to visualize sampling uniformity. (a) shows a light slab defined by two lines at right angles. (c) shows a light slab where one defining line is at infinity. This arrangement generates rays passing through the other defining line with an angle between -45° and +45°. (b) and (d) show the corresponding line space visualizations. Our use of (r, θ) to parameterize line space has the property that equal areas in line space correspond to equally dense sampling of position and orientation in Cartesian space; ideally the density of points in line space should be uniform. As these figures show, the singularity at the corner in (a) leads to a highly nonuniform and therefore inefficient sampling pattern, indicated by dark areas in (b) at angles of 0 and $-\pi/2$. (c) generates a more uniform set of lines. Although (c) does not provide full coverage of θ, four rotated copies do. Such an arrangement is suitable for outward-looking views by an observer standing near the origin. It was used to generate the hallway light field in figure 14c.

Many properties of light fields are easier to understand in line space (figures 2 through 4). In line space, each oriented line is represented by a point and each set of lines by a region. In particular, the set of lines represented by a light slab and the set of lines intersecting the convex hull of an object are both regions in line space. All views of an object could be generated from one light slab if its set of lines include all lines intersecting the convex hull of the object. Unfortunately, this is not possible. Therefore, it takes multiple light slabs to represent all possible views of an object. We therefore tile line space with a collection of light slabs, as shown in figure 3.

An important issue related to the parameterization is the sampling pattern. Assuming that all views are equally likely to be generated, then any line is equally likely to be needed. Thus all regions of line space should have an equal density of samples. Figure 4 shows the density of samples in line space for different arrangements of slabs. Note that no slab arrangement is perfect: arrangements with a singularity such as two polygons joined at a corner (4a) are bad and should be avoided, whereas slabs formed from parallel planes (3a) generate fairly uniform patterns. In addition, arrangements where one plane is at infinity (4c) are better than those with two finite planes (3a). Finally, because of symmetry the spacing of samples in uv should in general be the same as st. However, if the observer is likely to stand near the uv plane, then it may be acceptable to sample uv less frequently than st.

3. Creation of light fields

In this section we discuss the creation of both virtual light fields (from rendered images) and real light fields (from digitized images). One method to create a light field would be to choose a 4D sampling pattern, and for each line sample, find the radiance. This is easily done directly for virtual environments by a ray tracer. This could also be done in a real environment with a spot radiometer, but it would be very tedious. A more practical way to generate light fields is to assemble a collection of images.

3.1. From rendered images

For a virtual environment, a light slab is easily generated simply by rendering a 2D array of images. Each image represents a slice of the 4D light slab at a fixed uv value and is formed by placing the center of projection of the virtual camera at the sample location on the uv plane. The only issue is that the xy samples of each image must correspond exactly with the st samples. This is easily done by performing a sheared perspective projection (figure 5) similar to that used to generate a stereo pair of images. Figure 6 shows the resulting 4D light field, which can be visualized either as a uv array of st images or as an st array of uv images.

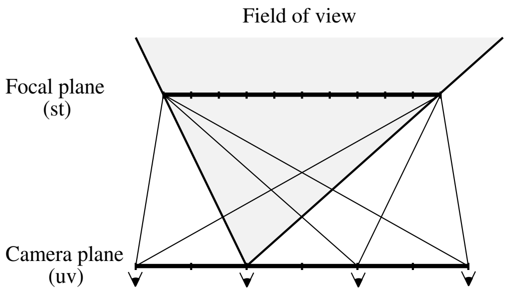


Figure 5: The viewing geometry used to create a light slab from an array of perspective images.

