Approximating the Plenoptic Function

Jonathan Shade

1 December 2002
UW CSE Technical Report 02-12-06

Abstract

The full plenoptic function describes all of the light passing through some volume of space. This information has at least seven dimensions: a ray of light requires five numbers to locate it (a point in space and a direction), plus wavelength and time. Given a digital representation of the plenoptic function, we can reconstruct arbitrary views of a scene. Unfortunately, adequately sampling the seven dimensions of the full plenoptic function requires an unmanageable amount of storage. Even if we assume we are concerned with a snapshot in time and that light is monochromatic, we are still faced with the five spatial dimensions of a ray. At the other end of the spectrum, if we consider only a single point in space then we can reconstruct just one view (a 2D photograph). In between the 2D photograph and 5D plenoptic function are many possibilities that provide more freedom than a single image, but less than the full freedom and consequent complexity of capturing and manipulating a 5D object. Recently, there has been considerable research on this topic, typically under the name image-based modeling and rendering (IBMR). We review and classify this work based on the structure of the representation used to approximate the 5D plenoptic function.

1 Introduction

One of the primary goals of computer graphics has been the creation of interactive photorealistic imagery. Unfortunately, the dual goals of interactivity and photorealism are at odds with each other. The process of creating physics-based photorealistic images has traditionally involved the simulation of the propagation of light through an environment. To do this, we must model the geometry of the objects in the environment, the properties of the materials that constitute the geometry, the light sources that emit energy into the environment, and the cameras or retinas that record the light. A computationally expensive light transport algorithm, such as ray tracing [38] or radiosity [11, 7, 26] is then used to compute the distribution of light energy throughout the scene. Modeling a scene that resembles the real world with this process is exceedingly difficult because the complexity of the geometry of the real world is so great that traditional geometric representations result in data sets so large that no computer can hold them in memory and no simulation of light transport will finish in a reasonable amount of time. Even if the simulation of light transport could be made fast enough, producing such complex geometric models is a very difficult and prohibitively time-consuming task. At the other end of the spectrum, interactive graphics has focused on hardware implementations of rendering algorithms. In order to achieve interactive rates these systems use a comparatively low level of geometric detail and local instead of global illumination. While the complexity of scenes that can be rendered by interactive systems is continually increasing, they are still a long way from producing photorealistic images.

Since the goal is producing photorealistic images, in the end, what matters is the final result of an image synthesis algorithm: the light that enters the eye of the observer. If we had a way of capturing, storing, and rendering this final result, light, then we could use an image synthesis algorithm just once: to create the light-based data structure. This observation is the basis of the field of image-based modeling and rendering (IBMR). In IBMR, images (samples of light) are used as the fundamental modeling primitive. Rendering from the sampled representation consists of reprojecting and interpolating the samples of light, a process that can be done at interactive rates. One of the consequences of using images as a modeling primitive is that IBMR algorithms are, for the most part, agnostic with respect to how the images are made. Because of this, images of the real world work just as well as images of synthetic environments. Thus, IBMR provides a framework for producing interactive photorealistic renderings of both real and synthetic environments.

2 Image-Based Representations

Image-based models are sampled representations of light. Specifically, they sample the plenoptic function. As defined by Adelson and Bergen [1], the plenoptic funtion has at least seven dimensions: three for position, two for direction, one for wavelength, and one for time. In other words, for every point in space, every direction one could look, every point in time, and at every wavelength of light, there is some measurable amount of electromagnetic energy. Humans perceive color because sensors in the retinas of our eyes respond to stimulation by energy at certain wavelengths of light and produce electrical signals that are sent our brains. Simply put, the goal of image-based modeling and rendering is to capture, store, and reconstruct the energies of light in such a way that a human looking at a computer-driven display sees the same thing they would have seen had they been in the environment that was captured. The details of capturing the light energy of an environment and the subsequent display to a human observer are beyond the scope of this paper; our focus is limited to the representation used to store the samples of light energy. However, to be precise in our discussion of image-based representations we must define the fundamental measure of light energy used in computer graphics, radiance.

The terminology used in discussing quantified light energy comes from the field of *radiometry*. Radiometric measurements are expressed in the SI units for energy and power, joules and watts. Radiometric terms, like the plenoptic function, are a function of wavelength, time, position, and direction. Likewise, we make some assumptions to reduce the

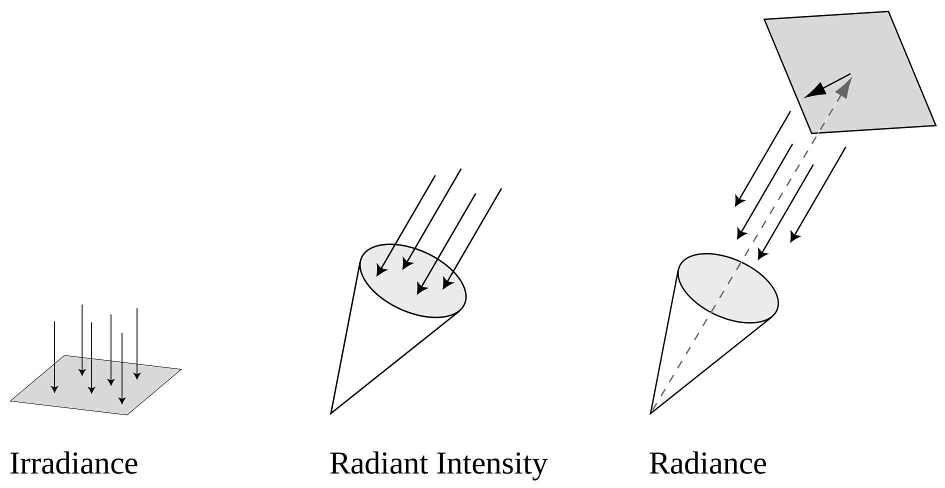

Figure 1: Radiometric quantities.

dimension of these terms. First, we assume that energy at different wavelengths is decoupled. In other words, for some point in space and some direction, the energy at any one wavelength is independent of the energies at all other wavelengths. Second, we assume that there is no time-varying behavior in the environment. In other words, light travels infinitely fast and the propagation of light energy in the environment has reached a steady state. Using these two assumptions, radiometric quantities can be expressed in terms of five variables: three for position and two for direction. The fundamental radiometric quantity *radiant energy*, Q, is measured in joules. In the particle model of light, every photon carries some number of joules of energy. If we count the number of photons, and the energy they carry, as they pass through a surface during a unit of time, we'll measure flux. In radiometry, the flux at a surface is called the *radiant power* or *radiant flux*, and is measured in watts, or joules per second:

$$\Phi = dQ/dt \qquad [W]$$

If we want to measure the flux per unit of surface area, we divide differential radiant flux by the differential area it passes through, and we get *radiant flux area density*:

$$u = d\Phi/dA \qquad \left[\frac{W}{m^2}\right]$$

The area measured doesn't have to belong to a surface, it can be any arbitrary plane in space. This quantity tells us how much energy passes through an area, but it doesn't tell us if the energy is arriving or leaving. This distinction is important in the mathematics of light transport, so special terms have been given to the two cases: if energy is arriving, it is called *irradiance*, and if it is leaving, it is called *radiant exitance* or *radiosity*. There are times when we need to measure the energy flowing through a point in space. In this case, the differential area would be zero, and the radiant flux area density would be undefined. Fortunately, we can reformulate this measurement in terms of the solid angle of the beam through which the light is arriving at or leaving from a point:

$$I = d\Phi / d\overrightarrow{\omega} \qquad \left[\frac{W}{sr}\right]$$

We call this quantity *radiant intensity*. Lastly, we often want to measure not just the energy arriving at or leaving from a surface, but the amount of energy leaving an area that arrives at a point, or conversely, the amount of energy leaving a point and arriving at an area. This quantity, the power arriving at or leaving from a surface per unit solid angle per unit projected area is called *radiance*:

$$L = \frac{d^2\Phi}{dA^{\Phi} d\overrightarrow{\omega}} \qquad \left[\frac{W}{sr \cdot m^2}\right]$$

Radiance is different from the other quantities because the projected area term adds a cosine factor. Consider an area being viewed by a point through some solid angle. In general, the area being viewed is not perpendicular to the ray running through the center of the solid angle, and we must project the area onto the direction of the ray in order to accurately measure the energy leaving this area in the direction of the ray (see Figure 1).

One of the important properties of radiance is encapsulated in what is called the *ray law*: the radiance in the direction of a light ray remains unchanged as it propagates along the ray. This means that, in the absence of occluding geometry or participating media, if two surfaces can see each other, the radiance leaving one surface must equal the radiance arriving at the other. Because of this, radiance is the quantity associated with rays in a

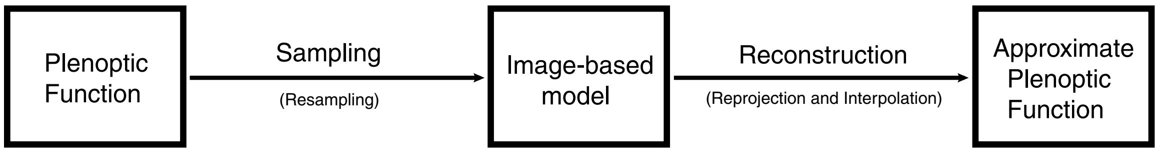

Figure 2: Image-based modeling and rendering pipeline.

physics-based light transport algorithm. A second important property of radiance is that the response of a sensor is proportional to the radiance of the surface visible to the sensor. Because of this, the radiance from a surface to the eye is the quantity that is recorded in an image that is created by a light transport algorithm. And likewise, radiance is the quantity measured by a camera when photographs of the real world are taken. Finally, radiance is the quantity stored in all image-based representations.

The process of image-based modeling and rendering can be broken down into two major stages as depicted in Figure 2: sampling and reconstruction. Sampling is typically conducted in one of two ways. In systems that use pictures of the real world, a digital camera or videocamera is used to record the environment of interest. This data is then resampled to fit into the organization of the chosen image-based representation. If a synthetic scene is being sampled, ray casting using an image-synthesis scheme can be used to sample the scene directly into the intended representation, thus bypassing the resampling step. Since we are primarily concerned with the structure of image-based representations and not how they are created, we are going to ignore the mechanics of how the sampling is performed (whether pixels are created using point sampling, area sampling, or some other method). What is important to us is the number of samples that must be taken, and how they are organized.

Sampling in IBMR is often reconstruction-driven: enough samples are taken to guarantee some measure of quality at reconstruction time. There are a wide variety of techniques used to reconstruct the plenoptic function from a sampled representation, and they are invariably tied to the representation chosen. For this reason, we will discuss the details of reconstruction on a system-by-sytem basis in Section 3. For now, we will use a simple scheme to illustrate the point. Suppose we wanted to capture the plenoptic function for an average graduate student office. The office is 5 meters wide and 7 meters long. We'll represent the plenoptic function by capturing a panoramic image of the office at the nodes of a regular grid with one centimeter spacing. To simplify the task, we'll capture just a plane of images, at the height of an average person. To reconstruct the plenoptic function, we'll display a different image to each eye of an observer (human eyes are spaced about three inches apart). The observer will be allowed to walk around and turn (but not tilt or nod) their head. This will be a fairly crude reconstruction since we allow the observer to turn their head: their eyes won't always fall exactly on a grid point. For now we'll ignore that and assume that displaying the image from the closest grid point is good enough. How much storage do we need for this data set? Let's assume that each eye of the observer will get an image that is 256 pixels wide with a field of view of 90 degrees. The panoramic image thus needs to be 1000 pixels wide. Since the head can't tilt, we can limit the vertical resolution to 500 pixels. A grid with centimeter spacing over a room 5 meters wide and 7 meters long has 700x500 grid points, and thus 350,000 panoramic images. Capturing 24-bit images means each panoramic image requires 1.5 MB of space. If we store the images using the JPEG format, we can assume a 20:1 compression ratio. So, the data set of 350,000 images at 75 Kb apiece requires 26 GB of space. This is clearly an unmanageabe amount of data.

