Dynamically Reparameterized Light Fields

Dynamically Reparameterized Light Fields

Aaron Isaksen[†*] Leonard McMillan[†] Steven J. Gortler[‡]

[†]Laboratory for Computer Science
Massachusetts Institute of Technology

[‡]Division of Engineering and Applied Sciences
Harvard University

Abstract

This research further develops the light field and lumigraph image-based rendering methods and extends their utility. We present alternate parameterizations that permit 1) interactive rendering of moderately sampled light fields of scenes with significant, unknown depth variation and 2) low-cost, passive autostereoscopic viewing. Using a dynamic reparameterization, these techniques can be used to interactively render photographic effects such as variable focus and depth-of-field within a light field. The dynamic parameterization is independent of scene geometry and does not require actual or approximate geometry of the scene. We explore the frequency domain and ray-space aspects of dynamic reparameterization, and present an interactive rendering technique that takes advantage of today's commodity rendering hardware.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing Algorithms I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures and data types

Additional Keywords: Image-based rendering, light field, lumigraph, ray space analysis, frequency domain analysis, autostereoscopic displays, synthetic aperture, depth of field, multitexturing

1 Introduction

The light field [13] and lumigraph [7] rendering methods synthesize novel images from a database of reference images. In these systems, rays of light are stored, indexed, and queried using a two-parallel plane parameterization [8]. Novel images exhibiting view-dependent shading effects are synthesized from this ray database by querying it for each ray needed to construct a desired view.

Several several shortcomings of the light field and lumigraph methods are addressed in this paper. At low to moderate sampling rates, a light field is only suitable for storing scenes with an approximately constant depth. A lumigraph uses depth-correction to reconstruct scenes with greater depth variation. However, it requires an approximate geometry of the scene which may be hard to obtain. Both systems exhibit static focus because they only produce a single reconstruction for a given queried ray. Thus, the pose and focal length of the desired view uniquely determine the image that is synthesized.

This paper presents solutions to the shortcomings stated above. Our goal is to represent moderately sampled light fields with wide variations in depth, without requiring geometry. This requires a more flexible parameterization of the ray database, based on a general mathematical formulation for a planar data camera array. To render novel views, our parameterization uses a generalized depth-correction based on focal surfaces. Because of the additional degrees of freedom expressed in the focal surfaces, we can interactively render images with dynamic photographic effects, such as depth-of-field and apparent focus. The presented dynamic reparameterization is as efficient as the static lumigraph and light field parameterizations, but permits more flexibility at almost no cost. To enable this additional flexibility, we do not perform aperture filtering as presented in [13]. We present a frequency domain analysis to explain the trade-offs of this omission.

Furthermore, our reparameterization techniques allow us to create directly-viewable light fields which are passively autostereoscopic. By using a fly's-eye lens array attached to a flat display surface, the computation for synthesizing a novel view can be solved directly by the optics of the display device. This three-dimensional display, based on integral photography [24, 16], requires no eye-tracking or special hardware attached to a viewer, and it can be viewed by multiple viewers simultaneously under variable lighting conditions.

2 Background

A continuous representation of a ray database would be sufficient for generating any desired ray. However, continuous databases are impractical or unattainable for all but the most trivial cases. In practice, we must work with finite representations in the form of discretely-sampled ray databases.

As with any sampling of a continuous signal, the issues of 1) choosing an appropriate initial sampling density and 2) defining a method for reconstructing the continuous signal are crucial factors in effectively representing the original signal. In the context of light fields and lumigraphs, researchers have explored various parameterizations and methods to facilitate better or more practical sampling [3, 22, 20, 4]. Other parameterizations have been presented to decrease the dimensionality of the light field, giving up vertical parallax and the ability to translate into the scene [21].

The choice of a ray database parameterization also affects the reconstruction methods that can be used in synthesizing desired views. Even with properly sampled data, a poor reconstruction filter can introduce post-aliasing artifacts into the result [15].

The two-parallel-plane parameterization of a ray database has a substantial impact on the choice of reconstruction filters. In the original light field system, a ray is parameterized by a predetermined entrance plane and exit plane (also referred to as the st and uv planes using lumigraph terminology). Figure 1 shows a typical moderate sampling on the st plane and three possible highly sampled exit planes, uv_1, uv_2, and uv_3. To reconstruct a desired ray

*http://graphics.lcs.mit.edu/∼aisaksen/projects/drlf

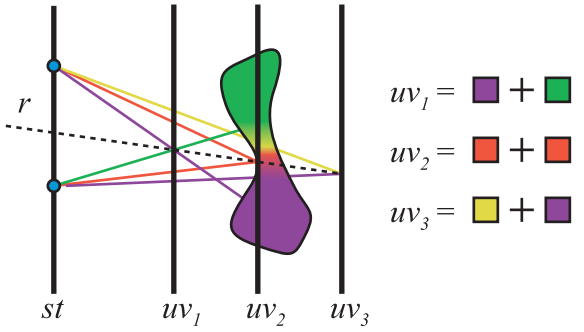

Figure 1: The parameterization of the exit plane, or uv plane, affects the reconstruction of a desired ray r. Here, the light field would be best parameterized using the uv_2 exit plane.