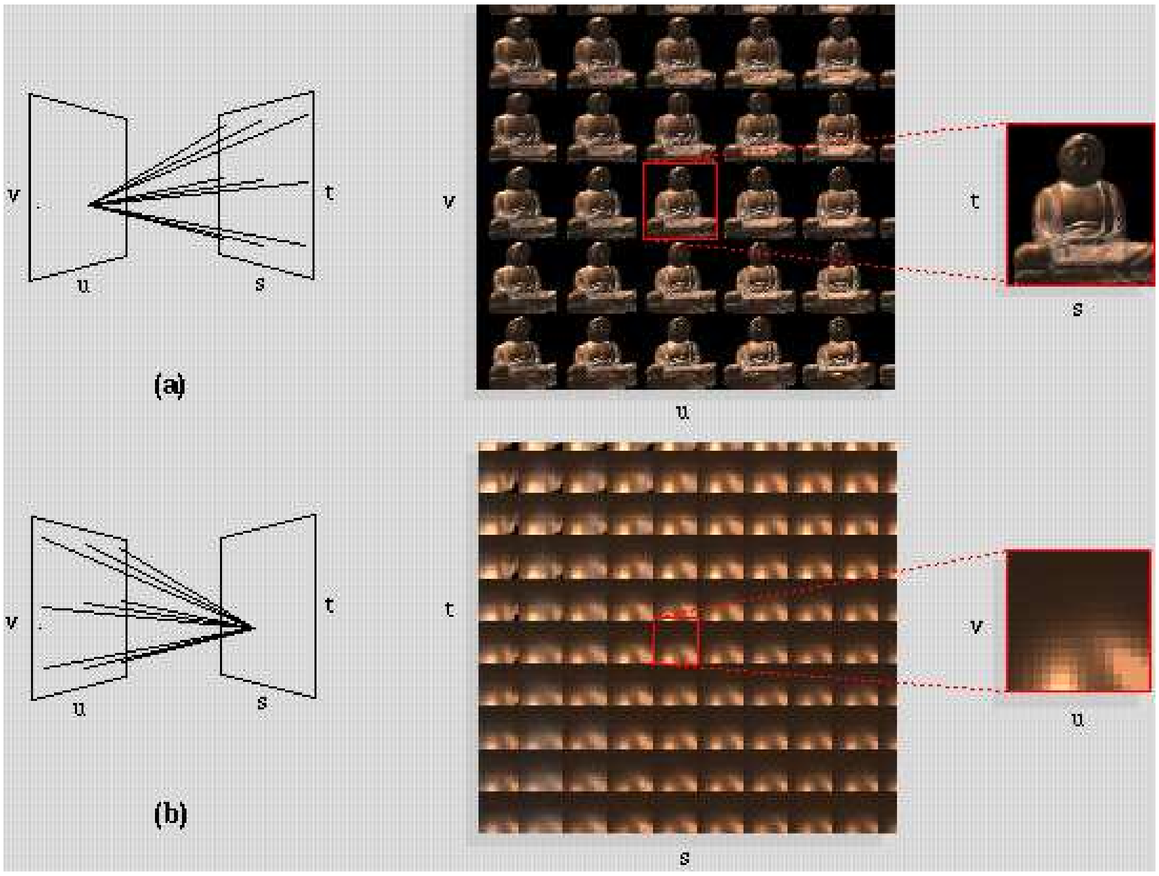


Figure 6: Two visualizations of a light field. (a) Each image in the array represents the rays arriving at one point on the uv plane from all points on the st plane, as shown at left. (b) Each image represents the rays leaving one point on the st plane bound for all points on the uv plane. The images in (a) are off-axis (i.e. sheared) perspective views of the scene, while the images in (b) look like reflectance maps. The latter occurs because the object has been placed astride the focal plane, making sets of rays leaving points on the focal plane similar in character to sets of rays leaving points on the object.

Two other viewing geometries are useful. A light slab may be formed from a 2D array of orthographic views. This can be modeled by placing the uv plane at infinity, as shown in figure 4c. In this case, each uv sample corresponds to the direction of a parallel projection. Again, the only issue is to align the xy and st samples of the image with the st quadrilateral. The other useful geometry consists of a 2D array of outward looking (non-sheared) perspective views with fixed field of view. In this case, each image is a slice of the light slab with the st plane at infinity. The fact that all these cases are equally easy to handle with light slabs attests to the elegance of projective geometry. Light fields using each arrangement are presented in section 6 and illustrated in figure 14.

As with any sampling process, sampling a light field may lead to aliasing since typical light fields contain high frequencies. Fortunately, the effects of aliasing may be alleviated by filtering before sampling. In the case of a light field, a 4D filter in the space of lines must be employed (see figure 7). Assuming a box filter, a weighted average of the radiances on all lines connecting sample squares in the uv and st planes must be computed. If a camera is placed on the uv plane and focussed on the st plane, then the filtering process corresponds to integrating both over a pixel corresponding to an st sample, and an aperture equal in size to a uv sample, as shown in figure 8. The theory behind this filtering process has been discussed in the context of holographic stereograms by Halle [Halle94].

Note that although prefiltering has the desired effect of antialiasing the light field, it has what at first seems like an undesirable side effect — introducing blurriness due to depth of field. However, this blurriness is precisely correct for the situation. Recall what happens when creating a pair of images from two adjacent camera locations on the uv plane: a given object point will project to different locations, potentially several pixels apart, in these two images. The distance between the two projected locations is called the stereo disparity. Extending this idea to multiple camera locations produces a sequence of images in which the object appears to jump by a distance equal to the disparity. This jumping is aliasing. Recall now that taking an image with a finite aperture causes points out of focus to be blurred on the film plane by a circle of confusion. Setting the diameter of the aperture to the spacing between camera locations causes the circle of confusion for each object point to be equal in size to its stereo disparity. This replaces the jumping with a sequence of blurred images. Thus, we are removing aliasing by employing finite depth of field.

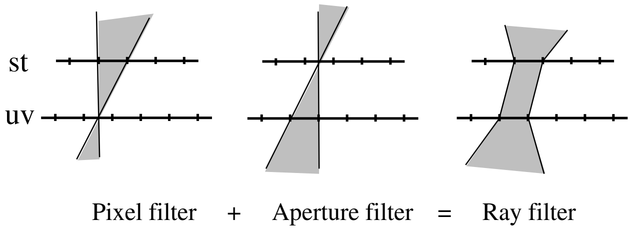


Figure 7: Prefiltering a light field. To avoid aliasing, a 4D low pass filter must be applied to the radiance function.

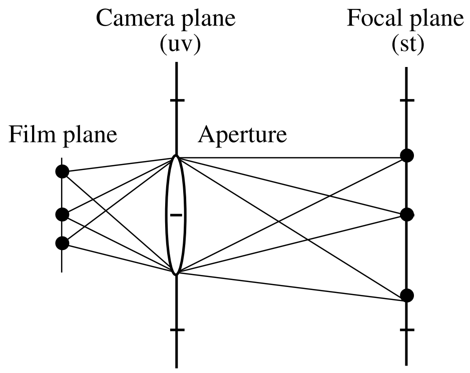


Figure 8: Prefiltering using an aperture. This figure shows a camera focused on the st plane with an aperture on the uv plane whose size is equal to the uv sample spacing. A hypothetical film plane is drawn behind the aperture. Ignore the aperture for a moment (consider a pinhole camera that precisely images the st plane onto the film plane). Then integrating over a pixel on the film plane is equivalent to integrating over an st region bounded by the pixel. Now consider fixing a point on the film plane while using a finite sized aperture (recall that all rays from a point on the film through the aperture are focussed on a single point on the focal plane). Then integrating over the aperture corresponds to integrating all rays through the uv region bounded by the aperture. Therefore, by simultaneously integrating over both the pixel and the aperture, the proper 4D integral is computed.

The necessity for prefiltering can also be understood in line space. Recall from our earlier discussion that samples of the light field correspond to points in line space. Having a finite depth of field with an aperture equal in size to the uv sample spacing insures that each sample adequately covers the interval between these line space points. Too small or too large an aperture yields gaps or overlaps in line space coverage, resulting in views that are either aliased or excessively blurry, respectively.