Sampled representations are, by nature, a trade-off in favor of increased size for less computation. Image-based representations are essentially precomputed evaluations of the the plenoptic function stored in a table. As the example above illustrates, there is a direct correlation between the dimension of a function and the size of the domain over which it is represented, and the amount of space required to store a sampled representation of it. As a consequence, all image-based models make simplifying assumptions in order to reduce the size of the representation used to approximate the plenoptic function. We have identified four simple principles IBMR systems have used to pare down the plenoptic function:

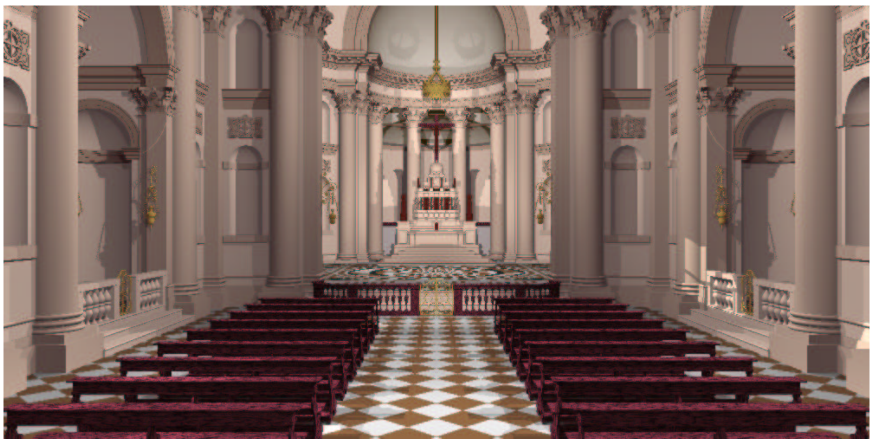

Figure 3: Planar perspective image.

Taking a subset of the plenoptic function. Obviously, a sampled representation must have a finite domain. In this respect, all image-based representations sample a subset of the plenoptic function. As the office exampled showed, representing the plenoptic function in even a small environment requires a huge data set. Because of this, image-based representations that sample the plenoptic function uniformly are typically limited to a very small spatial domain.

Taking a slice of the plenoptic function. By throwing away a degree of freedom in choosing samples, we take a slice through the plenoptic function. This effectively reduces the size of the representation, but at the cost of also reducing by a degree of freedom the space of views that can be reconstructed. In the office example, we restricted the panoramic images to lie on a plane. This reduced the degrees of freedom in position from three to two. The representation was therefore a 4D slice of the 5D plenoptic function.

Sparsely sampling a parameter. By sampling a parameter (a degree of freedom) sparsely instead of densely, we are taking a nonuniform subset of the function. This can be a versatile technique; instead of throwing away a whole dimension, we throw away most of it, but keep the interesting or useful part. Applying this to the office example, we may choose to

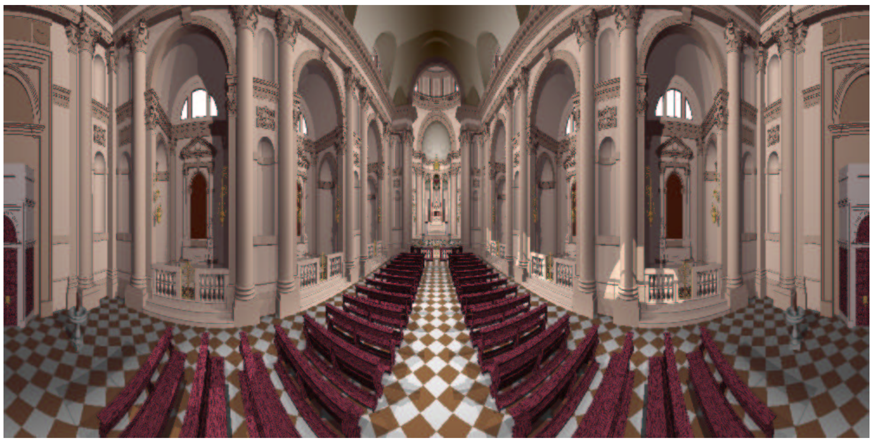

Figure 4: Cylindrical image.

sample the grid of panoramic images only inside each cubicle. Movement between cubicles would be handled by “teleporting” the viewer. Using this technique, we can represent larger environments with collections of densely sampled data sets placed at strategic points, or along paths. We designate representations that use this strategy with a fractional dimension. For instance, the sparse sampling of the office would be a 3.5D representation.

Eliminating redundancy. Another way to reduce the amount of data in the plenoptic function is to find and eliminate redundancy in the function. The *lumigraph* [12] and *light field* [18] representations do this by exploiting the ray law. By assuming the viewer is in a region of space free of geometry or participating media, the light entering this space can be described by a 4D instead of 5D function. This observation assumes a constancy in behavior of the light being sampled. This technique is elegant because the assumption of constancy is physics-based and correct. No information is lost in reducing the plenoptic function from five to four dimensions.

This paper presents a classification of image-based representations in two ways. First, we review the literature and show how these four principles have been applied. Second, we take a systematic approach to applying these four principles. As Tables 2 and 3 show,

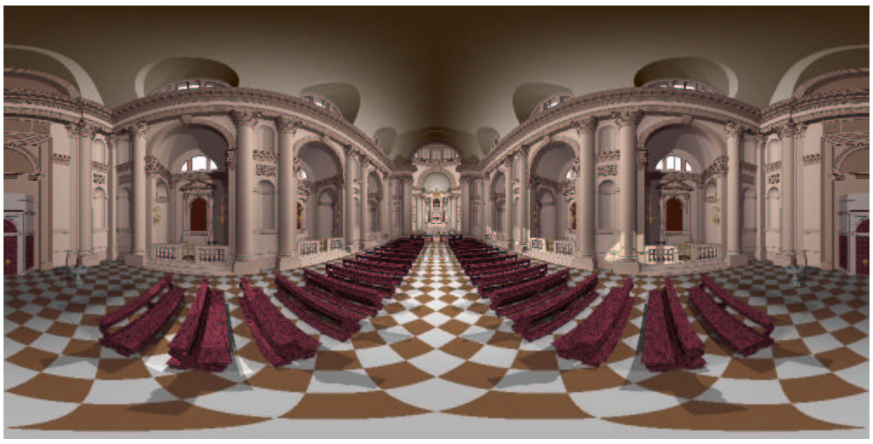

Figure 5: Spherical image.

we examine the variety of representations that can be made by working our way up from the singular 0D representation, point samples, through 1D, 2D, 2.5D, 3D, 3.5D, and 4D representations.

3 Overview of Representations

There has been a great deal of work in image-based modeling and rendering. This section gives an overview of some of the significant papers.

3.1 Images

There are many ways to approximate the plenoptic function; let's start with a familiar example, the 2D image. The most general definition of an image is: every ray of light that passes through a point in space (the center of projection of the image). By choosing a single point in space, we have fixed three of the degrees of freedom of the 5D plenoptic function. How we parameterize the remaining two parameters (the directions of the rays of incoming light) defines the kind of image we will make. Four common choices are: planar, cylindrical, spherical, and cubical. Figures 3 through 6 show examples of each type of image.

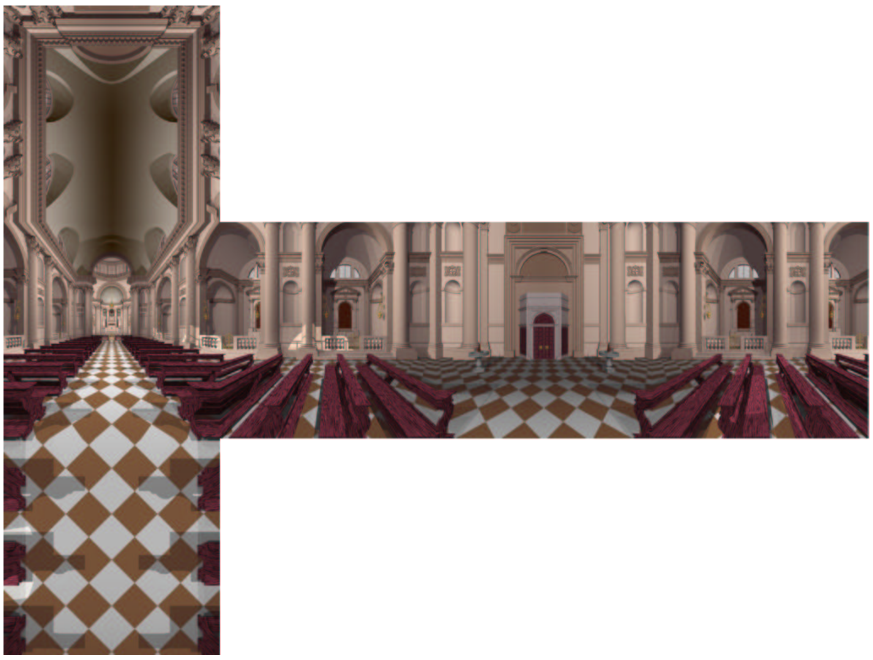

Figure 6: Cubical image.

The cylindrical, spherical and cubical projections are often called panoramic images. The cubical projection has become synonymous with the term *environment map* in the computer graphics literature.

By sampling the plenoptic function from just a single point (though over all directions), we have not only greatly reduced the size of the function sampled, but also the freedom we have in reconstructing the function. Strictly speaking, there is only one reconstruction possible, viewing the image from its center of projection. If the image is panoramic and we are viewing it with a planar projection, we can pan across the image, but we can't move the viewpoint. If we want to use the image to render a view of the same scene from a nearby viewpoint, we can produce an *approximate* reconstruction of the plenoptic function by assuming some depth for each point in the image. If we assume the same depth for each point, we are mapping our 2D image onto a plane in 3D. We can then view the image from a new viewpoint by projecting the plane to the novel viewpoint. This is something everyone has experienced when tilting a photograph back and forth in an attempt

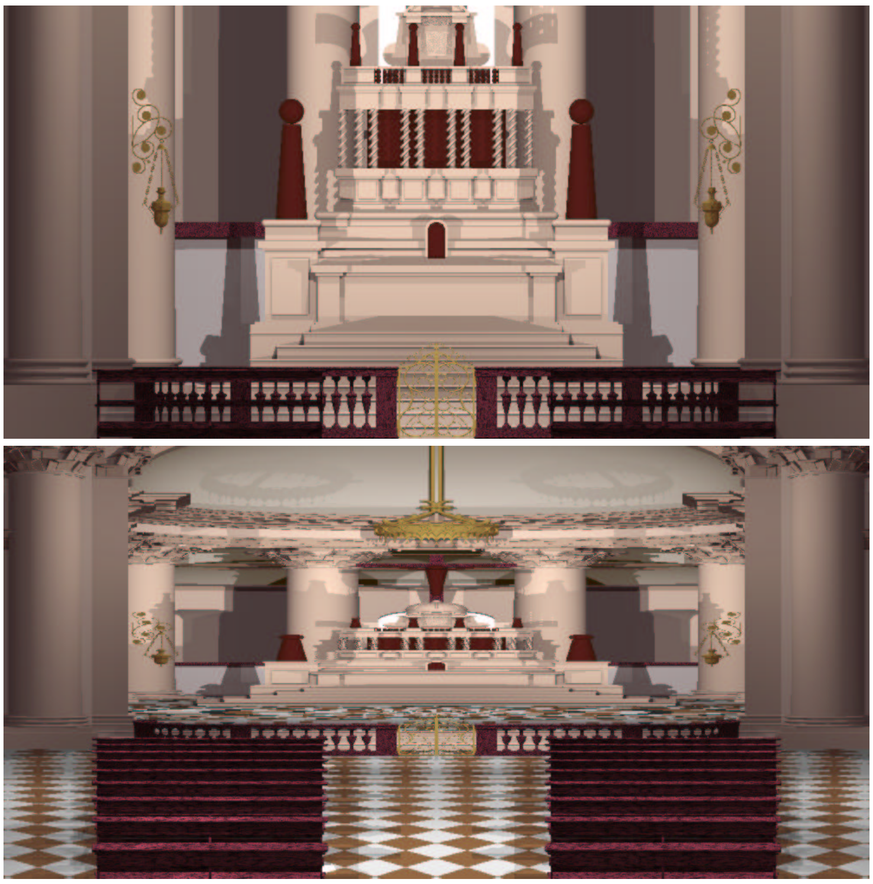

Figure 7: Planar orthographic (top) and line-perspective (bottom) images.

to breathe life into the image. Section 3.3 explores this line of thought and the issues that arise when associating depth with images. In the rest of this section, we'll look at the variety of representations that arise when we associate more than one center of projection with a single 2D image.