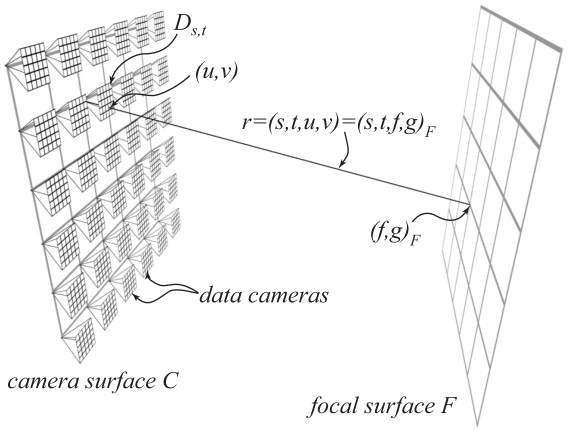

Figure 2: Our parameterization uses a camera surface C, a collection of data cameras $D_{s,t}$, and a dynamic focal surface F. Each ray r with coordinates (s,t,u,v) intersects the focal surface F at $(f,g)_F$ and is therefore also named $(s,t,f,g)_F$.

r which intersects the entrance plane at (s,t) and the exit plane at (u,v), a renderer combines samples with nearby (s,t) and (u,v) values. However, only the standard light field parameterized using exit plane uv_2 will give a satisfactory reconstruction. This is because the plane uv_2 well approximates the geometry of the scene,

The original light field system addresses this reconstruction problem by aperture filtering the ray database. Aperture filtering bandlimits the ray database with a low-pass prefilter. This removes high-frequency data that can not be reconstructed from a given sampling without aliasing. In Figure 1, aperture filtering on a ray database parameterized by either the exit plane uv_1 or uv_3 would store only a blurred version of the scene. Thus, any synthesized view of the scene will appear defocused. No post-processing can recover information that has been lost by aperture filtering. Because a particular fixed exit plane must be determined before performing aperture filtering, one can only represent scenes which can be sufficiently approximated by a single, fixed exit plane. In order to produce light fields that capture the full depth range of a deep scene without noticeable defocusing, the original light field system would require impractically large sampling rates.

The lumigraph system is able to reconstruct deep scenes stored at practical sampling rates by using depth-correction. In this process, the exit plane intersection coordinates (u,v) of each desired ray r are mapped to new coordinates (u',v') to produce an improved ray reconstruction. This mapping requires an approximate depth per ray, which can be efficiently stored as a polygonal model of the scene. If the geometry correctly approximates the scene, the reconstructed images will always appear in focus. The approximate geometry requirement imposes some constraints on the types of light fields that can be captured. Geometry is readily available for synthetic light fields, but acquiring geometry is difficult for photographically-acquired ray databases.

Both the light field and lumigraph systems are fixed-focus systems. That is, they will always produce the same result for a given geometric ray r. This is unlike a physical lens system which exhibits different behaviors depending on the focus setting and aperture size. In addition to proper reconstruction of a novel view, we would like to produce photographic effects such as variable focus and depth-of-field at interactive rendering rates. Systems have been built to render these types of lens effects using light fields, but this work was designed only for synthetic scenes where an entire light field is rendered for each desired image [11].

3 Focal Surface Parameterization

Our parameterization of ray databases is analogous to a two-dimensional array of pinhole cameras treated as a single optical system with a discrete synthetic aperture. Each constituent pinhole camera captures a focused image, and the camera array acts as a discrete aperture in the image formation process. By using an arbitrary focal surface, we can establish correspondences between the rays from different pinhole cameras.

In the two-parallel-plane ray database parameterization there is an entrance plane, with parameters (s,t) and an exit plane with parameters (u,v). Each ray r is uniquely determined by the 4-tuple (s,t,u,v).

Our parameterization is best described in terms of a camera surface, a 2-D array of data cameras and images, and a focal surface (see Figure 2). The camera surface C, parameterized by coordinates (s,t), is identical in function to the entrance plane of the standard parameterization. Each data camera $D_{s,t}$ represents a captured image taken from a given grid point (s,t) on the camera surface. Each $D_{s,t}$ can have a unique orientation and internal calibration, although we typically capture the light field using a common orientation and intrinsic parameters for the data cameras. We index each pixel in the data camera images using image coordinates (u,v), and we can think of each pixel (u,v) in a camera $D_{s,t}$ as a ray $r=(s,t,u,v)$. Samples in the ray database exist for values of (s,t) if there is a data camera $D_{s,t}$. The focal surface F is a dynamic two-dimensional manifold parameterized by coordinates $(f,g)_F$. Because the focal surface changes dynamically, we subscript the coordinates to tell us which focal surface we are using. Each ray (s,t,u,v) also intersects the focal surface F, and thus has an alternate naming $(s,t,f,g)_F$.

For each $D_{s,t}$ we define a mapping $\mathbf{M}_{s,t}^{F\rightarrow D}:(f,g)_F\rightarrow(u,v)$. This mapping tells us which data camera ray intersects the focal surface F at $(f,g)_F$. In other words, if $(f,g)_F$ was an imageable point on F, then the image of this point in camera $D_{s',t'}$ would lie at (u',v'), as in Figure 3. Given that we know the projection mapping $\mathbf{P}_{s,t}:(X,Y,Z)\rightarrow(u,v)$ which describes how three-dimensional points are mapped to pixels in the data camera $D_{s,t}$, and we know $\mathbf{T}_F:(f,g)_F\rightarrow(X,Y,Z)$ which maps points $(f,g)_F$ on the focal surface F to three-dimensional points in space, the mapping $\mathbf{M}_{s,t}^{F\rightarrow D}$ is easily determined, $\mathbf{M}_{s,t}^{F\rightarrow D}=\mathbf{P}_{s,t}\circ\mathbf{T}_F$.[1] Since the data cameras do not move or change their calibration, $\mathbf{P}_{s,t}$ is con-

[1] Since we create the focal surface at run time, we know $\mathbf{T}_F$. Likewise, $\mathbf{P}_{s,t}$ is known for synthetic light fields. For captured light fields, $\mathbf{P}_{s,t}$ can either be assumed or calibrated using readily available camera calibration techniques [26, 28].