3.2. From digitized images

Digitizing the imagery required to build a light field of a physical scene is a formidable engineering problem. The number of images required is large (hundreds or thousands), so the process must be automated or at least computer-assisted. Moreover, the lighting must be controlled to insure a static light field, yet flexible enough to properly illuminate the scene, all the while staying clear of the camera to avoid unwanted shadows. Finally, real optical systems impose constraints on angle of view, focal distance, depth of field, and aperture, all of which must be managed. Similar issues have been faced in the construction of devices for performing near-field photometric measurements of luminaires [Ashdown93]. In the following paragraphs, we enumerate the major design decisions we faced in this endeavor and the solutions we adopted.

Inward versus outward looking. The first decision to be made was between a flyaround of a small object and a flythrough of a large-scale scene. We judged flyarounds to be the simpler case, so we attacked them first.

Figure 9: Our prototype camera gantry. A modified Cyberware MS motion platform with additional stepping motors from Lin-Tech and Parker provide four degrees of freedom: horizontal and vertical translation, pan, and tilt. The camera is a Panasonic WV-F300 3-CCD video camera with a Canon f/1.7 10-120mm zoom lens. We keep it locked off at its widest setting (10mm) and mounted so that the pitch and yaw axes pass through the center of projection. While digitizing, the camera is kept pointed at the center of the focal plane. Calibrations and alignments are verified with the aid of a Faro digitizing arm, which is accurate to 0.3 mm.

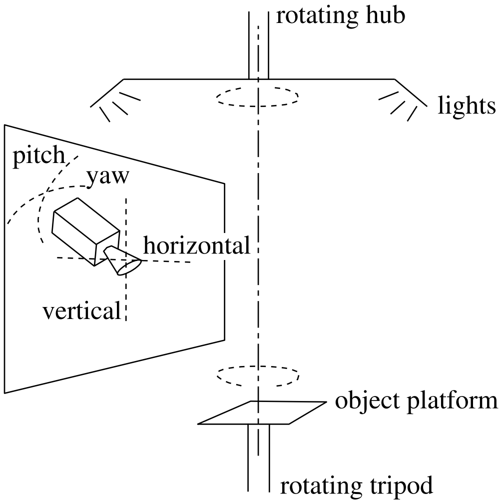


Figure 10: Object and lighting support. Objects are mounted on a Bogen fluid-head tripod, which we manually rotate to four orientations spaced 90 degrees apart. Illumination is provided by two 600W Lowell Omni spotlights attached to a ceiling-mounted rotating hub that is aligned with the rotation axis of the tripod. A stationary 6' x 6' diffuser panel is hung between the spotlights and the gantry, and the entire apparatus is enclosed in black velvet to eliminate stray light.

Human versus computer-controlled. An inexpensive approach to digitizing light fields is to move a handheld camera through the scene, populating the field from the resulting images [Gortler96]. This approach necessitates estimating camera pose at each frame and interpolating the light field from scattered data - two challenging problems. To simplify the situation, we chose instead to build a computer-controlled camera gantry and to digitize images on a regular grid.

Spherical versus planar camera motion. For flyarounds of small objects, an obvious gantry design consists of two concentric hemicycles, similar to a gyroscope mounting. The camera in such a gantry moves along a spherical surface, always pointing at the center of the sphere. Apple Computer has constructed such a gantry to acquire imagery for Quick-Time VR flyarounds [Chen95]. Unfortunately, the lighting in their system is attached to the moving camera, so it is unsuitable for acquiring static light fields. In general, a spherical gantry has three advantages over a planar gantry: (a) it is easier to cover the entire range of viewing directions, (b) the sampling rate in direction space is more uniform, and (c) the distance between the camera and the object is fixed, providing sharper focus throughout the range of camera motion. A planar gantry has two advantages over a spherical gantry: (a) it is easier to build; the entire structure can be assembled from linear motion stages, and (b) it is closer to our light slab representation. For our first prototype gantry, we chose to build a planar gantry, as shown in figure 9.

Field of view. Our goal was to build a light field that allowed 360 degrees of azimuthal viewing. To accomplish this using a planar gantry meant acquiring four slabs each providing 90 degrees. This can be achieved with a camera that translates but does not pan or tilt by employing a wide-angle lens. This solution has two disadvantages: (a) wide-angle lenses exhibit significant distortion, which must be corrected after acquisition, and (b) this solution trades off angle of view against sensor resolution. Another solution is to employ a view camera in which the sensor and optical system translate in parallel planes, the former moving faster than the latter. Horizontal parallax holographic stereograms are constructed using such a camera [Halle94]. Incorporating this solution into a gantry that moves both horizontally and vertically is difficult. We instead chose to equip our camera with pan and tilt motors, enabling us to use a narrow-angle lens. The use of a rotating camera means that, in order to transfer the acquired image to the light slab representation, it must be reprojected to lie on a common plane. This reprojection is equivalent to keystone correction in architectural photography.

Standoff distance. A disadvantage of planar gantries is that the distance from the camera to the object changes as the camera translates across the plane, making it difficult to keep the object in focus. The view camera described above does not suffer from this problem, because the ratio of object distance to image distance stays constant as the camera translates. For a rotating camera, servo-controlled focusing is an option, but changing the focus of a camera shifts its center of projection and changes the image magnification, complicating acquisition. We instead mitigate this problem by using strong lighting and a small aperture to maximize depth of field.

Sensor rotation. Each sample in a light slab should ideally represent the integral over a pixel, and these pixels should lie on a common focal plane. A view camera satisfies this constraint because its sensor translates in a plane. Our use of a rotating camera means that the focal plane also rotates. Assuming that we resample the images carefully during reprojection, the presence of a rotated focal plane will introduce no additional error into the light field. In practice, we have not seen artifacts due to this resampling process.

Aperture size. Each sample in a light slab should also represent the integral over an aperture equal in size to a uv sample. Our use of a small aperture produces a light field with little or no uv antialiasing. Even fully open, the apertures of commercial video cameras are small. We can approximate the required antialiasing by averaging together some number of adjacent views, thereby creating a *synthetic aperture*. However, this technique requires a very dense spacing of views, which in turn requires rapid acquisition. We do not currently do this.