Standard images have a single center of projection. In some instances, it is advantageous to use more than one center of projection within a single image. Multiperspective images (MPIs) can be made in many different ways. A familiar example is the planar orthographic image (see Figure 7). An orthographic image can be thought of in two ways. The first is that it is a perspective image with a focal plane that is infinitely far away. The second is that it is a multiperspective image where each pixel has associated with it a unique

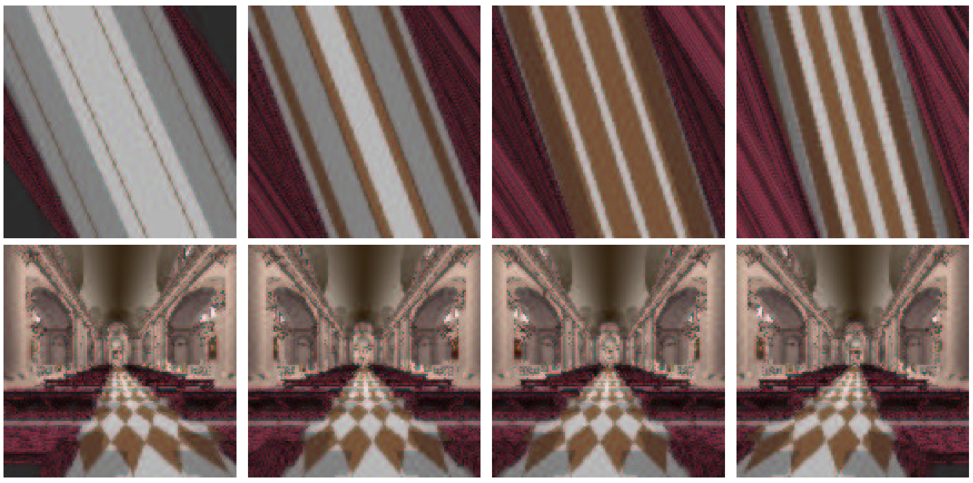
Figure 8: Epipolar plane images. The bottom row shows 4 out of a set of 128 images taken as the camera translated left-to-right. The top row shows 4 epipolar plane images built from scanlines near the bottom of the input images..

camera such that the pixel represents the ray going through the center of the camera's image plane. Along this ray there is no perspective foreshortening. Another way to create a multiperspective image is to use a line of cameras all facing the same direction. The MPI is created by taking the central vertical line of pixels from each camera (see Figure 7). This vertical slit camera structure is similar to images used in a lenticular display [30]. The first system we review use images built not from vertical slits, but from horizontal slits.

Katayama et al. [15] presented a system that could display view-dependent stereoscopic reconstructions of a real scene. A series of images of a scene was acquired by translating a camera horizontally along a rigid slide stage. The orientation of the camera was fixed to be at a right angle to the direction of its movement. After the images were captured, they were factored into a representation called Epipolar Plane Images [4]. An EPI is derived from a series of images by extracting the same horizontal line from each image, stacking them bottom to top in the EPI. Thus, there are the same number of EPIs as there are horizontal lines of resolution in the camera that captured the scene, and the EPI vertical resolution is equal to the number of original images. Figure 8 shows an example of a set of input

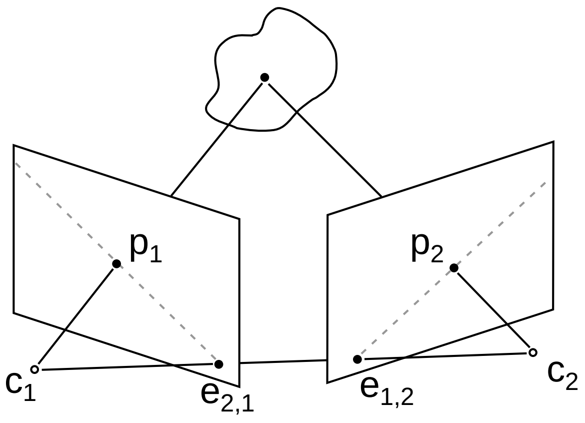

Figure 9: Epipolar geometry.

images and the EPIs created from them. If we stacked each of the original images to create a 3D volume with camera position as the third dimension, an EPI is simply a horizontal slice through this volume. EPIs were originally created by Bolles et al. [4] to automatically extract depth in stereo pairs of images. Figure 9 shows the epipolar geometry for two images and a point in a scene. The image plane of each of the two cameras shows two points: p is the projection of the scene point, and e, the *epipole*, is the projection of the other camera's center of projection. The plane formed by the point in the scene and the two camera centers is called the *epipolar plane*, and the intersection of this plane with the image plane of each of the camera centers forms an *epipolar line*. Epipolar lines have been used to automatically compute point correspondances between images [24], to automatically compute per-pixel depth [4], and as a tool to resolve visibility when using 3D image warping [23]. An interesting property of epipolar lines is that for a given pair of cameras, the set of epipolar lines for all points in the scene radiate in a fan-like configuration about the epipole, as shown in Figure 10(a). If the two cameras are parallel (positioned adjacent to one another and looking in the same direction), then the epipoles project to infinity and the epipolar lines become parallel, as shown in Figure 10(b). The representation built by Katayama uses this special configuration.

The EPIs used by Katayama capture the parallax in a scene: the lines in an EPI trace the

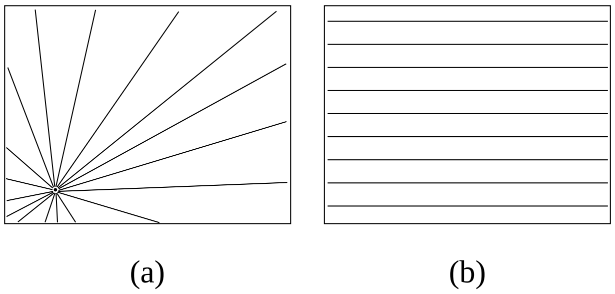

Figure 10: (a) Pencil of epipolar planes (b) epipole at infinity.

movement of points in the scene. If a point is moving very quickly across the field of view of the camera as it translates, the line it produces will have a steep slope. As a consequence, the slope of a line in an EPI is directly proportional to the depth of the point in the scene that traces out that line. Bolles et al. [4] used this observation to automatically calculate depth for acquired images. Katayama used the slope of these lines to create interpolated novel views of the scene that lie between the acquired images. Each horizontal line in an EPI represents a different horizontal camera position; so, to create a novel view, all we have to do is create an intermediate horizontal line in an EPI that correctly interpolates the trace lines that cross it. This representation is a 3D slice of the plenoptic function: a set of 2D images taken along a linear path. The space of reconstruction views is limited to those that lie on the line of the input cameras and look in he same direction as the original cameras.

Wood et al. [40] used multiperspective images to create distorted panaoramas for use in cel animation. Cel animation is fundamentally a 2D process where a number of layers of paintings are composited in a back-to-front order to compose a pseudo 3D scene. Translating the layers at different rates produces an approximation to the parallax that would be seen in a true 3D scene. Long sweeping camera motions can be produced by panning a camera over a large distorted panorama of a scene. The panorama shown in Figure 11 portrays a sweeping camera motion simulating the view from a helicopter as it rotates and descends into a city. For every frame of an animation, a small window is cropped from the source panaorama. As the window is moved around the panorama a convincing 3D animation is produced. In Figure 11, if we start at the upper right of the image and rotate a crop

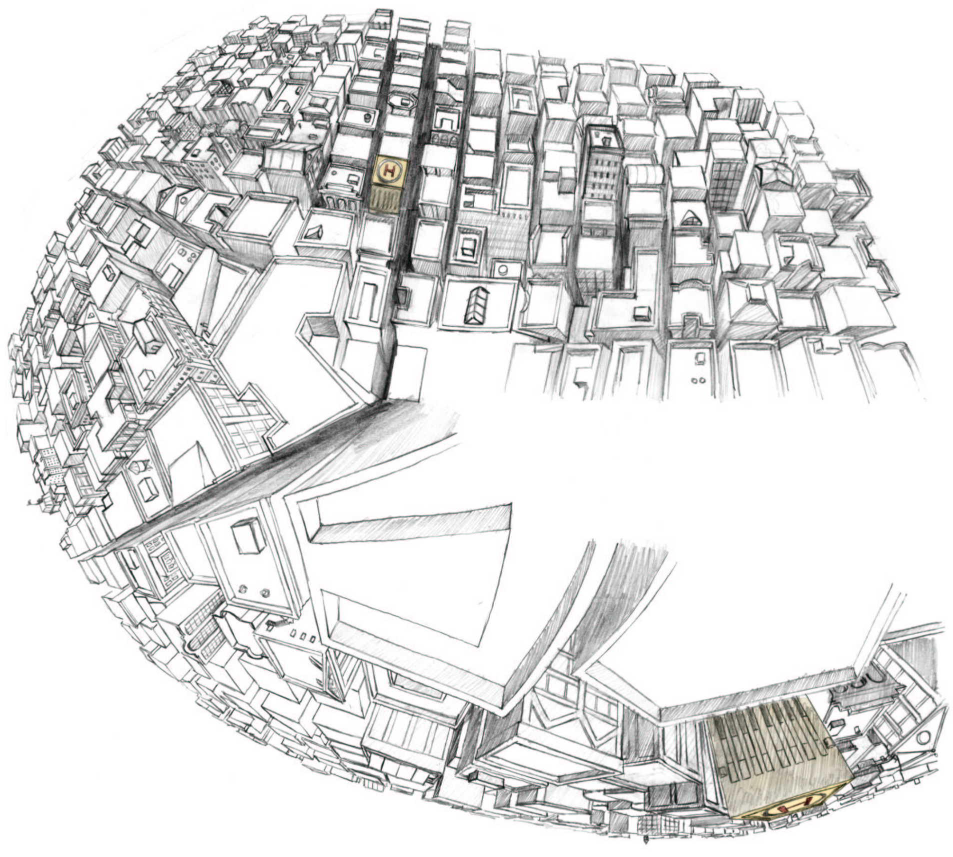

Figure 11: A multiperspective panorama. From Wood et al. [40].

window in a counter-clockwise fashion we'll get the illusion that the camera is rotating and descending towards the city.