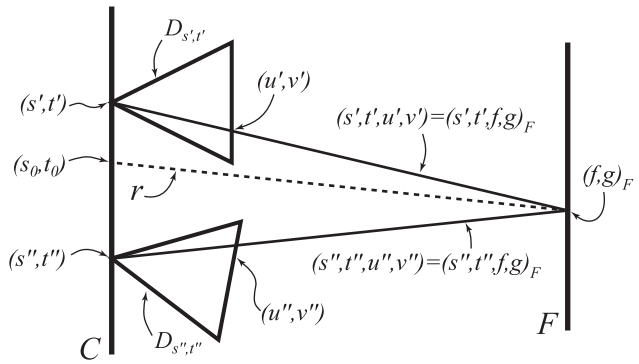

Figure 3: Given a ray $r = (s_0, t_0, f, g)_F$, we find the rays (s', t', u', v') and (s'', t'', u'', v'') in the data cameras $D_{s',t'}$ and $D_{s'',t''}$ which intersect F at the same point $(f, g)_F$.

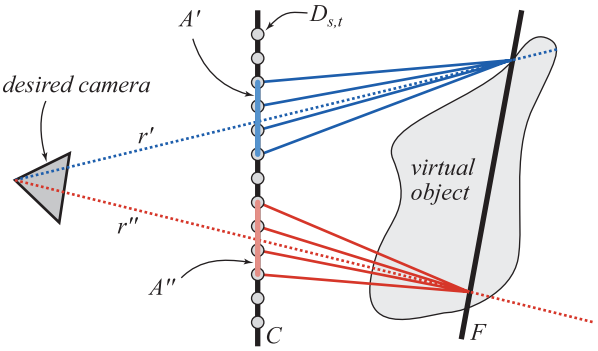

Figure 4: For each desired ray, our synthetic aperture system centers the aperture at the intersection of the ray with the camera surface. Thus the ray r' uses the aperture A' while r'' uses A''. The ray r' will appear in focus, while r'' will not.

stant for each data camera $D_{s,t}$. For a dynamic focal surface, we will modify the mapping $\mathbf{T}_F$, which changes the placement of the focal surface. A static $\mathbf{T}_F$ with a focal surface that conforms to the scene geometry gives us a depth-correction identical to the lumigraph [7].

To reconstruct a ray r from the ray database, we use a generalized depth-correction. We first find the intersections of r with C and F. This gives us the 4-D ray coordinates $(s_0, t_0, f, g)_F$ as in Figure 3. Using cameras near (s_0, t_0), say $D_{s',t'}$ and $D_{s'',t''}$, we apply $\mathbf{M}_{s',t'}^{F\to D}$ and $\mathbf{M}_{s'',t''}^{F\to D}$ to $(f, g)_F$, giving us (u', v') and (u'', v''), respectively. This gives us two rays (s', t', u'', v'') and (s'', t'', u'', v'') which are stored as the pixel (u', v') in the data camera $D_{s',t'}$ and (u'', v'') in the data camera $D_{s'',t''}$. We can then apply a filter to combine the values for these two rays. In the diagram, we have used two rays, although in practice, we can use more rays with appropriate filter weights.

4 Variable Aperture and Focus

We can use our dynamic parameterization to efficiently create images that simulate variable focus and variable depth-of-field. This allows us to create focused images of moderately sampled scenes with large depth variation without requiring geometric information. In addition, this new parameterization provides the user significant artistic expression when composing novel images.

4.1 Variable Aperture

In a traditional camera, the aperture controls the amount of light that can enter the optical system. It also influences the extent of depth-of-field present in the images. With smaller apertures, more of the scene appears in focus; larger apertures produce images with a narrow range of focus. We simulate synthetic apertures not to affect exposure, but to control the amount of depth-of-field present in an image.

We can interactively emulate a depth-of-field effect by combining rays from several cameras. In Figure 4, we are trying to reconstruct two rays, r' and r''. In this example, the extent of our synthetic apertures A' and A'' is four data cameras. We center the synthetic apertures at the intersection of r' and r'' with the camera surface C. We then recall ray database samples by applying $\mathbf{M}_{s,t}^{F\to D}$ for all (s, t) such that $D_{s,t}$ lies within the aperture. These samples are combined to create a single reconstructed ray.

Note that r' intersects F near the surface of the virtual object, whereas r'' does not. Our synthetic aperture reconstruction will cause r' to appear in focus, while r'' will not. The size of the synthetic aperture affects the amount of depth-of-field.

It is important to note that our model is not necessarily equivalent to an aperture attached to the desired camera. For example, if one rotates the desired camera, our effective aperture remains parallel to the camera surface. Modeling the aperture on the camera surface and not on the desired camera makes the ray reconstruction more efficient and still produces the desired depth-of-field effect (See Figure 5). A more realistic and complete lens model is given in [11], although this is less efficient to render and impractical for captured light fields.

We do not weight the queried samples that fall within the synthetic aperture equally. Using a dynamic filter that controls the weighting, we can improve the frequency response of our reconstruction. In Figure 6, we attempt to reconstruct the pink dotted ray $r = (s_0, t_0, f, g)_F$. We use a two-dimensional function $w(x, y)$ to describe the point-spread function of the synthetic aperture. Typically w has a maximum at $w(0, 0)$ and is bounded by a square of width δ. The filter is defined such that $w(x, y) = 0$ whenever $x \le -\delta/2$, $x \ge \delta/2$, $y \le -\delta/2$, or $y \ge \delta/2$. The filters should also be designed so that the sum of sample weights will add up to 1. That is, $\sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} w(x+i, y+j) = 1$ for all $(x, y) \in \Re^2$.

We use the aperture filter on the ray $r = (s_0, t_0, f, g)_F$ as follows. The center of the aperture filter is translated to the point (s_0, t_0). Then, for each camera $D_{s,t}$ that is inside the aperture, we will construct a ray $(s, t, f, g)_F$ and then calculate (s, t, u, v) using the appropriate mapping $\mathbf{M}_{s,t}^{F\to D}$. Then each ray (s, t, u, v) is weighted by $w(s - s_0, t - t_0)$ and all weighted rays within the aperture are summed together. [2]

Our system can create arbitrarily large synthetic apertures. The size of the aperture is only limited to the extent to which there are cameras on the camera surface. With sufficiently large apertures, we can see through objects, as in Figure 7. One problem with making large apertures occurs when the aperture function falls outside the populated region of the camera surface. When this occurs, the weighted samples will not add up to one. This creates a vignetting effect where the image darkens when using samples near the edges of the camera surface. This can be solved by either adding more data cameras to the camera surface or by reweighting the samples on a pixel by pixel basis so the weights always add up to one.