Object support. In order to acquire a 360-degree light field in four 90-degree segments using a planar gantry, either the gantry or the object must be rotated to each of four orientations spaced 90 degrees apart. Given the massiveness of our gantry, the latter was clearly easier. For these experiments, we mounted our objects on a tripod, which we manually rotate to the four positions as shown in figure 10.

Lighting. Given our decision to rotate the object, satisfying the requirement for fixed illumination means that either the lighting must exhibit fourfold symmetry or it must rotate with the object. We chose the latter solution, attaching a lighting system to a rotating hub as shown in figure 10. Designing a lighting system that stays clear of the gantry, yet provides enough light to evenly illuminate an object, is a challenging problem.

Using this gantry, our procedure for acquiring a light field is as follows. For each of the four orientations, the camera is translated through a regular grid of camera positions. At each position, the camera is panned and tilted to point at the center of the object, which lies along the axis of rotation of the tripod. We then acquire an image, and, using standard texture mapping algorithms, reproject it to lie on a common plane as described earlier. Table II gives a typical set of acquisition parameters. Note that the distance between camera positions (3.125 cm) exceeds the diameter of the aperture (1.25 mm), underscoring the need for denser spacing and a synthetic aperture.

4. Compression

Light field arrays are large — the largest example in this paper is 1.6 GB. To make creation, transmission, and display of light fields practical, they must be compressed. In choosing from among many available compression techniques, we were guided by several unique characteristics of light fields:

Data redundancy. A good compression technique removes redundancy from a signal without affecting its content. Light fields exhibit redundancy in all four dimensions. For example, the smooth regions in figure 6a tell us that this light field contains redundancy in s and t, and the smooth regions in figure 6b tell us that the light field contains redundancy in u and v. The former corresponds to our usual notion of interpixel coherence in a perspective view. The latter can be interpreted either as the interframe coherence one expects in a motion sequence or as the smoothness one expects in the bidirectional reflectance distribution function (BRDF) for a diffuse or moderately specular surface. Occlusions introduce discontinuities in both cases, of course.

Random access. Most compression techniques place some constraint on random access to data. For example, variable-bitrate coders may require scanlines, tiles, or frames to be decoded at once. Examples in this class are variable-bitrate vector quantization and the Huffman or arithmetic coders used in JPEG or MPEG. Predictive coding schemes further complicate random-access because pixels depend on previously decoded pixels, scanlines, or frames. This poses a problem for light fields since the set of samples referenced when extracting an image from a light field are dispersed in memory. As the observer moves, the access patterns change in complex ways. We therefore seek a compression technique that supports low-cost random access to individual samples.

Asymmetry. Applications of compression can be classified as symmetric or asymmetric depending on the relative time spent encoding versus decoding. We assume that light fields are assembled and compressed ahead of time, making this an asymmetric application.

Computational expense. We seek a compression scheme that can be decoded without hardware assistance. Although software decoders have been demonstrated for standards like JPEG and MPEG, these implementations consume the full power of a modern microprocessor. In addition to decompression, the display algorithm has additional work to perform, as will be described in section 5. We therefore seek a compression scheme that can be decoded quickly.

The compression scheme we chose was a two-stage pipeline consisting of fixed-rate vector quantization followed by entropy coding (Lempel-Ziv), as shown in figure 11. Following similar motivations, Beers et al. use vector quantization to compress textures for use in rendering pipelines [Beers96].

4.1. Vector quantization

The first stage of our compression pipeline is vector quantization (VQ) [Gersho92], a lossy compression technique wherein a vector of samples is quantized to one of a number of predetermined reproduction vectors. A reproduction vector is called a codeword, and the set of codewords available to encode a source is called the codebook, Codebooks are constructed during a training phase in which the quantizer is asked to find a set of codewords that best approximates a set of sample vectors, called the training set. The quality of a codeword is typically characterized

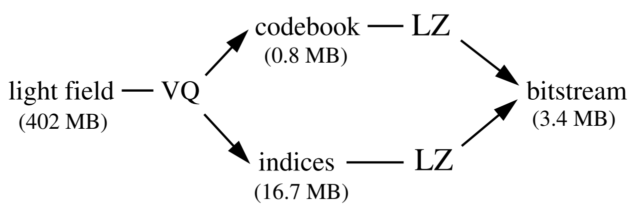


Figure 11 Two-stage compression pipeline. The light field is partitioned into tiles, which are encoded using vector quantization to form an array of codebook indices. The codebook and the array of indices are further compressed using Lempel-Ziv coding. Decompression also occurs in two stages: entropy decoding as the file is loaded into memory, and dequantization on demand during interactive viewing. Typical file sizes are shown beside each stage.

using mean-squared error (MSE), i.e. the sum over all samples in the vector of the squared difference between the source sample and the codeword sample. Once a codebook has been constructed, encoding consists of partitioning the source into vectors and finding for each vector the closest approximating codeword from the codebook. Decoding consists of looking up indices in the codebook and outputting the codewords found there — a very fast operation. Indeed, decoding speed is one of the primary advantages of vector quantization.

In our application, we typically use 2D or 4D tiles of the light field, yielding 12-dimensional or 48-dimensional vectors, respectively. The former takes advantage of coherence in s and t only, while the latter takes advantage of coherence in all four dimensions. To maximize image quality, we train on a representative subset of each light field to be compressed, then transmit the resulting codebook along with the codeword index array. Since light fields are large, even after compression, the additional overhead of transmitting a codebook is small, typically less than 20%. We train on a subset rather than the entire light field to reduce the expense of training.

The output of vector quantization is a sequence of fixed-rate codebook indices. Each index is log N bits where N is the number of codewords in the codebook, so the compression rate of the quantizer is (kl) / $(log\ N)$ where k is the number of elements per vector (i.e. the dimension), and l is the number of bits per element, usually 8. In our application, we typically use 16384-word codebooks, leading to a compression rate for this stage of the pipeline of (48 x 8) / (log 16384) = 384 bits / 14 bits = 27:1. To simplify decoding, we represent each index using an integral number of bytes, 2 in our case, which reduces our compression slightly, to 24:1.