In order to do this convincingly, implausible and unintuitive perspectives often have to be employed to make the transitions work. For instance, in Figure 11 we see the yellow-shaded building twice, with different perspectives. The work presented by Wood et al. sought to automate this process by creating a rough draft of the warped panorama using 3D geometric models. The computer-generated panorama was then used as a guide by an artist who painted the final panorama by hand. The warped panorama is a multiperspective image created using a series of vertical wedges from the central areas of a camera placed along a space curve. Since the source cameras are not constrained to linear horizontal motion,

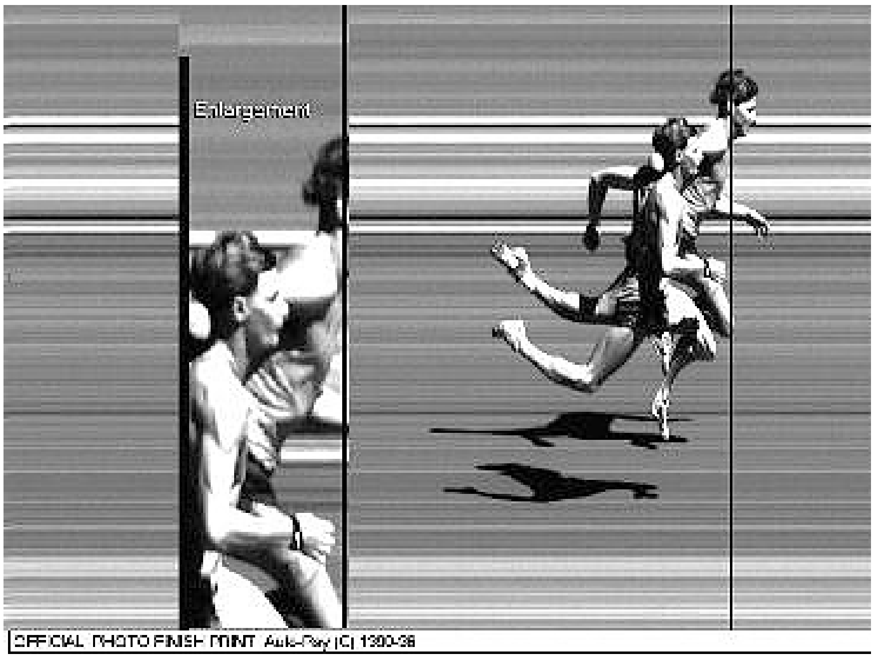

Figure 12: A single-perspective multi-time image. Used by permission of the Auto-Ray Corporation.

taking a single vertical slit from each image is not sufficient. The freedom of the cameras to tilt and pan means a wedge of pixels is needed from each camera to ensure that there are no holes in the MPI. A normal panoramic image is a 2D slice of the plenoptic function in which the center of projection is fixed and all the directions are sampled. This type of MPI is a 2D slice of the plenoptic function in which the center of projection is allowed to move and only a small subset of the directions are sampled for any one center of projection. Reconstruction is restricted to panning a 2D window across the panorama, but the resulting visual effect can encompass trucking, dollying, and zooming in addition to panning.

In related work, Glassner [10] proposes using multiperspective panoramas to facilitate storytelling. Instead of viewing the panorama through a window that pans across the image, the panorama is viewed as a whole. Using this technique a scene can be viewed from multiple perspectives simultaneously.

A related type of image is what we might call a single-perspective multi-time image.

With the exception of the multiperspective panorama, the MPIs we've encountered so far have had a regular parameterization where one axis is camera position and the other axis is view direction. If we keep the camera fixed and let time vary, an interesting type of MPI can be created: a single-perspective multi-time image. Figure 12 shows an example of such an image being utilized in a photo finish camera. In this image, a vertical-slit camera is fixed in position and time is allowed to run forward. The same vertical slice of space is represented at each column of the image, but at a different time. This is why the background appears to be smeared across the image. If an object were to move across the camera's field of view at a constant rate, the image of the object would appear as it would in an orthographic image. If an animating object moves past the camera (as the runners in Figure 12 do), the parts of the object moving fast will appear very slender, while the parts moving more slowly will be fatter. This is evident in the sickle shape of the runners' legs. This type of image is a 3D slice of a 6D function. There are no reconstruction issues because, like Glassner's multiperspective panoramas, they are meant to be viewed as a whole.

Lastly, each pixel of a 2D image represents one sample of the radiance arriving at the center of projection of the image. It is sometimes useful to consider the radiance leaving a point instead of the radiance arriving at a point. If we turn an image inside out, and look at the radiance leaving a point, we are essentially describing a point light source (where an image would be a light sink). Since the term point light source is already used in computer graphics to mean something more specific, we'll use the notation of Wood et al. [39] and call these inverted images lumispheres. Images record incoming radiance, lumispheres record outgoing radiance.

3.2 Collections of Images

The second group of representations we consider use a set of 2D images (or photographs) to model an environment.

Movie Maps [19] was a system that allowed a virtual walkthrough of a small town by replaying predetermined video sequences that were stored on a videodisc. The video sequences followed the edges of a graph overlayed on the streets of Aspen, Colorado. At the nodes in the graph, the user could decide which subsequent path (edge in the graph) to follow, and the appropriate video sequence was played. Since Movie Maps used a set of 2D images, it was a 3D representation of the plenoptic function. However, it was a very restricted 3D representation: the view position could not deviate from that of the video camera used to record the images. Creating a set of movies over a predetermined set of paths is a form of sparsely sampling a degree of freedom. So, this representation lies somewhere between 3D and 4D; it is a 3.5D representation.

Shum and He [35] built a system called Concentric Mosaics that captures a dense set of images that are tangential to a circular path. A viewer is allowed to roam anywhere within the plane enclosed by the circlular path, but the gaze direction of the novel camera is restricted to lie in the plane. Novel views are reconstructed by rescaling selected vertical scanlines from the set of captured images. The reconstructed images exhibit significant distortions since the views reconstructed are typically far away from the path along which the source images are captured. The authors discuss an alternate representation that stores the captured images as a set of multi-perspective images in which the center of projection varies across vertical scanlines.

The image caching systems built by Shade et al. [34], Schaufler and Sturzlinger [31], and Aliaga [2], used dynamically created images to replace the geometry of a synthetic scene. These systems trade the lack of parallax in 2D images for speed of rendering. For small changes in the movement of an observer, the lack of parallax in a warped image is not very noticeable. It is therefore possible to dynamically cache images of portions of a scene and use the images in place of the geometry. As the observer moves, the cached images are rendered using a planar perspective warp to approximate the appearance of the portion of the scene that has been replaced. Both Shade et al. [34] and Schaufler and Sturzlinger [31] use a geometric error metric to control error. If a warped image exhibits too much error, a new image is created from the current observer's point of view. These systems cache a sparse set of images covering the entire scene, so they are 2.5D representations. Aliaga's system [2] took a slightly different approach. When the cached image was warped, the portions of the scene adjacent to the image that were still being rendered as geometry were warped slightly to match the distortion in the image. This minimized the abrupt boundary that would have been noticeable between the warped image cache and the geometry. This system also creates a sparse set of images and thus is a 2.5D representation.

Talisman [36] was a hardware graphics system designed to use image caching. Talisman differs from the previous image caching systems in that it used an affine warp instead of a planar perspective warp. Affine warps are less expensive to compute, but are more limited in scope. In a subsequent paper, Lengyel and Snyder [17] presented a quantitative analysis of various warping methods and showed convincing results that affine warps could be effectively used in place of planar perspective warps.

Image caching systems [34, 31, 2, 36, 17] exhibit two primary artifacts: discontinuities in shading and in occlusion relationships. Since these systems dynamically create images, when a new image cache is created there will often be large changes in the shading and occlusion relationships of the objects within the image. This sudden change is referred to as "popping". The error metrics presented by Lengyel and Snyder [17] can be used to control the severtity of the artifacts, but there is a fundamental tension between the amount of error that is allowed and the frequency with which new image caches must be created. Popping due to shading discontinuities can be completely eliminated if the scenes being sampled consist entirely of ideal diffuse surfaces. A diffuse surface has no specular highlights or glossy reflections, and thus looks the same regardless of the direction from which it is viewed. This assumption is an example of reducing the dimension of the plenoptic function by assuming a constancy of behavior. This effectively removes two dimensions from the plenoptic function. However, in practice, diffuse surfaces are rarely encountered. When the assumption of constancy of shading over viewing directions is imposed on a non-diffuse scene, information is being thrown away. As we will see later, the light field and lumigraph systems use contancy in a way that does not throw any information away.

3.3 Flow-based Representations

Everyone has had the experience of holding a photograph in their hands and tilting it back and forth in an attempt to breathe life into the image. This is not convincing because the viewer does not perceive any *parallax*. In the real world, when we move our heads back and forth, nearby objects move across our field of view very quickly while objects that are far away move slowly. Parallax is the difference in motion of two objects as seen by a moving observer. Tilting a photograph produces no parallax because we have projected an image of a scene onto a single plane: every point on the image lies at the same depth from the viewer. In computer graphics terms, as we tilt the photograph we are performing a planar perspective warp (the same type of warp used by the image caching systems of the previous section).

If we knew the actual depth for each individual pixel, we could accurately render an image from a new viewpoint using 3D image warping. A *flow field* can be computed for a pair of images (call them the source and destination images) by finding a corresponding point in the destination image for every pixel in the source image. The difference in image coordinates of the two points defines a vector in the flow field. For points that are far away, and thus likely to project to nearly the same pixel in both images, the vectors of the flow field will have a small magnitude. For points that are nearby, and thus likely to project to different areas of the image plane, the magnitude of the vectors of the flow field will be large. There is a direct correspondance between the magnitude of the flow field and the depth of a point in the image, so knowing the flow field is equivalent to knowing the depth of every point in the scene and vice versa. This observation has been used in computer vision to compute per-pixel depth values for acquired images by using point correspondances between images. [24]

Flow-based representations use *depth images*, also known as *range images*, as the fundamental modeling primitive. Until now, we have considered a pixel to be a sample of light. If we associate a depth value with every pixel, the pixel is more than just a sample of light,

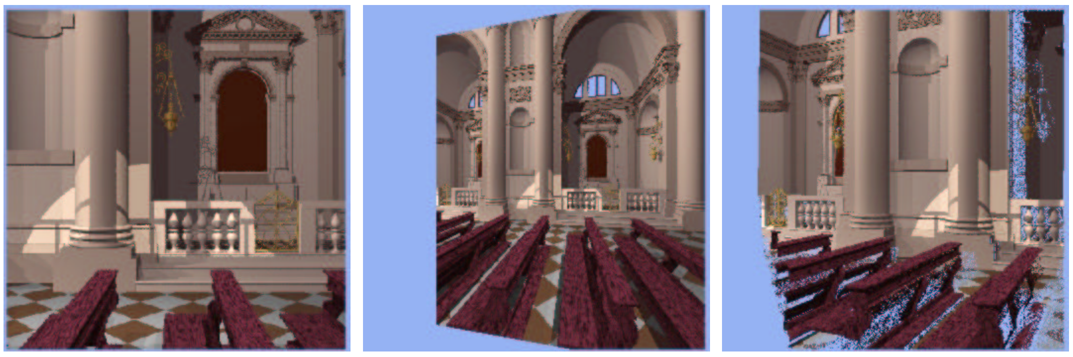

Figure 13: Planar perspective vs. 3D image warping.

it is the light that has been reflected or emitted from a known piece of geometry. Knowing the depth of every pixel in an image is equivalent to knowing the world space coordinates of the point of geometry represented by the pixel. Adding per-pixel depth transforms an image from a 2D representation into a 2.5D representation. The reconstruction step in the IBMR pipeline (see Figure 2) was very simple for 2D representations: either display the entire image, or display a regular subset of the image. Reconstruction for the 3D Movie Maps system was also simple: we projected the 3D representation onto a 2D view by taking an appropriate 2D slice (a frame of the movie). Reconstruction for a 2.5D representation is a process of reprojection: 2D data is "unprojected" into 3D by adding depth, and then projected back to the 2D view of an observer.