[2] One could also use the aperture function $w(x, y)$ as a basis function at each sample to reconstruct the continuous light field, although this is not computationally efficient.

Figure 5: By changing the shape and width of the dynamic aperture filter, we can interactively change the amount of depth of field.

4.2 Variable Focus

Photographers using cameras can not only change the depth-of-field, but they vary what is in focus. Using our dynamic parameterization, one can create the same effect at run-time by modifying the focal surface. As before, a ray r is defined by its intersections with the camera surface C and focal surface F and can be written $(s_0, t_0, f, g)_F$. The mapping $\mathbf{M}_{s,t}^{F \rightarrow D}$ tells us which ray (s, t, u, v) in the data camera $D_{s,t}$ approximates $(s_0, t_0, f, g)_F$.

When the focal surface is changed to F', the same ray r now intersects a different focal surface at a different point $(f', g')_{F'}$. This gives us a new coordinate $(s_0, t_0, f', g')_{F'}$ for the ray r. The new mapping $\mathbf{M}_{s,t}^{F' \rightarrow D}$ gives us a pixel (u', v') for each data camera $D_{s,t}$ within the aperture.

In Figure 9, we have three focal surfaces, F_1, F_2, and F_3. Note that for a single ray r, we reconstruct the sample using different pixels, depending on which focal surface we use. For example, if we are using focal surface F_n, then we will use the rays (s', t', u'_n, v'_n) and (s'', t'', u''_n, v''_n).

Note that this selection is a dynamic operation. In the light field and lumigraph systems, the ray r would always return the same reconstructed sample. As we see in Figure 8, we can effectively control which part of the scene is in focus by simply moving the focal surface. If the camera surface is too sparsely sampled, then the out-of-focus objects can appear aliased. This aliasing is analyzed in Section 6.

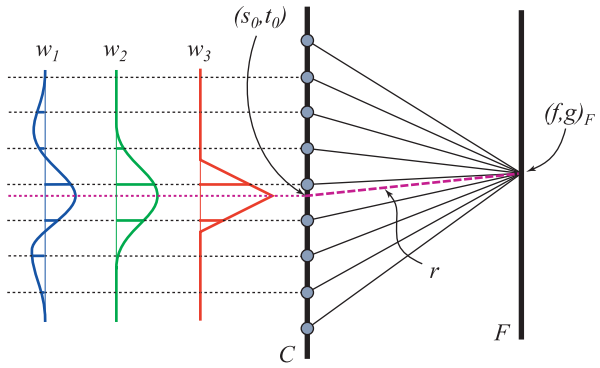

Figure 6: By changing the shape of our aperture filter, we can control the quality of reconstruction and the amount of depth-of-field. In this figure, filter w_1 will reconstruct r by combining six rays, w_2 will combine four rays, and w_3 will combine two.

Figure 7: With a very large aperture, we can see through objects. No one image in the light field sees the entire hillside.

Many scenes can not be brought entirely into focus with a single focal plane. As in Figure 9, the focal surfaces do not have to be planar. One could create a focal surface out of a parameterized surface patch that passes through key points in a scene, a polygonal model, or a depth map. Analogous to depth-corrected lumigraphs, this would insure that all visible surfaces are in focus. But, in reality, these depth maps would be hard and/or expensive to obtain with captured data sets. However, using a system similar to ours, a user can dynamically modify a non-planar focal surface until a satisfactory image is produced.

4.3 Multiple Apertures and Focal Surfaces

In general, we may like to have more than just the points near a single surface in clear focus. One solution is to use a different focal surface and aperture for each ray, something not available to real cameras. In a real lens system, only one continuous plane is in focus at one time. However, since we are not confined by physical optics, we can have two or more distinct regions that are in focus. Using a real camera, this can be done by first taking a set of pictures with different planes of focus, and then taking the best parts of each image and compositing them together as a post-process [17]. Using our parameterization, one can choose an aperture and focal surface on a per region or per pixel basis. We have previously described methods for rendering with multiple focal surfaces [12]. Multiple apertures would be useful to help reduce vignetting near the edges of the camera surfaces. If the aperture passes near the edge of the camera surface, then one could reduce its size so that it remains inside the boundary.

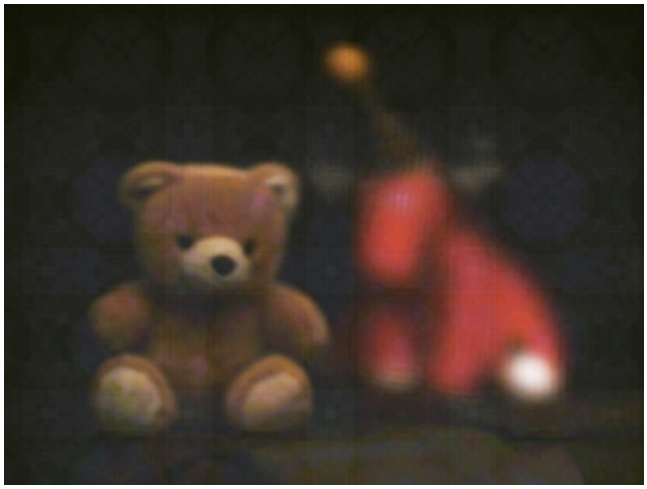

Figure 8: By varying the placement of the focal surface, one can interactively control what appears in focus.

5 Ray-Space Analysis

It is instructive to consider the effects of dynamic reparameterization on light fields when viewed from ray space [13, 7] and, in particular, within epipolar plane images (EPIs) [2]. It is well-known that 3-D structures of the observed scene are directly related to features within these particular light field slices. The two-parallel plane parameterization is particularly suitable for analysis under this regime as shown by [8]. Our system can also be analyzed in ray space, especially when the focal surface is planar. In our analysis, we consider a 2-D subspace of rays corresponding to fixed values of t and g on a dynamic focal plane F. When the focal surface is parallel to the camera surface, the sf slice is identical to an EPI.