4.2. Entropy coding

The second stage of our compression pipeline is an entropy coder designed to decrease the cost of representing high-probability code indices. Since our objects are typically rendered or photographed against a constant-color background, the array contains many tiles that occur with high probability. For the examples in this paper, we employed gzip, an implementation of Lempel-Ziv coding [Ziv77]. In this algorithm, the input stream is partitioned into nonoverlapping blocks while constructing a dictionary of blocks seen thus far. Applying gzip to our array of code indices typically gives us an additional 5:1 compression. Huffman coding would probably yield slightly higher compression, but encoding and decoding would be more expensive. Our total compression is therefore 24 x 5 = 120:1. See section 6 and table III for more detail on our compression results.

4.3. Decompression

Decompression occurs in two stages. The first stage — gzip decoding — is performed as the file is loaded into memory. The output of this stage is a codebook and an array of code indices packed in 16-bit words. Although some efficiency has been lost by this decoding, the light field is still compressed 24:1, and it is now represented in a way that supports random access.

The second stage — dequantization — proceeds as follows. As the observer moves through the scene, the display engine requests samples of the light field. Each request consists of a (u, v, s, t) coordinate tuple. For each request, a subscripting calculation is performed to determine which sample tile is being addressed. Each tile corresponds to one quantization vector and is thus represented in the index array by a single entry. Looking this index up in the codebook, we find a vector of sample values. A second subscripting calculation is then performed, giving us the offset of the requested sample within the vector. With the aid of precomputed subscripting tables, dequantization can be implemented very efficiently. In our tests, decompression consumes about 25% of the CPU cycles.

5. Display

The final part of the system is a real time viewer that constructs and displays an image from the light slab given the imaging geometry. The viewer must resample a 2D slice of lines from the 4D light field; each line represents a ray through the eye point and a pixel center as shown in figure 12. There are two steps to this process: step 1 consists of computing the (u, v, s, t) line parameters for each image ray, and step 2 consists of resampling the radiance at those line parameters.

As mentioned previously, a big advantage of the light slab representation is the efficiency of the inverse calculation of the line parameters. Conceptually the (u, v) and (s, t) parameters may be calculated by determining the point of intersection of an image ray with each plane. Thus, any ray tracer could easily be adapted to use light slabs. However, a polygonal rendering system also may be used to view a light slab. The transformation from image coordinates (x, y) to both the (u, v) and the (s, t) coordinates is a projective map. Therefore, computing the line coordinates can be done using texture mapping. The uv quadrilateral is drawn using the current viewing transformation, and during scan conversion the (uw, vw, w) coordinates at the corners of the quadrilateral are interpolated. The resulting $u = uw/w$ and $v = vw/w$ coordinates at each pixel represent the ray intersection with the uv quadrilateral. A similar procedure can be used to generate the (s, t) coordinates by drawing the st quadrilateral. Thus, the inverse transformation from (x, y) to (u, v, s, t) reduces essentially to two texture coordinate calculations per ray. This is cheap and can be done in real time, and is supported in many rendering systems, both hardware and software.

Only lines with (u, v) and (s, t) coordinates inside both quadrilaterals are represented in the light slab. Thus, if the texture coordinates for each plane are computed by drawing each quadrilaterial one after the other, then only those pixels that have both valid uv and st coordinates should be looked up in the light slab array. Alternatively, the two quadrilaterals may be simultaneously scan converted in their region of overlap to cut down on unnecessary calculations; this is the technique that we use in our software implementation.

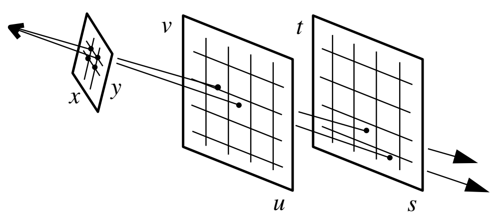


Figure 12: The process of resampling a light slab during display.

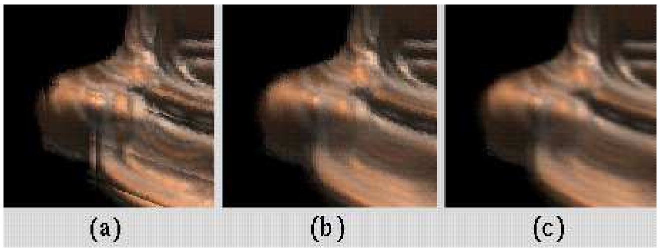


Figure 13: The effects of interpolation during slice extraction. (a) No interpolation. (b) Linear interpolation in uv only. (c) Quadralinear interpolation in uvst.

To draw an image of a collection of light slabs, we draw them sequentially. If the sets of lines in the collection of light slabs do not overlap, then each pixel is drawn only once and so this is quite efficient. To further increase efficiency, "back-facing" light slabs may be culled.

The second step involves resampling the radiance. The ideal resampling process first reconstructs the function from the original samples, and then applies a bandpass filter to the reconstructed function to remove high frequencies that may cause aliasing. In our system, we approximate the resampling process by simply interpolating the 4D function from the nearest samples. This is correct only if the new sampling rate is greater than the original sampling rate, which is usually the case when displaying light fields. However, if the image of the light field is very small, then some form of prefiltering should be applied. This could easily be done with a 4D variation of the standard mipmapping algorithm [Williams83].

Figure 13 shows the effect of nearest neighbor versus bilinear interpolation on the uv plane versus quadrilinear interpolation of the full 4D function. Quadralinear interpolation coupled with the proper prefiltering generates images with few aliasing artifacts. The improvement is particularly dramatic when the object or camera is moving. However, quadralinear filtering is more expensive and can sometimes be avoided. For example, if the sampling rates in the uv and st planes are different, and then the benefits of filtering one plane may be greater than the other plane.