Representations based on depth images typically sample one dimension of space sparsely. The two dimensions covered by the width and height of the view volume of the image are sampled densely, but the dimension corresponding to depth is sampled sparsely. When these samples (now 3D points) are viewed from an oblique angle, holes will appear where data is missing. Figure 13 shows a comparison of 2D versus 3D image warping. Figure 13(a) shows the original 2D image. Figure 13(b) shows this image rendered from an oblique angle using a planar perspective warp. Figure 13(c) shows a rendering from the same oblique view using 3D image warping. The geometry of the scene is distorted when using 2D image warping, but the reconstruction has no holes. In contrast, the 3D image warping preserves the correct geometric relationships among the objects in the scene, but the reconstruction is incomplete. Since the 2D warping assumes the entire image lies on a plane, a continuous reconstruction can be made by interpolating between neighboring samples as the plane is drawn. The 3D reconstruction doesn't have this luxury because neighboring pixels in the depth image can have vastly different depths and thus project to vastly different areas in the reconstructed view. Finding ways to fill these gaps in reconstruction is often referred to as the "hole-filling problem". Holes appear for two reasons. The first, as already stated, result from missing data: surfaces that are occluded in the original view become visible in an oblique view. Holes also appear when the novel view translates towards the scene. This movement will cause the pixels in the source image to spread apart.

Three different methods of reprojecting depth images are used: polygon rendering, forward 3D image warping, and backward 3D image warping. In polygon rendering, each pixel in a depth image is treated as small piece of geometry and rendered using a hardware polygon renderer. If each pixel is rendered independently, hole filling is ignored. Some systems, however, convert the pixels in a depth image into a 3D mesh by connecting together neighboring pixels with similar depth values. Drawing triangles between pixels in this manner helps fill holes in the reconstruction process.

An alternative is to use 3D image warping, which can be implemented efficiently in software. If the relative positions of two cameras are known, a single 4*x*4 matrix (called the fundamental matrix [16]) establishes a correspondance between the pixels in the two images. A pixel in one image can be warped to the other image by mapping the homogeneous vector, $< x, y, z, 1 >$, composed of the pixel's screen coordinates and its depth through the fundamental matrix and then renormalizing the vector by the homogeneous coordinate. Since this operation is symmetric, we have two choices: given a source and a destination image we can either *forward warp* all of the pixels in the source image to the destination image, or we can *backward warp* every pixel in the destination image to find out which source image pixel to paint at that location.

In forward warping, every pixel in the source image is mapped to the output image. In order to fill holes, pixels are typically *splatted* [37] into the output image. A splat is an idealized point sample rendered as a semitransparent disc. The size of the disc is adjusted based on the position and resolution of the source and destination views. Splatting can only fill holes that result from the output camera translating into the scene; holes due to occluded geometry becoming visible will still be visible.

In backward warping, every pixel in the destination image is mapped to a location in the source image. The color for the pixel in the destination image is determined by interpolating between the pixels in the neighborhood around the location found in the source image. When all of the source image pixels are assumed to lie on a plane, this interpolation is straightforward. However, when each pixel can have an arbitrary depth, this interpolation becomes a search. Laveau and Faugeras [16] used epipolar geometry to show that this search can be restricted to a line. However, compared to splatting even a linear search is time-consuming. As a result, forward warping with splatting is the most common approach to 3D image warping. The three systems we review in this section use polygon rendering, a 2D approximation to forward warping, and full 3D forward warping, respectively.

Greene and Kass [13] extended the work of Movie Maps by relaxing the restriction of the viewpoint to a predetermined path. Greene and Kass used a regular grid of cubical panoramic images with per-pixel depth. Using the per-pixel depth, the panoramic image closest to the viewpoint was reprojected by rendering each pixel as a 3D quadrilateral. A z-buffer was used to resolve visibility. They solved the occlusion problem by only using the panoramic depth image to render objects that were far away. Nearby objects were rendered using their actual geometry.

Chen and Williams' view interpolation [6] also used a set of images with per-pixel depth, but used image warping instead of polygon rendering to reproject the depth images. Using per-pixel depth, flow fields were computed for every pair of adjacent input images. Because flow fields are many-to-one mappings, two fields must be computed for every pair of images, one for each direction. An intermediate view between two of the input images was created by linearly interpolating between the two flow fields. An interpolation parameter controlled how far along each flow field the images were warped. For instance if the new view were exactly halfway between each of two input images, the warping algorithm would follow halfway along each of the two flow fields. The warping was computed using a 2D approximation to full 3D image warping [3] that was efficient to compute in software. Hole filling was done as a post-process by interpolating across unfilled regions of the image using an algorithm akin to flood filling. Visibility was determined by presorting the pixels in pairs of images by depth. When rendering an intermediate view the pixels from the two input images were warped simultaneously in a back-to-front order.

The Plenoptic Modeling work of McMillan and Bishop [24] extended the work of view interpolation to use cylindrical images of acquired data and 3D image warping. McMillan and Bishop used a novel analysis of epipolar geometry to augment standard computer vision techniques to compute flow fields between pairs of cylindrical images of acquired data. They also presented an algorithm that reprojects the pixels of a cylidrical image in the order needed such that they are painted in a back-to-front order on the image plane of the novel view.

Like the image caching systems, these flow-based systems also use 2.5D representations. However, using per-pixel depth facilitates the use of more accurate reprojection algorithms. The quality of reconstruction provided by these systems depends on the number and resolution of depth images created. In the limit, if a panorama was created at every possible position, the 5D plenoptic function would be sampled comprehensively.

3.4 Hybrid Representations

The next group of papers we'll consider all use representations that extend the depth image to incorporate more geometric information. These representations fall into two broad categories: mesh-based representations and volumetric representations.

Mesh-based Representations. The first class of hybrid representations we consider create 3D geometric meshes from depth images. Darsa et al. [8] presented a system for interactively rendering complex static scenes by creating meshes from depth images. Given a cubical depth image of a scene, a mesh is created by triangulating the depth information. The mesh is then texture-mapped with the color image. For viewpoints close to the original one, the renderings made using this technique will exhibit the proper parallax. However, the shading captured is static, so this will only faithfully render scenes that are completely diffuse. This is a 2.5D representation that solves the hole-filling problem by rendering a mesh. This produces a "rubber sheet" artifact when an oblique view of the scene is taken.

Mark et al. [21], constructed a similar system, post-rendering 3D warping, that created meshes from depth images dynamically. This work was targeted towards the real-time display of a complex environment in a walk-through style application. As a viewer moved through a scene, depth images were created on a server at 5 frames per second and sent to a local workstation where they were rendered as meshes at 60 frames per second. Since the depth images were created dynamically, there was no restriction on the movement of the viewer. Like the image caching schemes, the transitions between an out-of-date depth mesh and the newly made one can produce discontinuities in shading. Like that of Darsa et al., this is a 2.5D representation.

The view-based rendering system of Pulli et al. [27] was geared towards displaying scanned real objects. Viewpoints were scattered around an object, and a collection of colored range images covering the object were acquired. The object was reconstructed by rendering the three meshes that were captured from directions closest to the virtual viewpoint. This technique captures both the shape and the shading of an object using a sparse set of acquired images. While the shape of an object can be faithfully reproduced using a sparse set of acquired images, shading can require a dense set of images. So, this technique exhibits popping artifacts in the shading as the viewpoint orbits about the object. Acquiring multiple images per range image would alleviate this problem at the expense of higher storage cost. Since this technique uses a small set of depth images, we consider it to be a

2.5D representation.

Debevec et al. [9] presented a view-dependent rendering algorithm, view-dependent texture mapping (VDTM), that used a single geometric mesh along with a collection of images. Given a geometric model and a set of registered images of that model, novel views of the scene were created by texture mapping the polygons of the model with images that are closest to the viewpoint. Unlike Pulli et al., it is assumed that every polygon in the scene is visible in more than one photograph. Blending together the image information from the three closest images produces smooth transitions in shading of the model as the viewpoint orbits the model. The degree to which this technique can faithfully reproduce specularity is determined by the number of images used. We consider this to be a 2.5D representation. Building a single 3D mesh for an object fixes the of the degrees of freedom in position to lie on a 2D surface, and is thus equivalent to taking a 2D slice of the plenoptic function. Sparsely sampling the space of directions about each point of the surfaces adds another half dimension to the representation. A dense sampling of the direction space would make this a 4D representation. As we'll see in the next section, the surface light field [39, 25] does just this.

Rademacher and Bishop [28] employ a style of MPI similar to the multiperspective panoramas of Wood et al. [40] to create what they call multiple-center-of-projection (MCOP) images. An MCOP image is an MPI with per-pixel depth in which each vertical strip of pixels is taken from the center vertical strip of a perspective range camera that is allowed to move freely along a space curve. It is assumed that the camera positions and viewing directions vary smoothly. The smoothness constraint ensures that the epipolar geometry varies smoothly across the columns of an MCOP image. As a consequence, incremental warping can be used to efficiently reproject MCOP images. Unfortunately, using multiple cameras complicates McMillan's occulsion compatible warp ordering, so a z-buffer was used to resolve visibility. Alternatively, a 3D mesh can be constructed using the per-pixel range image. The 3D mesh can then be rendered by texture-mapping it with the color information from the MCOP. Like Pulli et al., each triangle in the mesh is seen from only one

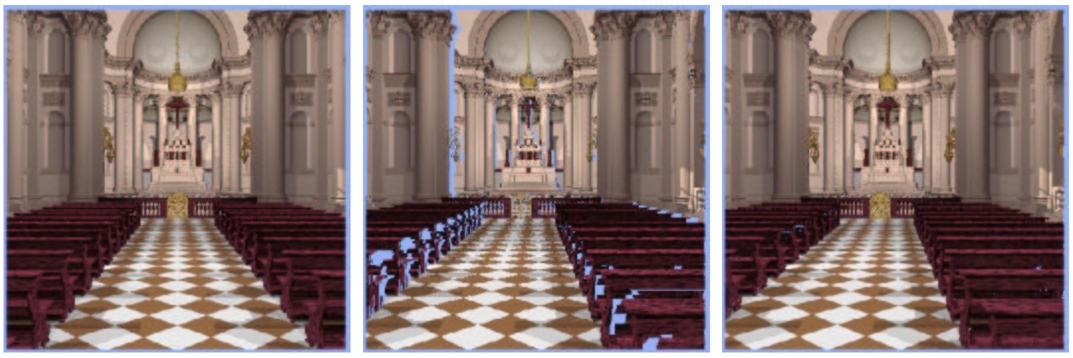

Figure 14: Layered depth images.

view, so this is a 2.5D representation.

Volumetric Representations. The second class of hybrid representations we consider build sparse volumes of point-sampled geometry. Layered Depth Images [33] extend depth images by adding multiple samples of the plenoptic function along each ray of an image. As the viewer moves away from the center of projection of the LDI, surfaces that had been hidden become visible. Figure 14 illustrates this. Figure 14(a) shows the LDI from its center of projection. Figure 14(b) shows the LDI from an oblique angle, but only one layer is rendered (mimicking the behavior of a depth image). Figure 14(c) shows the LDI from the same oblique angle, but using all of the layers. Using McMillan's occlusion compatible warp ordering for planar images [23], LDIs can be rendered efficiently in software without the use of a z-buffer. In practice, LDIs sparsely sample the dimension of a scene corresponding to depth in the image. A naive way of creating an LDI would be to treat each ray of an image as a skewer, and to record every intersection of the ray with the geometry of the scene. This would produce a dense sampling of the scene and thus a 3D representation. This method isn't used for several reasons, one of which is run-time performance; the data set would be too large to render in real-time. To combat this problem, Shade et al. [33] only stored intersections that were visible from a view nearby the center of projection of the LDI. This visibility-driven sampling results in a sparse data set that can be rendered in real-time. In addition, if an intersection was visible from more than one nearby view, the radiance values from all views were averaged to create a single view-independent sample. This averaging only produces the correct radiance value when the scene consists solely of purely diffuse surfaces. A similar data structure was used by Max [22] to create anti-aliased renderings of trees.