A dynamically reparameterized light field with four point features is shown in Figure 10a. The dotted point is a point at infinity. A light field parameterized with the focal plane F_1 will have a sf_1 ray-space slice similar to Figure 10b. Each point feature corresponds to a linear feature in ray space, where the slope of the line indicates the relative depth of the point. Vertical features in the slice represent points on the focal plane; features with positive slopes represent points that are further away and negative slopes represent points that are closer. Points infinitely far away will have a slope of 1 (for example, the dashed line) Although not shown in the figure, variation in color along the line in ray space represents the view-dependent radiance of the point. If the same set of rays is reparameterized using a new focal plane F_2 that is parallel to the original F_1 plane, the sf_2 slice shown in Figure 10c results. These two slices are related by a shear transformation along the dashed line. If the focal plane is oriented such that it is not parallel with the camera surface, as with F_3, then the sf slice is transformed non-linearly, as shown in Figure 10d. However, each horizontal line of constant s in Figure 10d is a linearly transformed (i.e. scaled and shifted) version of the corresponding horizontal line of constant s in Figure 10b. In summary, dynamic reparameterization of light fields amounts to a simple transformation of ray space. When the focal surface remains perpendicular to the camera surface but its position is changing, this results in a shear transformation of ray space.

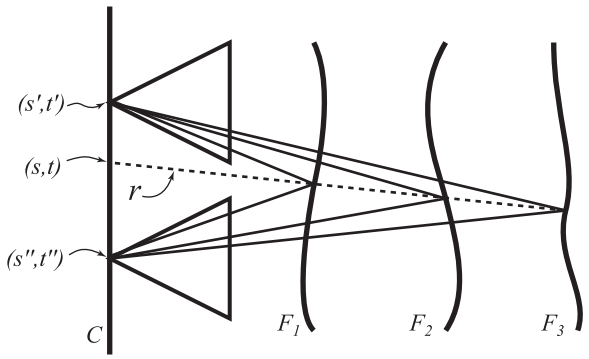

Figure 9: By changing the shape or placement focal surfaces, we can dynamically control which samples in each data camera will contribute to the reconstructed ray.

Changing the focal plane position thus affects which ray-space features will be axis-aligned. Thus, we can use a separable, axis-aligned reconstruction filter along with the focal plane to select which features will be axis-aligned, allowing us to dynamically select which features will be properly reconstructed. Equivalently, one can interpret focal plane changes as aligning the reconstruction filter to a particular feature slope, while keeping the ray space parameterization constant.

Under the interpretation that a focal plane shears ray space and keeps uses axis-aligned reconstruction filters, our aperture filtering methods amount to varying the extent of the reconstruction filters along the s dimension. In Figure 10e, the dashed horizontal lines depict the s extent of three different aperture filters (we assume they are infinitely thin in the f_1 dimension). When creating a line image from the ray-space using the three filters, we construct line images as shown in Figure 10f. Varying the extent of the aperture filter has the effect of "blurring" features located far from the focal plane while features located near on the focal plane will be relatively sharp. However, the filter will reduce the amount of view-dependent radiance for features aligned with the filter. If we shear ray space to produce the parameterization of Figure 10c and use the same three filters, we produce the line images of Figure 10g.

6 Frequency Domain Analysis

Ray space transformations have other effects on the reconstruction process. Since shears can arbitrarily modify the relative sampling frequencies between dimensions in ray space, they present considerable difficulties when attempting to bandlimit the source signal. Furthermore, any attempt to bandlimit the sampled function based on any particular parameterization will severely limit the fidelity of the reconstructed signals from the light field.

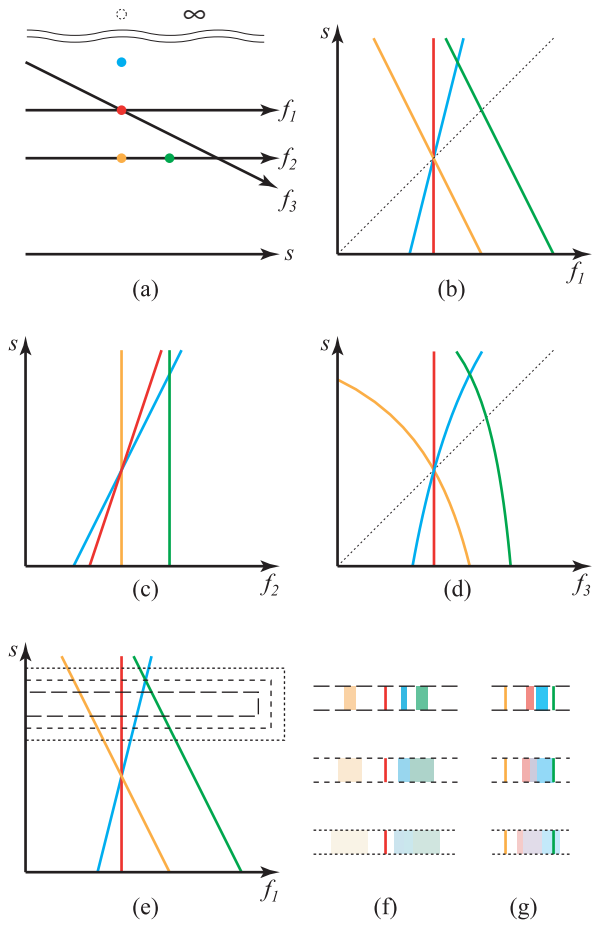

Figure 10: **(a)** A light field of four points, with 3 different focal planes. **(b,c,d)** sf slices using the three focal planes. **(e)** Three aperture filters drawn on the ray space of (b). **(f)** Line images constructed using the aperture filters of (e); red feature is in focus. **(g)** Line images constructed using the aperture filters of (e) but the ray space diagram of (c); orange and green features are in focus.