6. Results

Figure 14 shows images extracted from four light fields. The first is a buddha constructed from rendered images. The model is an irregular polygon mesh constructed from range data. The input images were generated using RenderMan, which also provided the machinery for computing pixel and aperture antialiasing. The light field configuration was a single slab similar to that shown in figure 3a.

	buddha	**kidney**	**hallway**	**lion**
Number of slabs	1	1	4	4
Images per slab	16x16	64x64	64x32	32x16
Total images	256	4096	8192	2048
Pixels per image	256^2	128^2	256^2	256^2
Raw size (MB)	50	201	1608	402
Prefiltering	uvst	st only	uvst	st only

Table I: Statistics of the light fields shown in figure 14.

Our second light field is a human abdomen constructed from volume renderings. The two tan-colored organs on either side of the spine are the kidneys. In this case, the input images were orthographic views, so we employed a slab with one plane at infinity as shown in figure 4c. Because an orthographic image contains rays of constant direction, we generated more input images than in the first example in order to provide the angular range needed for creating perspective views. The images include pixel antialiasing but no aperture antialiasing. However, the dense spacing of input images reduces aperture aliasing artifacts to a minimum.

Our third example is an outward-looking light field depicting a hallway in Berkeley's Soda Hall, rendered using a radiosity program. To allow a full range of observer motion while optimizing sampling uniformity, we used four slabs with one plane at infinity, a four-slab version of figure 4c. The input images were rendered on an SGI RealityEngine, using the accumulation buffer to provide both pixel and aperture antialiasing.

Our last example is a light field constructed from digitized images. The scene is of a toy lion, and the light field consists of four slabs as shown in figure 3c, allowing the observer to walk completely around the object. The sensor and optical system provide pixel antialiasing, but the aperture diameter was too small to provide correct aperture antialiasing. As a result, the light field exhibits some aliasing, which appears as double images. These artifacts are worst near the head and tail of the lion because of their greater distance from the axis around which the camera rotated.

Table I summarizes the statistics of each light field. Table II gives additional information on the lion dataset. Table III gives the performance of our compression pipeline on two representative datasets. The buddha was compressed using a 2D tiling of the

Camera motion	
translation per slab	100 cm x 50 cm
pan and tilt per slab	90° x 45°
number of slabs	4 slabs 90° apart
total pan and tilt	360° x 45°
Sampling density	
distance to object	50 cm
camera pan per sample	3.6°
camera translation per sample	3.125 cm
Aperture	
focal distance of lens	10mm
F-number	f/8
aperture diameter	1.25 mm
Acquisition time	
time per image	3 seconds
total acquisition time	4 hours

Table II: Acquisition parameters for the lion light field. Distance to object and camera pan per sample are given at the center of the plane of camera motion. Total acquisition time includes longer gantry movements at the end of each row and manual setup time for each of the four orientations. The aperture diameter is the focal length divided by the F-number.

	buddha	lion
Vector quantization		
raw size (MB)	50.3	402.7
fraction in training set	5%	3%
samples per tile	2x2x1x1	2x2x2x2
bytes per sample	3	3
vector dimension	12	48
number of codewords	8192	16384
codebook size (MB)	0.1	0.8
bytes per codeword index	2	2
index array size (MB)	8.4	16.8
total size (MB)	8.5	17.6
compression rate	6:1	23:1
Entropy coding		
gzipped codebook (MB)	0.1	0.6
gzipped index array (MB)	1.0	2.8
total size (MB)	1.1	3.4
compression due to gzip	8:1	5:1
total compression	45:1	118:1
Compression performance		
training time	15 mins	4 hrs
encoding time	1 mins	8 mins
original entropy (bits/pixel)	4.2	2.9
image quality (PSNR)	36	27

Table III: Compression statistics for two light fields. The buddha was compressed using 2D tiles of RGB pixels, forming 12-dimensional vectors, and the lion was compressed using 4D tiles (2D tiles of RGB pixels from each of 2 x 2 adjacent camera positions), forming 48-dimensional vectors. Bytes per codeword index include padding as described in section 4. Peak signal-to-noise ratio (PSNR) is computed as $10\log_{10}(255^2/MSE)$.

light field, yielding a total compression rate of 45:1. The lion was compressed using a 4D tiling, yielding a higher compression rate of 118:1. During interactive viewing, the compressed buddha is indistinguishable from the original; the compressed lion exhibits some artifacts, but only at high magnifications. Representative images are shown in figure 15. We have also experimented with higher rates. As a general rule, the artifacts become objectionable only above 200:1.

Finally, table IV summarizes the performance of our interactive viewer operating on the lion light field. As the table shows, we achieve interactive playback rates for reasonable image sizes. Note that the size of the light field has no effect on playback rate; only the image size matters. Memory size is not an issue because the compressed fields are small.

7. Discussion and future work

We have described a new light field representation, the light slab, for storing all the radiance values in free space. Both inserting images into the field and extracting new views from the field involve resampling, a simple and robust procedure. The resulting system is easily implemented on workstations and personal computers, requiring modest amounts of memory and cycles. Thus, this technique is useful for many applications requiring interaction with 3D scenes.

Display times (ms)	**no bilerp**	**uv lerp**	**uvst lerp**
coordinate calculation	13	13	13
sample extraction	14	59	214
overhead	3	3	3
total	30	75	230

Table IV: Display performance for the lion light field. Displayed images are 192 x 192 pixels. Sample extraction includes VQ decoding and sample interpolation. Display overhead includes reading the mouse, computing the observer position, and copying the image to the frame buffer. Timings are for a software-only implementation on a 250 MHz MIPS 4400 processor.

There are three major limitation of our method. First, the sampling density must be high to avoid excessive blurriness. This requires rendering or acquiring a large number of images, which may take a long time and consume a lot of memory. However, denser sample spacing leads to greater inter-sample coherence, so the size of the light field is usually manageable after compression. Second, the observer is restricted to regions of space free of occluders. This limitation can be addressed by stitching together multiple light fields based on a partition of the scene geometry into convex regions. If we augment light fields to include Z-depth, the regions need not even be convex. Third, the illumination must be fixed. If we ignore interreflections, this limitation can be addressed by augmenting light fields to include surface normals and optical properties. To handle interreflections, we might try representing illumination as a superposition of basis functions [Nimeroff94]. This would correspond in our case to computing a sum of light fields each lit with a different illumination function.