Similar in nature to Lippman's Movie Map [19], a Layered Depth Movie can be created from a series of LDIs placed along a predetermined path. The viewer is constrained to move along the path, but since an LDI is being displayed instead of an image, the viewer can move freely in a volume of space surrounding the path. Creating a dense set of LDIs along a path results in a 3.5D representation.

Lischinski and Rappoport [20] extended LDIs to handle non-diffuse scenes. They store two sets of LDIs for a scene: a Layered Depth Cube (LDC) consisting of three high resolution LDIs (corresponding to three orthogonal directions) that stores the view-independent component of the sampled geometry (diffuse color, depth, normal, and an index into a table of bidirectional reflection distribution functions, BRDFs); and a Layered Light Field (LLF) consisting of a dense set, 66 to 1026, of low resolution LDIs that store the view-dependent component of the sampled geometry (the total radiance leaving the scene in the direction of projection). The scene is rendered by first projecting the LDC to an intermediate image. After hole filling, each point in the image is shaded by integrating the incoming radiance from the LLF and applying this to the BRDF. This representation adds a dense sampling of the space of directions to an LDI, resulting in a 4.5D representation.

3.5 Holographic Representations

The roots of image-based modeling and rendering can be found in the field of holography. A hologram is a photographic record of the interaction of light with an object which has the property that when it is illuminated properly, it displays the object of record with full parallax. Holograms have become commonplace objects. Most credit or debit cards have a rainbow hologram (also known as a Benton hologram) in the corner. The rainbow hologram is a horizontal parallax only (HPO) hologram: a change in viewpoint in the vertical direction changes the hue of the object, while a change in the horizontal direction changes the perspective view of the object. More precisely, a hologram is the record of the interference pattern created when two beams of correlated light interact, one of which has been perturbed by placing the object to be recorded in the path of the beam of light. When this interference pattern is illuminated by the same light source, the pattern of light that hit the photographic plate is reconstructed, resulting in an image of the object. Holograms are very effective, but they have a number of limitations: The reconstructed image is necessarily the same size as the subject that was recorded. The recording apparatus must be stable to within one tenth of a wavelength of light during exposure (meaning that holograms are more or less confined to a table top setting). Natural color is very hard to reproduce. Finally, the images created are snapshots in time: the scene being imaged must be static.

Holographic stereograms are a hybrid of holography and photography. The primary advantage holographic sterograms have over holograms is that they can be used to make images of anything that can be photographed. A holographic stereogram is created by exposing a piece of holographic film to a series of perspective views. When the hologram is illuminated with the reference light, a different perspective image will be view by each eye of the observer resulting in a stereo image of the scene.

3.6 Digital Holographic Representations

The light field [18] and lumigraph [12] papers presented 4D representations that are essentially digital holographic stereograms. These two papers use the same device for reducing the plenoptic function from five dimensions to four: applying the ray law to the set of rays entering a volume of free space. As light travels through space, it remains unchanged unless it encounters some obstacle: geometry or a participating media (like a gas or a cloud). If we assume that a viewer is going to stay within a volume of free space, then we can assume that every ray of light that enters this space remains unchanged until it exits. A simple example of such a volume is a sphere. We can parameterize all of the rays that intersect a sphere using 4 coordinates: the lattitude and longitude of the ray as it enters and exits the sphere. This reduces the number of dimensions of the plenoptic function from five to four. Normally, it takes five numbers to describe a ray: three for position and two for direction. Consider a line intersecting a sphere. The free space assumption implies that every 5D ray originating on that line, looking along the line is equivalent: they all see the same value of radiance. Because the behavior of light is constant in the region of free space, this collection of 5D rays can be named by one 4D ray.

In practice, these representations use two planes to parameterize the space of rays. To represent the entire environment surrounding a viewer, six sets of planes can be used to describe all of the rays entering a box. The rays for a set of planes are typically stored as a 2D array of 2D images. The near plane acts as a locus of camera centers, while the far plane acts as the focal plane for the cameras. Each image in the representation is a skewed perspective pinhole projection. To avoid aliasing, camera centers on the near plane need to be spaced no further apart than the aperture of the reconstruction camera [14]. This is a very dense representation that can produce very high quality images, but requires an enormous amount of storage.

A surface light field [25, 39] is an alternative parameterization of a light field. Like view-dependent texture mapping, the surface light field representation assumes that a 3D mesh of an object exists. In contrast to VDTM, however, surface light fields record the radiance leaving the surface using a dense set of cameras. This results in a 4D representation: for every point on a surface there is an associated lumisphere, a 2D image of the radiance leaving the surface at that point. By taking advantage of the fact that lumispheres vary smoothly over the surface of an object, surface light fields can be compressed with higher fidelity than the standard two-plane parameterization.

Light field representations can reproduce the plenoptic function exactly in regions of free space. However, since light fields are a 4D slice of the plenoptic function, they typically require a great deal of storage space. If the object being represented can be represented by

Representation	# Dimensions	Technique		
		Slice	Sparse	Constancy
Image	2	√		
Katayama et al.	3	√		
MPPCA	2	√		
Photo Finish	2	√		
Concentric Mosaics	3	√		
Movie Maps	3.5	√		
Image Caching	2.5	√	√	
Greene and Kass	2.5	√	√	
View Interpolation	2.5	√	√	
Plenoptic Modeling	2.5	√	√	
Darsa et al.	2.5	√	√	√
Post-rendering 3D warping	2.5	√	√	
MCOP	2.5	√		
View-based rendering	2.5	√	√	
VDTM	2.5	√	√	
Layered Depth Image	2.5	√	√	√
Layered Depth Movie	3.5	√	√	√
Lischinski and Rappoport	4.5			√
Hologram	4			√
Holographic stereogram	4			√
Light field	4			√
Lumigraph	4			√
Surface light field	4			√

Table 1: A summary of the papers: the dimension of the representation and the techniques used.

a 3D mesh, surface light fields offer the advantages of a 4D representation in concert with a compressed representation.

3.7 Summary

Table 1 shows a summary of the papers we have reviewed. The representations are listed in the order they have be presented. Each line of the table lists the dimension of the representation and which of the data reduction techniques were employed.

4 Taxonomy of Representations

This section presents a walk through the representations made possible by applying the four tools for reducing the dimension of the plenoptic function: taking a subset of the plenoptic function, taking a slice of the plenoptic function, sparsely sampling a parameter, and eliminating redundancy. Tables 2 and 3 give a concise description of each of the representations. There are six distinctions to each representation: how many of the dimensions are fixed, how many dimensions are free, which parameters are fixed, which parameters are free, whether radiance is considered entering or leaving, and what the representation is called. We have grouped the representations into categories based on the number of dimensions that are fixed. Starting with the singular 0 dimensional object, which requires 5 dimensions to name it, we work our way up to four dimensional objects, which require one dimension to name them. Within each category are a number of variations on which parameters are fixed and which are left free. It is assumed that we are approximating the 5D plenoptic function: time and wavelength are fixed. However, it will sometimes be interesting to reincorporate time and consider approximations to a 6D plenoptic function. The three degrees of freedom in position are *x*, *y*, and *z*, and the two degrees of freedom in direction are θ and ϕ. In each entry, we'll fix some of the degrees of freedom and let other vary. For instance, to describe an image, we would fix *x*, *y*, and *z* and leave θ and ϕ free. To describe an orthographic image, we would fix θ, ϕ, and *z*, and leave *x* and *y* free. Fixing *z* means we are choosing some plane in space. The fact that *z* was the position variable chosen to be fixed is unimportant: we could have chosen any of the three. To describe basic geometric constructs like points, lines, and planes, we'll fix one or more of *x*, *y*, and *z*. To describe more general geometry, we'll say that the free variables vary over the geometric construct. For instance, to choose a surface, we'll say: *z* is fixed while *x* and *y* vary over a surface.

4.1 0D Representations

Choose:	x, y, z, θ, ϕ	**Vary:**	*None*	**Radiance:**	*both*

Point (0d slice): The basic element of the plenoptic function is the point sample. A point sample in the plenoptic function is defined by a point in space and a direction. This is specified using 5 parameters: 3 for a point $< x, y, z >$, and 2 for a direction $< \theta, \phi >$. A point in this function describes a single radiance value. If the radiance is incoming, this is a pixel in an image. If the radiance is outgoing, this is a pixel in a radiance map.

4.2 1D Representations

Choose:	x, y, z, θ	**Vary:**	ϕ	**Radiance:**	*arriving*

1D slice: This representation is simply a 1D slice of an image. If we were to pick a point and render an image of the scene (or take a photograph of the world), and then chose a line or curve through it, we would get a 1D slice. The number of free parameters defines the dimension of a slice. Consider a spherical image, if we extract a single line of latitude or longitude from the image we get a 1D slice. If we take a vertical line from a planar image we also get a 1D slice. This representation forms the basis of work done using vertical slit cameras in holography, as illustrated in Figure 7.

Choose:	x, y, z, θ	**Vary:**	ϕ	**Radiance:**	*leaving*

If we consider the radiance leaving a point instead of the radiance arriving, we get a 1D slice of a lumisphere.

Choose:	x, y, θ, ϕ	**Vary:**	z	**Radiance:**	*arriving*

This representation is equivalent to choosing a single pixel in a set of images that are taken along a line. If planar images are used and θ and ϕ are chosen to look in a direction perpendicular to the line of motion, this amounts to a vertical slice of an Epipolar Plane Image. If we chose θ and ϕ to point along the direction of the line of motion, we would find all of the radiance values along a single ray as it travels through a scene.

Choose:	x, y, θ, ϕ	**Vary:**	z	**Radiance:**	*leaving*

This is the same representation, but with radiance leaving. One interpretation of this structure is that it is a slice of an area light source. If we choose θ and ϕ to point along the direction of the line of motion, we get the same result as in the previous entry, but with the ray pointed in the opposite direction.

4.3 2D Representations

Choose:	*x, y, z*	**Vary:**	θ, ϕ	**Radiance:**	*arriving*

Image(2D slice): This representation is the standard 2D image. Depending on how we choose θ and ϕ, we get one of the many types of images: planar, cubical, spherical, or cylindrical.

Choose:	*x, y, z*	**Vary:**	θ, ϕ	**Radiance:**	*leaving*

Lumisphere: An image that records outgoing radiance.

Choose:	*x, y*, θ	**Vary:**	*z*, ϕ	**Radiance:**	*arriving*

Multiperspective image: In this case we vary one parameter of position (a line or curve) and one parameter of direction (a slice through an image). This encompasses the entire class of multiperspective images. If we choose a horizontal slice through each image we get the epipolar plane images used by Katayama et al. [15]. On the other hand, if we choose a vertical slice through each image we get the multiperspective panoramas of Wood et al. [40] and Glassner [10], and the MCOP images of Rademacher and Bishop [28].

Choose:	*x, y*, θ	**Vary:**	*z*, ϕ	**Radiance:**	*leaving*

Lumisphere slices: As we mentioned above, the position parameter can be varied over an arbitrary line or curve. If we choose a curve that lies on the surface of an object and choose the direction parameter to draw out a great circle on the sphere of directions, we'll get a series of slices through the lumispheres of a surface light field. Wood et al. show this type of image in Figure 5 of [39], reproduced here in Figure 15.