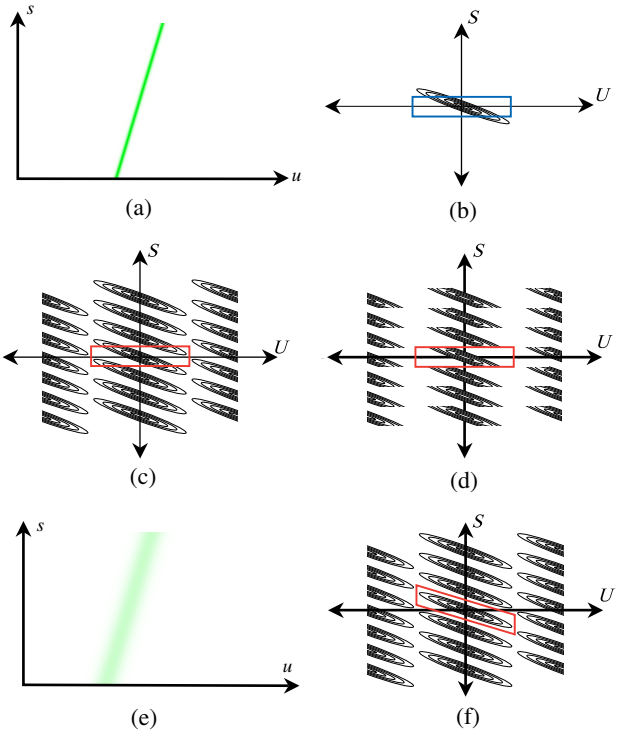

Figure 11: **(a)** su slice of a single feature. **(b)** Frequency domain power spectrum of (a). Aperture prefilter drawn in blue. **(c)** Power spectrum after typical sampling. Traditional reconstruction filter shown in red. **(d)** Power spectrum of sampled data after prefilter of (b). Traditional reconstruction filter shown in red. **(e)** In space domain, the result of (d) is a blurred version of (a). **(f)** By using alternative reconstruction function filter, we can accurately reconstruct (a).

In this section, we will analyze the frequency-domain dual of a dynamically reparameterized light field. Whereas in section 5 we interpreted dynamic reparameterization as ray space shearing, in this section we will interpret the ray space as fixed (using dimensions s and u) and instead shear the reconstruction filters.

Consider an 'ideal' continuous light field of a single feature located slightly off the u plane as shown in the EPI of Figure 11a. In the frequency domain, this su slice will have the power spectrum shown in Figure 11b. The blue box represents a bandlimiting prefilter. Sampling generates copies of this spectrum as shown in Figure 11c. Typical light fields have a higher sampling density on the data camera images than on the camera surface, and our example reflects this convention. If the data camera images are adequately sampled, there will be no overlap between the copies in the horizontal direction of the frequency domain. If we attempt to reconstruct this signal with the separable reconstruction filter under the original parameterization shown by the red box in Figure 11c, the resulting image will exhibit considerable post-aliasing, because of the high-frequency leakage from the other copies. This quality degradation will show up as ghosting in the reconstructed image, where multiple copies of a feature can be faintly seen. Note that this ghosting is a form of post-aliasing; the original sampling process has not lost any information.

One method for remedying this problem is to apply an aggressive bandlimiting pre-filter to the continuous signal before sampling. This approach is approximated by the aperture-filtering step described in [13]. When the resulting bandlimited light field is sampled using the prefilter of Figure 11b, the power spectrum shown in Figure 11d results. This signal can be reconstructed exactly with an ideal separable reconstruction filter as indicated by the red box. However, the resulting EPI, shown in Figure 11e, contains only the low-frequency portion of the original signal energy, giving a blurry image.

Dynamic reparameterization allows many equally valid reconstructions of the light field. The shear transformation of the ray-space filter effectively allows for the application of reconstruction filters that would be non-separable in the original parameterization. Thus, using dynamic reparameterization, the spectrum of the single point can be recovered exactly without post-aliasing using the filter indicated by the red box in Figure 11f.

Issues are more complicated in the case when multiple point features are represented in the light field, as shown in the su slice in Figure 12a. The power spectrum of this signal is shown in Figure 12b. After sampling, multiple copies of the original signal's spectrum interact, causing a form of pre-aliasing that cannot be undone by processing. Dynamic reparameterization allows for a single feature from the spectrum to be extracted, as shown by the red box overlaid on Figure 12c. However, some residual energy from the other points will also be captured, and will appear in the recon-

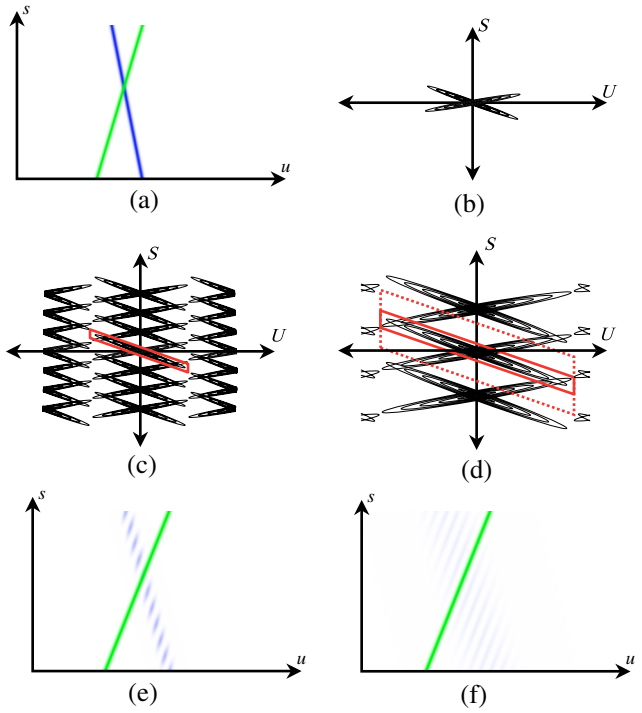

Figure 12: **(a)** su slice of two features. **(b)** Frequency domain power spectrum of the features. **(c)** Frequency domain power spectrum, the red box represents a possible reconstruction filter. **(d)** Wide aperture reconstruction corresponds to the thinner filters in frequency domain. **(e)** Result of small aperture reconstruction. **(f)** Result of large aperture reconstruction.

structed image as ghosting (see Figure 12e).