It is useful to compare this approach with depth-based or correspondence-based view interpolation. In these systems, a 3D model is created to improve quality of the interpolation and hence decrease the number of pre-acquired images. In our approach, a much larger number of images is acquired, and at first this seems like a disadvantage. However, because of the 3D structure of the light field, simple compression schemes are able to find and exploit this same 3D structure. In our case, simple 4D block coding leads to compression rates of over 100:1. Given the success of the compression, a high density compressed light field has an advantage over other approaches because the resampling process is simpler, and no explicit 3D structure must be found or stored.

There are many representations for light used in computer graphics and computer vision, for example, images, shadow and environment maps, light sources, radiosity and radiance basis functions, and ray tracing procedures. However, abstract light representations have not been systematically studied in the same way as modeling and display primitives. A fruitful line of future research would be to reexamine these representations from first principles. Such reexaminations may in turn lead to new methods for the central problems in these fields.

Another area of future research is the design of instrumentation for acquisition. A large parallel array of cameras connected to a parallel computer could be built to acquire and compress a light field in real time. In the short term, there are many interesting engineering issues in designing and building gantries to move a small number of cameras and lights to sequentially acquire both inward- and outward-looking light fields. This same instrumentation could lead to breakthroughs in both 3D shape acquisition and reflection measurements. In fact, the interaction of light with any object can be represented as a higher-dimensional interaction matrix; acquiring, compressing, and manipulating such representations are a fruitful area for investigation.

8. Acknowledgements

We would like to thank David Addleman and George Dabrowski of Cyberware for helping us design and build the camera gantry, Craig Kolb and James Davis for helping us calibrate it, Brian Curless for scanning the buddha, Julie Dorsey for shading it and allowing us to use it, Carlo Sequin for the Soda Hall model, Seth Teller, Celeste Fowler, and Thomas Funkhauser for its radiosity solution, Lucas Pereira for rendering it, Benjamin Zhu for reimplementing our hardware-accelerated viewer in software, and Navin Chaddha for his vector quantization code. We also wish to thank Eric Chen and Michael Chen for allowing us to examine the Apple ObjectMaker, and Alain Fournier and Bob Lewis for showing us their wavelet light field work. Finally, we wish to thank Nina Amenta for sparking our interest in two-plane parameterizations of lines, Michael Cohen for reinforcing our interest in image-based representations, and Gavin Miller for inspiring us with his grail of volumetric hyperreality. This work was supported by the NSF under contracts CCR-9157767 and CCR-9508579.

9. References

[Adelson91] Adelson, E.H., Bergen, J.R., "The Plenoptic Function and the Elements of Early Vision," In *Computation Models of Visual Processing*, M. Landy and J.A. Movshon, eds., MIT Press, Cambridge, 1991.

[Ashdown93] Ashdown, I., "Near-Field Photometry: A New Approach," *Journal of the Illuminating Engineering Society*, Vol. 22, No. 1, Winter, 1993, pp. 163-180.

[Beers96] Beers, A., Agrawala, M., Chaddha, N., "Rendering from Compressed Textures." In these proceedings.

[Benton83] Benton, S., "Survey of Holographic Stereograms," *Processing and Display of Three-Dimensional Data*, Proc. SPIE, Vol. 367, 1983.

[Blinn76] Blinn, J.F., Newell, M.E., "Texture and Reflection in Computer Generated Images," *CACM*, Vol. 19, No. 10, October, 1976, pp. 542-547.

[Bolles87] Bolles, R., Baker, H., Marimont, D., "Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion," *International Journal of Computer Vision*, Vol. 1, No. 1, 1987, pp. 7-55.

[Chen93] Chen, S.E., Williams, L., "View Interpolation for Image Synthesis," Proc. SIGGRAPH '93 (Anaheim, California, August 1-6, 1993). In *Computer Graphics Proceedings*, Annual Conference Series, 1993, ACM SIGGRAPH, pp. 279-288.

[Chen95] Chen, S.E., "QuickTime VR — An Image-Based Approach to Virtual Environment Navigation," Proc. SIGGRAPH '95 (Los Angeles, CA, August 6-11, 1995). In *Computer Graphics* Proceedings, Annual Conference Series, 1995, ACM SIGGRAPH, pp. 29-38.

[Fuchs94] Fuchs, H., Bishop, G., Arthur, K., McMillan, L., Bajcsy, R., Lee, S.W., Farid, H., Kanade, T., "Virtual Space Teleconferencing Using a Sea of Cameras," *Proc. First International Conference on Medical Robotics and Computer Assisted Surgery*, 1994, pp. 161-167.

[Gersho92] Gersho, A., Gray, R.M., *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1992.

[Gershun36] Gershun, A., "The Light Field," Moscow, 1936. Translated by P. Moon and G. Timoshenko in *Journal of Mathematics and Physics*, Vol. XVIII, MIT, 1939, pp. 51-151.

[Gortler96] Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M., "The Lumigraph." In these proceedings.

[Greene86] Greene, N., "Environment Mapping and Other Applications of World Projections," *IEEE Computer Graphics and Applications*, Vol. 6, No. 11, November, 1986, pp. 21-29.

[Greene94] Greene, N. and Kass, M., "Approximating Visibility with Environment Maps," Apple Technical Report No. 41, November, 1994.

[Halle94] Halle, M., "Holographic Stereograms as Discrete Imaging Systems." *Practical Holography*, Proc. SPIE, Vol. 2176, February, 1994.

[Katayama95] Katayama, A., Tanaka, K., Oshino, T., Tamura, H., "Viewpoint-Dependent Stereoscopic Display Using Interpolation of Multiviewpoint Images," *Stereoscopic Displays and Virtual Reality Systems II*, Proc. SPIE, Vol. 2409, S. Fisher, J. Merritt, B. Bolas eds. 1995, pp. 11-20.