Choose:	*x, y, z*, θ	**Vary:**	ϕ, *time*	**Radiance:**	*arriving*

Single perspective, multitime image: This type of image uses the same center of projec-

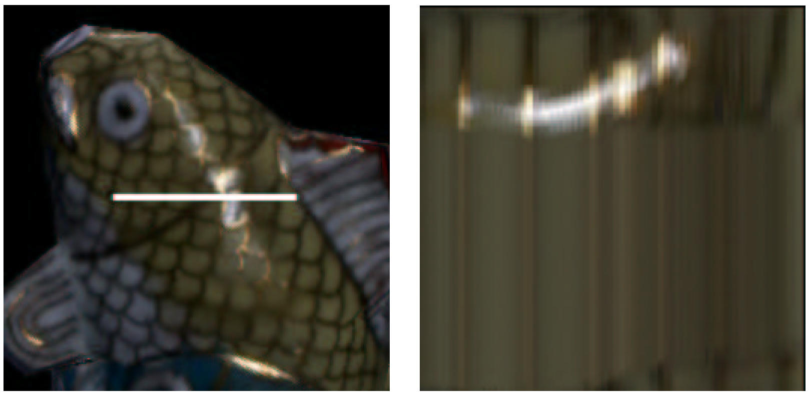

Figure 15: (a) Surface light field. (b) Transect of surface light field. Horizontal axis shows position of along white line across fish. Vertical axis shows a slice of a lumisphere on a user-selected great circle.

tion for each vertical slit, but a different point in time. Figure 12 shows an example of this type of image taken with a "photo finish" camera.

Choose:	*z*, θ, ϕ	**Vary:**	*x*, *y*	**Radiance:**	*arriving*

Orthographic planar image: If the camera is chosen such that it is looking perpendicular to some plane, and the sample taken from the camera is the one along its line of sight. The result is an orthographic planar image. Otherwise, it will be a skewed orthographic image.

Choose:	*z*, θ, ϕ	**Vary:**	*x*, *y*	**Radiance:**	*leaving*

Area directional light source: If we choose an arbitrary plane in space and a direction and record the radiance leaving the plane, we will be constructing an area light source that emits light in only one direction.

4.4 2.5D Representations

Choose:	*x*, *y*, *z*	**Vary:**	θ, ϕ, *independent ray parameters*	**Radiance:**	*arriving*

Layered depth image: In this representation, we choose a single point in space but look at the light coming in from all directions. This is the definition we have used for an image. If we record multiple values of radiance for each incoming ray, we get a layered depth image. The ray parameter is allowed to vary independently along each ray. If we sample the ray parameter sparsely, we'll get the 2.5D layered depth image described in Section 3. If we perform a dense sampling the ray parameter, we'll get a 3D representation that is essentially a volumetric representation of the geometry in a scene.

Choose:	*x,y,z*	**Vary:**	*θ, ϕ, constrained ray parameters*	**Radiance:**	*arriving*

Multi-layered image: A layered depth image allows the ray parameter to vary along each ray independetly. If we force the all of the rays of an image to be sampled at the same set of parameter values, we'll get a set of concentric images (or if using a planar image, a set of planes). This type of representation has been used in cel animation, side-scrolling video games, and in the immersive virtual reality system built by Regan and Pose [29].

Choose:	*set of x,y,z*	**Vary:**	*θ, ϕ*	**Radiance:**	*arriving*

Set of 2D images: This representation encompasses all systems that use a collection of 2D images, including: image caching, view interpolation, and QuickTime VR [5].

Choose:	*set of θ, ϕ*	**Vary:**	*x,y,z over a surface*	**Radiance:**	*arriving*

View-dependent texture mapping: This is the representation used in view-base rendering, view-dependent texture mapping: a sparse set of photographs of a surface.

4.5 3D Representations

Choose:	*x, y*	**Vary:**	*z, θ, ϕ*	**Radiance:**	*arriving*

Space movie: In this representation, images are taken as a camera moves along a line. Since time is frozen, the environment is static. Therefore, this is a movie in space only, the camera is moving but the world is not. If we were to freeze the camera position and let time move forward, we would still have a 3D representation, but now it would be a time movie. If we let both time and camera position vary, we would get a representation of one higher dimension, 4, a space-time movie.

If the gaze direction is fixed to be perpendicular to the line of motion, we get the set of input images Bolles et al. [4] and Katayama et al. [15] used to make epipolar plane images. This would also correspond to a row of images from a light field or lumigraph. If the line of motion is a curve in space, and the gaze is fixed on an object, we'll get image suitable for use in a horizontal-parallax-only (HPO) holographic stereogram. If the gaze is tangential to a circular path, we get input suitable to construct a Concentric Mosaic [35].

Choose:	*x, y*	**Vary:**	*z, θ, ϕ*	**Radiance:**	*leaving*

Set of lumispheres: If we constrain the position of the line to lie along the surface and record the outgoing radiance, we'll capture a set of lumispheres. When viewed as a movie, this would show the change in reflectance of an object as a point moved along its surface.

Choose:	*x, y, z*	**Vary:**	*θ, ϕ, ray parameter*	**Radiance:**	*arriving*

Dense Layered Depth Image: The standard LDI stores only some of the intersections as each ray of an image travels through a scene. If we stored every intersection, we would build a complete 3D volumetric representation of the geometry of a scene.

Choose:	*x, y, z*	**Vary:**	*θ, ϕ, time*	**Radiance:**	*arriving*

Time movie: Let's add time back into the plenoptic function and consider a 3D structure that is a subset of 6D space, not 5D. Adding time to an image results in a movie in which the camera is stationary, but the world is moving: a time movie.

Choose:	*x, y*	**Vary:**	*θ, ϕ, time*	**Radiance:**	*arriving*

Space-time movie: If we take this notion one step further and add movement to the camera we get a space-time movie: a 4D subset of a 6D function. The familiar motion picture (or video) is a 3D slice of this 4D space because the camera position is not "omnipresent", it is a function of time.

4.6 3.5D Representations

Choose:	*x, y*	**Vary:**	*z, θ, ϕ*	**Radiance:**	*arriving*

Movie Map: In this representation, we restrict the free position parameter to lie on one of a set of predetermined paths. Connecting the paths together in a graph gives us a movie map.

Choose:	*x, y*	**Vary:**	*z, θ, ϕ, discrete ray parameter*	**Radiance:**	*arriving*

Layered Depth Movie: Just as taking a series of images along a line results in a movie, taking a series of layered depth images along a line results in a layered depth movie.

4.7 4D Representations

Choose:	*z*	**Vary:**	*x, y, θ, ϕ*	**Radiance:**	*arriving*

Light field: The lightfield and lumigraph representations reduce the dimension of the plenoptic function by making a simple assumption: the space in which the viewer moves is empty. If the viewer is constrained to a box that is free of geometry and participating media, the ray law may be used to reduce the plenoptic function from five dimensions to four. Since there is nothing in the box for light to interact with, any ray that enters the box will leave the box unchanged as per the ray law. Thus all of the light a viewer could possibly see from inside the box is completely described by the light hitting the sides of the box. The rays entering one side of the box can be described using the two-plane parameterization: 2 parameters for position on the near plane (locus of camera centers), and 2 for position on the far plane (direction). Following the terminology of Levoy and Hanrahan, a pair of planes is a slab, and the light enterring (or leaving) the box can be described by a set of six slabs.

Choose:	*z*	**Vary:**	*x, y (over surface), θ, ϕ*	**Radiance:**	*leaving*

Surface light field: A lightfield need not be restricted to a plane parameterization. Any two dimensional set of positions combined with the sphere of directions defines a lightfield. Restricting *x* and *y* to a surface results in the surface light fields of Miller et al.and Wood et al..

Choose:	*z*	**Vary:**	*x, y (over sphere), θ, ϕ*	**Radiance:**	*arriving*

Object movie: This is a special case of a lightfield where the surface chosen is a sphere, and the viewing direction of the camera is fixed at the center of an object. This representation is found in the Quicktime VR system [5]. In the QuickTime VR implementation, reconstruction is limited to positions on the sphere.

Choose:	*z, ray parameter*	**Vary:**	x, y, θ, ϕ	**Radiance:**	*arriving*

Planar set of panoramas: In this representation, we choose a plane over which we will capture a set of panoramic images and a fixed depth from the center of projection of the cameras at which we'll capture the panoramas. Conceptually, this is a 3D version of cel animation or an extension of the work of Regan and Pose [29] to represent viewpoints on a plane.

4.8 4.5D Representations

Choose:	*none*	**Vary:**	x, y, z, θ, ϕ	**Radiance:**	*arriving*

Non-diffuse layered depth image: A layered depth image with view-dependent shading information. Since an LDI sparsely samples one direction of space and assumes constancy in shading, it is a 2.5D representation. The work of Lischinski and Rappoport [20] adds view-dependent shading to LDIs and thus increases the dimension of the representation by two dimensions. This is very close to being a 5D representation of the plenoptic function.

5 Summary

Tables 2 and 3 show a summary of the image-based representations we have classified.

# Dims fixed	**# Dims free**	**Fixed/Free Parameters**	**Radiance**	**What it is**
5	O (Point)	choose: x, y, z, θ, ϕ vary: none	N/A	A single radiance value.
4	1 (Line)	choose: x, y, z, θ vary: ϕ	arriving	1D slice of an image
			leaving	1D slice of a lumispere
		choose: x, y, θ, ϕ vary: z	arriving	A line in an EPI.
		choose: x, y, θ, ϕ vary: z	arriving	A line in an area light source.
3	2 (Image)	choose: x, y, z vary: θ, ϕ	arriving	image
		choose: x, y, z vary: θ, ϕ	leaving	lumisphere
		choose: x, y, θ vary: z, ϕ	arriving	multiperspective image
		choose: x, y, θ vary: z, ϕ	arriving	lumisphere slices
		[2D slice of 6D] choose: x, y, z, θ vary: ϕ, time	arriving	single-perspective, multi-time image
		choose: z, θ, ϕ vary: x ,y	arriving	orthographic planar image
		choose: z, θ, ϕ vary: x, y	leaving	area directional light source
3.5	2.5	choose: x, y, z vary: θ, ϕ, independent ray parameters	arriving	layered depth image
		choose: x, y, z vary: θ, ϕ, constrained ray parameters	arriving	multi-layered image
		choose: set of x, y, z vary: θ, ϕ	arriving	Quicktime VR
		choose: set of θ, ϕ vary: x, y, z over a surface	arriving	view-dependent texture mapping

Table 2: Plenoptic Representations

# Dims fixed	**# Dims free**	**Fixed/Free Parameters**	**Radiance**	**What it is**
2	3 (Movie)	choose: x, y vary: z, θ, ϕ	arriving	space movie, epipolar plane images, HPO holographic stereogram, HPO light field
		choose: x, y vary: z, θ, ϕ	leaving	set of lumispheres
		choose: x, y, z vary: θ, ϕ, ray parameter	arriving	dense layered depth image
		[3D slice of 6D] choose: x, y, z vary: θ, ϕ, time	arriving	time movie
		[4D slice of 6D] choose: x, y vary: z, θ, ϕ, time	arriving	space-time movie
2.5	3.5	choose: x, y vary: z, θ, ϕ		movie map
		choose: x, y vary: z, θ, ϕ, ray parameter		layered depth movie
1	4	choose: z vary: x, y, θ, ϕ	arriving	light field
		choose: z vary: x, y (over surface), θ, ϕ	leaving	surface light field
		choose: z vary: x, y (over sphere), θ, ϕ	arriving	object movie
		choose: z, ray parameter vary: x, y (over plane), θ,ϕ	arriving	set of panoramas
0.5	4.5	choose: vary: x, y, z, θ, ϕ	arriving	non-diffuse LDI

Table 3: Plenoptic Representations Continued

6 Future Work

In reviewing the summary of published work presented in Table 1 one aspect stands out: there are very few 3D representations. Most of the representations either concentrate on geometric (depth image) constructions with shading sampled very sparsely, or they sample a 4D plenoptic function. Surface light fields and non-diffuse LDIs combine geometric data with a dense sampling of shading, but both construct at least a 4D representation. Even though a representation like surface light fields achieves a high degree of compression, when the environment sampled is scaled up from a handheld object to a room, building, or city street, a 4D representation may be too large. We can still produce a compelling experience by eliminating one dimension of direction and create horizontal-parallax-only. The observation was made long ago by holographers that, for a human observer, the most important parallax information lies in the horizontal portion of our field of view. The focus of the future work we suggest is therefore on adapting the some of the systems we have seen to produce 3D HPO representations. We'll use two design princples that have been previously used successfully: separating diffuse and specular shading information, and parameterizing shading information with respect to a surface. Lastly, we'll step outside of the 5D plenoptic function and suggest a representation that incorporates time.