One method for reducing this artifact is to increase the size of the synthetic aperture. In the frequency domain, this reduces the width of the reconstruction filters as shown in Figure 12d. Using this approach, we can, in the limit, reduce the contribution of spurious features to a small fraction of the total extracted signal energy. The part we cannot extract is the result of the pre-aliasing. By choosing sufficiently wide reconstruction apertures (or narrow in the frequency domain), the effect of the pre-aliasing can be made imperceptible (below our quantization threshold). Figure 12f is reconstructed by using a wider aperture than that in Figure 12e. Note that the aliasing in Figure 12f has less energy and is more spread out than in Figure 12e.

This leads to a general trade-off that must be considered when working with moderately sampled light fields. We can either 1) apply prefiltering at the cost of limiting the range of images that can be synthesized from the light field and endure the blurring and attenuation artifacts that are unavoidable in deep scenes or 2) endure some aliasing artifacts in exchange for greater flexibility in image generation. The visibility of aliasing artifacts can be effectively limited by selecting appropriate apertures for a given desired image.

7 Rendering

As in the lumigraph and light field systems, we can construct a desired image by querying rays from the ray database. Given arbitrary cameras, camera surfaces, and focal surfaces, one can ray-trace the desired image. If the desired camera, data cameras, camera surface, and focal surface are all planar, then a texture mapping approach can be used similar to that proposed by the lumigraph system. We extend the texture mapping method using multi-texturing for rendering with arbitrary non-negative aperture filters, including bilinear and higher order filters.

7.1 Memory Coherent Ray Tracing

We first describe a memory coherent ray tracing method with the following pseudo-code. Instead of rendering pixel by pixel in the desired image, we can render the contribution of each data camera sequentially. This causes us to write to each pixel in the desired image many times.

The intersection techniques are those used in standard ray tracing. In the following description, a ray $r = (s, t, u, v)$ has a color $c(r) = c(s, t, u, v)$. Likewise, a pixel (x, y) in the desired image has a color $c(x, y)$. Let K be the desired camera with a center of projection o and pixels (x, y) on its image plane. Let $w(x, y)$ be the aperture weighting function, where δ is the width of the aperture.

Initialize the frame buffer to black
For each data camera $D_{s,t}$
  $R_C :=$ a polygon on C defined by $\{(s \pm \delta/2, t \pm \delta/2)\}$
  $R_K :=$ projection of R_C onto the desired image plane
  For each pixel (x, y) within R_K
    $r :=$ the ray through o and (x, y)
    Intersect r with C and F to get (s', t') and $(f, g)_F$
    $(u, v) := \mathbf{M}_{s,t}^{F \to D}(f, g)_F$
    $weight := w(s' - s, t' - t)$
    $c(x, y) := c(x, y) + weight * c(s, t, u, v)$

7.2 Texture Mapping

Although the ray tracing method is simple to understand and easy to implement, there are more efficient methods for rendering when the camera surface, image surface, and focal surface are planar. We extend the lumigraph texture mapping approach [7] to support dynamic reparameterization. We render the contribution of each data camera $D_{s,t}$ using multi-texturing and an accumulation buffer [9]. Our method works with arbitrary non-negative aperture functions.

Multi-texturing, supported by Microsoft Direct3D 7's texture stages [14], allows a single polygon to have multiple textures and multiple projective texture coordinates. At each pixel, two sets of texture coordinates are calculated, and then two texels are accessed. The two texels are multiplied, and the result is stored in the frame buffer. We write to the frame buffer using the Direct3D alpha mode "source + destination," which makes the frame buffer act as a 8-bit, full-color accumulation buffer.

Our rendering technique is illustrated in Figure 13. For each camera $D_{s,t}$, we create a rectangular polygon $R_{s,t}^C$ on the camera surface with coordinates $\{(s \pm \delta/2, t \pm \delta/2)\}$. We then project this polygon on to the desired camera K's image plane using a projection matrix $\mathbf{P}_{C \to K}$, giving us a polygon $R_{s,t}^K$. This polygon $R_{s,t}^K$ represents the region of the image plane which uses samples from the data camera $D_{s,t}$. That is, only pixels inside polygon $R_{s,t}^K$ will use texture from data camera $D_{s,t}$.

We then project $R_{s,t}^K$ onto the focal plane F using a planar homography $\mathbf{H}_{K \to F}$, a 3x3 matrix which changes one projective 2-D basis to another. This projection is done from the desired camera K's point of view. The resulting polygon $R_{s,t}^F$ lies on the focal plane F. Finally, we use the mapping $\mathbf{M}_{s,t}^{F \to D}$ to calculate the (u, v) pixel values for the polygon. This gives us a polygon $R_{s,t}^D$, which represents the (u, v) texture coordinates for polygon $R_{s,t}^F$.

We can compose many of these operations into a single matrix, which takes us directly from polygon $R_{s,t}^C$ to texture coordinates $R_{s,t}^D$. This matrix $\mathbf{M}_{s,t}^{C \to D}$ can be written as $\mathbf{M}_{s,t}^{C \to D} = \mathbf{M}_{s,t}^{F \to D} \mathbf{H}_{K \to F} \mathbf{P}_{C \to K}$.