[Laveau94] Laveau, S., Faugeras, O.D., "3-D Scene Representation as a Collection of Images and Fundamental Matrices," INRIA Technical Report No. 2205, 1994.

[Levin71] Levin, R., "Photometric Characteristics of Light Controlling Apparatus," *Illuminating Engineering*, Vol. 66, No. 4, 1971, pp. 205-215.

[McMillan95a] McMillan, L., Bishop, G., "Head-Tracked Stereoscopic Display Using Image Warping," *Stereoscopic Displays and Virtual Reality Systems II*, Proc. SPIE, Vol. 2409, S. Fisher, J. Merritt, B. Bolas eds. 1995, pp. 21-30.

[McMillan95b] McMillan, L., Bishop, G., Plenoptic Modeling: An Image-Based Rendering System, Proc. SIGGRAPH '95 (Los Angeles, CA, August 6-11, 1995). In *Computer Graphics* Proceedings, Annual Conference Series, 1995, ACM SIGGRAPH, pp. 39-46.

[Miller95] Miller, G., "Volumetric Hyper-Reality: A Computer Graphics Holy Grail for the 21st Century?," *Proc. Graphics Interface '95*, W. Davis and P. Prusinkiewicz eds., Canadian Information Processing Society, 1995, pp. 56-64.

[Moon81] Moon, P., Spencer, D.E., *The Photic Field*, MIT Press, 1981.

[Narayanan95] Narayanan, P.J., "Virtualized Reality: Concepts and Early Results," *Proc. IEEE Workshop on the Representation of Visual Scenes*, IEEE, 1995.

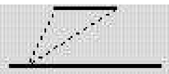
[Nimeroff94] Nimeroff, J., Simoncelli, E., Dorsey, J., "Efficient Re-rendering of Naturally Illuminated Scenes," *Proc. Fifth Eurographics Rendering Workshop*, 1994, pp. 359-373.

[Sbert93] Sbert, A.M., "An Integral Geometry Based Method for Form-Factor Computation," *Computer Graphics Forum*, Vol. 13, No. 3, 1993, pp. 409-420.

[Seitz95] Seitz, S., Dyer, C., "Physically-Valid View Synthesis by Image Interpolation," *Proc. IEEE Workshop on the Representation of Visual Scenes*, IEEE, 1995.

[Williams83] Williams, L., "Pyramidal Parametrics," *Computer Graphics (Proc. Siggraph '83)*, Vol. 17, No. 3, July, 1983, pp. 1-11.

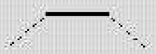
[Ziv77] Ziv, J., Lempel, A., "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, IT-23:337-343, 1977.



(a) Buddha.

Input images: RenderMan renderings from an irregular polygon mesh model.

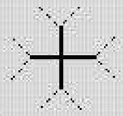
Light field: one slab, shown above in plan view. See also figure 3a.



(b) Kidney.

Input images: volume renderings from computed tomography data.

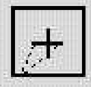
Light field: one slab, shown above. See also figure 4c.



(c) Hallway.

Input images: SGI RealityEngine renderings from a polygon model.

Light field: four slabs like figure 4c, arranged in a cross.



(d) Lion.

Input images: digitized video from a computer-controlled camera

Light field: four slabs arranged in a box. See also figure 3c.

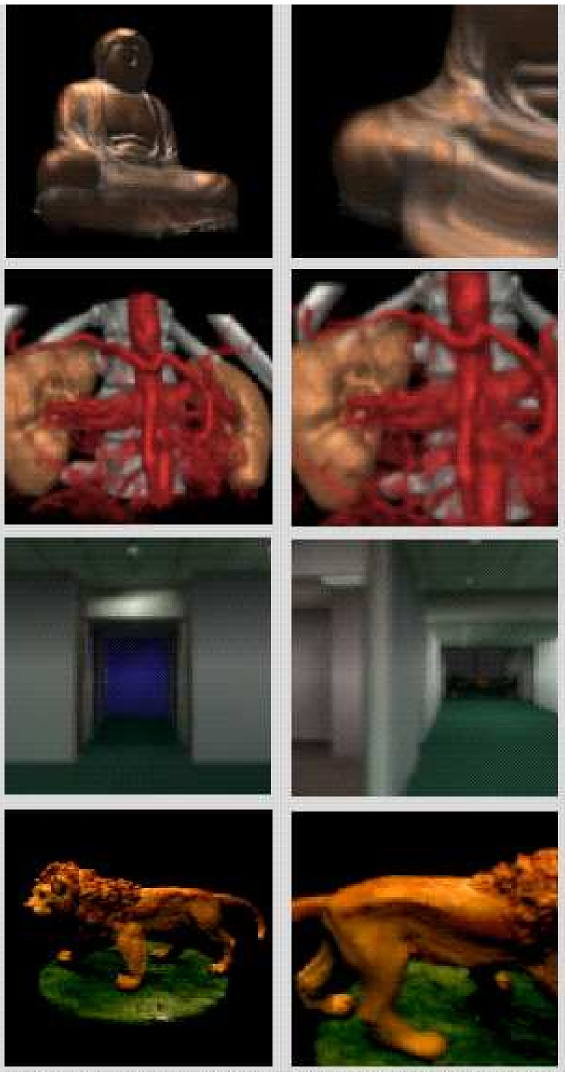
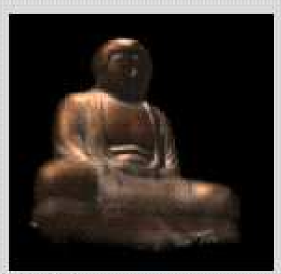


Figure 14: Example images from four light fields, extracted during a typical interactive viewing session.



(a) Buddha. *Vector dimension:* 12 *Compression:* 45:1

(b) Lion. *Vector dimension:* 48 *Compression:* 118:1

Figure 15: Images extracted from compressed light fields.