HPO Light Fields. Light fields offer a compelling experience, but require very large data sets. The size of the environment over which a light field can be constructed could be expanded by sacrificing one direction of parallax. A problem with this tradeoff is that the volume of views from which an HPO light field would be valid is very small. To combat this, we suggest building hierarchical HPO light fields. A high resolution light field would be used to render objects that are nearby while a lower resolution light field would be used to render the surrounding environment. As a user move through the environment, HPO lightfields would be streamed off a disk. The greatest challenge in bulding this representation is separating objects into foreground and background lightfields. Capturing such a data set in a forest would be especially challenging.

HPO Surface Light Fields. This is essentially the same idea, but applied to the surface parameterization of light fields. It is important to separate the two because both representations are important: light fields can capture objects that have complex non-manifold geometry while surface light fields offer a compact, optimized representation for objects that have well defined surfaces. There are two ways of thinking about an HPO surface light field: it is either a 4D surface light field with one degree of freedom in shading removed, or it is view-dependent texture mapping with one dimension of dense shading added. This is the ideal representation for urban simulation. Since people usually experience urban environments from the ground, a faithfull reproduction could be made by restricting sampling of the shading of buildings to street level views. In addition to reducing the size of the data, this restriction makes acquisition easier.

Separating diffuse and specular shading in light fields. Separating diffuse and specular shading would aid in compressing the size of light fields. Wood et al. [39] and Lischinski and Rappoport [20] have shown success in doing this for 4D and 4.5D representations. In the urban setting, a compact representation would be possible if we could extract a single diffuse texture map for each building from a set of images. The view-dependent information would then be just specular highlights and reflections, which (ignoring glass building) appear only on a small portion of a building's surface (ie. windows and metal adornments). Creating a standard lightfield using separate data structures for diffuse and specular shading has not been done before.

HPO Layered Depth Images. Layered depth images offer an efficient way to render complex geometry. Adding HPO shading information to an LDI provides a mechanism for extending the notion of surface light fields to objects that have non-manifold geometry, like trees, bushes and flowers. In order to do this for real objects, we would need a way to reliably extract per-pixel depth using passive scanning technology. Active scanners (ie. lasers) don't work well on surfaces that aren't smooth. In addition, they are typically limited to scanning small objects; scanning a willow tree would be very difficult. A promising approach would combine the voxel coloring algorithms of Seitz et al. [32] to recover geometry, the function approximation techniques of Wood et al. [39] to recover shading, and the rendering techniques of Shade et al. [33] to display the data set in real time.

HPO Time Light Fields. Lastly, IBMR research has, to this point, ignored time. A simple an elegant demonstration of the effects of time could be shown by capturing an HPO lightfield of a static object as time changes the lighting conditions. As an example, one might want to view an outdoor statue as the time of day changes, or as the seasons change. This would produce a 4D data set, but one that is more compelling than a standard light field because, in some sense, the object would not be static. If the light conditions are carefully controlled, the reflectance properties of the object could be inferred, enabling "relighting" the object: rendering it under lighting conditions other than those under which it was captured.

References

[1] E. H. Adelson and J. R. Bergen. The Plenoptic Function and the Elements of Early Vision. In M. Landy and J. A. Movshon, editors, *Computational Models of Visual Processing*, chapter 1. MIT Press, Cambridge, MA, 1991.

[2] Daniel G. Aliaga. Visualization of Complex Models Using Dynamic Texture-based Simplification. *IEEE Visualization '96*, pages 101–106, October 1996. ISBN 0-89791-864-9.

[3] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):35–42, July 1992. ISBN 0-201-51585-7. Held in Chicago, Illinois.

[4] Robert C. Bolles, Harlyn H. Baker, and David H. Marimont. Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion. *International Journal of Computer Vision*, 1(7):7–55, 1987.

[5] Shenchang Eric Chen. Quicktime VR - An Image-Based Approach to Virtual Environment Navigation. *Proceedings of SIGGRAPH 95*, pages 29–38, August 1995. ISBN 0-201-84776-0. Held in Los Angeles, California.

[6] Shenchang Eric Chen and Lance Williams. View Interpolation for Image Synthesis. *Proceedings of SIGGRAPH 93*, pages 279–288, August 1993. ISBN 0-201-58889-7. Held in Anaheim, California.

[7] Michael F. Cohen and Donald P. Greenberg. The Hemi-Cube: A Radiosity Solution for Complex Environments. *Computer Graphics (Proceedings of SIGGRAPH 85)*, 19(3):31–40, August 1985. Held in San Francisco, California.

[8] Lucia Darsa, Bruno Costa Silva, and Amitabh Varshney. Navigating Static Environments Using Image-Space Simplification and Morphing. In *Proc. 1997 Symposium on Interactive 3D Graphics*, pages 25–34. 1997.

[9] Paul E. Debevec, Yizhou Yu, and George D. Borshukov. Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping. *Eurographics Rendering Workshop 1998*, pages 105–116, June 1998. ISBN 3-211-83213-0. Held in Vienna, Austria.

[10] Andrew Glassner. Cubism and Cameras: Free-form Optics for Computer Graphics. Technical Report MSR-TR-2000-5, Microsoft Corporation, 2000.

[11] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modelling the Interaction of Light Between Diffuse Surfaces. *Computer Graphics (Proceedings of SIGGRAPH 84)*, 18(3):213–222, July 1984. Held in Minneapolis, Minnesota.

[12] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The Lumigraph. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 43–54. ACM SIGGRAPH, Addison Wesley, August 1996.

[13] Ned Greene and Michael Kass. Approximating Visibility with Environment Maps. Technical Report 41, Apple Computer, Inc., 1994.

[14] Michael W. Halle. Holographic Stereograms as Discrete Imaging Systems. In *Practical Holography VIII*, volume 2176, pages 73–84. SPIE - The International Society for Optical Engineering, Bellingham, WA, 1994.

[15] Akihiro Katayama, Koichiro Tanaka, Takahiro Oshino, and Hideyuki Tamura. A Viewpoint Dependent Stereoscopic Display Using Interpolation of Multi-Viewpoint Images. *Stereoscopic Displays and Virtual Reality Systems II(SPIE)*, 2409:11–20, 1995.

[16] S. Laveau and O. D. Faugeras. 3-D Scene Representation as a Collection of Images. In *Twelfth International Conference on Pattern Recognition (ICPR'94)*, volume A, pages 689–691. IEEE Computer Society Press, Jerusalem, Israel, October 1994.

[17] Jed Lengyel and John Snyder. Rendering with Coherent Layers. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 233–242. ACM SIGGRAPH, Addison Wesley, August 1997.

[18] Marc Levoy and Pat Hanrahan. Light Field Rendering. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 31–42. ACM SIGGRAPH, Addison Wesley, August 1996.

[19] A. Lippman. Movie-Maps: an Application of the Optical Videodisc to Computer Graphics. *Computer Graphics*, 14(3):32–42, July 1980.

[20] Dani Lischinski and Ari Rappoport. Image-Based Rendering for Non-Diffuse Synthetic Scenes. *Eurographics Rendering Workshop 1998*, pages 301–314, June 1998. ISBN 3-211-83213-0. Held in Vienna, Austria.

[21] William R. Mark, Leonard McMillan, and Gary Bishop. Post-Rendering 3D Warping. *1997 Symposium on Interactive 3D Graphics*, pages 7–16, April 1997. ISBN 0-89791-884-3.

[22] Nelson Max. Hierarchical Rendering of Trees from Precomputed Multi-Layer Z-Buffers. In Xavier Pueyo and Peter Schröder, editors, *Eurographics Rendering Workshop 1996*, pages 165–174. Eurographics, Springer Wein, New York City, NY, June 1996.

[23] Leonard McMillan. Computing Visibility Without Depth. Technical Report 95-047, University of North Carolina, 1995.

[24] Leonard McMillan and Gary Bishop. Plenoptic Modeling: An Image-Based Rendering System. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 39–46. ACM SIGGRAPH, Addison Wesley, August 1995.

[25] Gavin S. P. Miller, Steven Rubin, and Dulce Ponceleon. Lazy Decompression of Surface Light Fields for Precomputed Global Illumination. *Eurographics Rendering Workshop 1998*, pages 281–292, June 1998. ISBN 3-211-83213-0. Held in Vienna, Austria.

[26] Tomoyuki Nishita and Eihachiro Nakamae. Continuous Tone Representation of Three-Dimensional Objects Taking Account of Shadows and Interreflection. *Computer Graphics (Proceedings of SIGGRAPH 85)*, 19(3):23–30, July 1985. Held in San Francisco, California.

[27] Kari Pulli, Michael Cohen, Tom Duchamp, Hugues Hoppe, Linda Shapiro, and Werner Stuetzle. View-based Rendering: Visualizing Real Objects from Scanned Range and Color Data. *Eurographics Rendering Workshop 1997*, pages 23–34, June 1997. ISBN 3-211-83001-4. Held in St. Etienne, France.

[28] Paul Rademacher and Gary Bishop. Multiple-Center-of-Projection Images. *Proceedings of SIGGRAPH 98*, pages 199–206, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

[29] Matthew Regan and Ronald Pose. Priority Rendering with a Virtual Reality Address Recalculation Pipeline. *Proceedings of SIGGRAPH 94*, pages 155–162, July 1994. ISBN 0-89791-667-0. Held in Orlando, Florida.

[30] Graham Saxby. *Practical Holography*. Prentice Hall, second edition, 1994.

[31] Gernot Schaufler and Wolfgang Stürzlinger. A Three Dimensional Image Cache for Virtual Reality. *Computer Graphics Forum*, 15(3):227–236, August 1996. ISSN 1067-7055.

[32] Steven M. seitz and Charles R. Dyer. Photorealistic Scene Reconstruction by Voxel Coloring. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 1067–1073. 1997.

[33] Jonathan Shade, Steven J. Gortler, Li wei He, and Richard Szeliski. Layered Depth Images. *Proceedings of SIGGRAPH 98*, pages 231–242, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

[34] Jonathan Shade, Dani Lischinski, David Salesin, Tony DeRose, and John Snyder. Hierarchical Image Caching for Accelerated Walkthroughs of Complex Environments. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 75–82. ACM SIGGRAPH, Addison Wesley, August 1996.

[35] Heung-Yeung Shum and Li-Wei He. Rendering with Concentric Mosaics. *Proceedings of SIGGRAPH 99*, pages 299–306, August 1999.

[36] Jay Torborg and Jim Kajiya. Talisman: Commodity Real-time 3D Graphics for the PC. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 353–364. ACM SIGGRAPH, Addison Wesley, August 1996.

[37] Lee Westover. Footprint Evaluation for Volume Rendering. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 367–376. August 1990.

[38] Turner Whitted. An Improved Illumination Model for Shaded Display. *Communications of the ACM*, 23(6):343–349, June 1980.

[39] Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. Surface Light Fields for 3D Photography. In Kurt Akeley, editor, *to appear in SIGGRAPH 2000 Conference Proceedings*, Annual Conference Series. ACM SIGGRAPH, Addison Wesley, August 2000.

[40] Daniel N. Wood, Adam Finkelstein, John F. Hughes, Craig E. Thayer, and David H. Salesin. Multiperspective Panoramas for Cel Animation. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 243–250. ACM SIGGRAPH, Addison Wesley, August 1997.