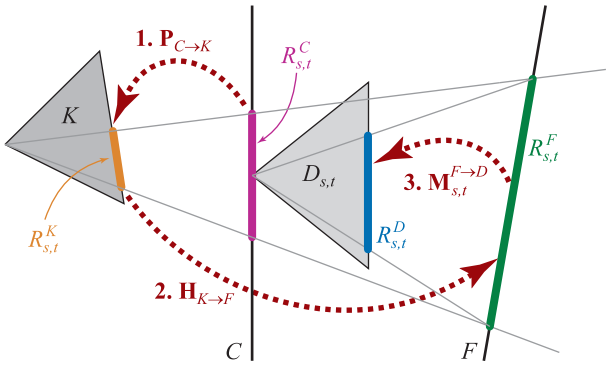

Figure 13: Projection matrices and planar homographies allow us to render the image using texture mapping on standard commodity PC rasterizing hardware.

This process gives us the correct rays (s, t, u, v), but we still require the appropriate weights from the aperture filter. Because we are drawing a polygon with the shape of the aperture filter, we can simply modulate the texture $D_{s,t}$ with the aperture filter texture A. For $D_{s,t}$ we use the projective texture coordinates $R^D_{s,t}$; for the aperture filter A, we use the texture coordinates $\{(\pm\delta/2, \pm\delta/2)\}$.

Initialize the frame buffer to black
For each data camera $D_{s,t}$
  $R^C_{s,t}$:= polygon on C defined by $\{(s \pm \delta/2, t \pm \delta/2)\}$
  $R^D_{s,t} := \mathbf{M}^{F\to D}_{s,t}\mathbf{H}_{K\to F}\mathbf{P}_{C\to K}R^C_{s,t}$
  Render $R^C_{s,t}$ using...
    texture $D_{s,t}$
    projective texture coordinates $R^D_{s,t}$
    modulated by aperture texture A
  Accumulate rendered polygon into frame buffer

Using this method, dynamically reparameterized light fields can be rendered in real-time on readily available PC graphics cards that support multi-texturing. Frame rate decreases linearly with the number of data cameras that fit inside the aperture functions, so narrow apertures render faster. Vignetting occurs near the edges of the camera surface when using wide filters.

7.3 Using the Focal Surface as a User Interface

In typical light field representations, there is no explicit depth information making it difficult to navigate about an object using a keyboard or mouse. For example, it can be hard to rotate the camera about an object when we don't know where it is located in space. Head tracking can make navigation simpler, although specialized tracking hardware is considerably less accessible [19]. We have found the focal surface can be used to help navigate about an object in the light field. When we move the focal surface so that a particular pixel p belonging to that object is in focus, we can find the 3-D position P of p using the equation $P = \mathbf{H}_{K\to F}p$. Once we know the effective 3-D position of the object, we can rotate (or some other transformation) relative to that point.

8 Autostereoscopic Light Fields

Our flexible reparameterization framework allows for other useful reorganizations of light fields. One interesting reparameterization permits direct viewing of a light field. The directly-viewed light field is similar to an integral or lenticular photograph. In integral photography, a large array of small lenslets, usually packed in a hexagonal grid, is used to capture and display an autostereoscopic image [16, 24]. Traditionally, two identical lens arrays are used to capture and display the image: this requires difficult calibration and complicated optical transfer techniques [27]. Furthermore, the viewing range of the resulting integral photograph mimics the configuration of the capture system. Holographic stereograms [10] also can present directly-viewed light fields, although the equipment and precision required to create holographic stereograms can be prohibitive. Using our reparameterizations, we can capture a scene using light field capture hardware, reparameterize it using our techniques, and thus create novel 3-D views which can be redisplayed using standard lens arrays with few restrictions. This makes it much easier to create integral photographs since a light field is much easier to collect and a variety of different integral photographs can be created from the same light field.

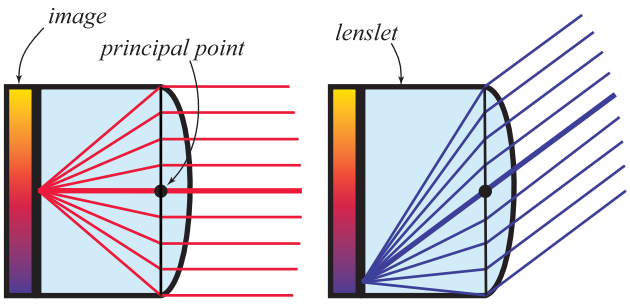

Figure 14: Each lenslet in the lens array acts as a view-dependent pixel. One can determine which color will be seen by drawing a ray through the principal point of the lenslet.

In an integral photograph, a single lenslet acts as a view-dependent pixel, as seen in Figure 14. For each lenslet, the focal length of the lens is equal to the thickness of the lens array. A reparameterized light field image is placed behind the lens array, such that a subset of the ray database lies behind each lenslet. When the lenslet is viewed from a particular direction, the entire lenslet takes on the color of a single point in the image. To predict which color will be seen from a particular direction, we use a paraxial lens approximation [23]. We can draw a line parallel to the viewing direction which passes through the principal point of the lenslet. This line will intersect the image behind the lenslet at some point; this point determines the view-dependent color. If the viewing direction is too steep, then the intersection point might fall under a neighboring lenslet. This causes a repeating "zoning" pattern which can be eliminated by limiting the viewing range or by embedding blockers in the lens array.

Since each lenslet acts as a view-dependent pixel, the entire lens array acts as a view-dependent, autostereoscopic image. The complete lens array system can be seen in Figure 16. Underneath each lenslet is a view of the object from a virtual camera located at each lenslet's principal point.

To create the autostereoscopic image from a dynamically reparameterized light field, we position a model of the lens array into our light field scene. This is analogous to positioning a desired camera to take a standard image. We then create an array of tiny sub-images, each the size of a lenslet. Each sub-image is created using our dynamically reparameterized light field system, with the focal surface passing through the object of interest. Each sub-image is taken from the principal point of a lenslet, with the image plane parallel to the flat face of the lens array. The sub-images are then composited together to create a large image, as in the background of Figure 15, which can be placed under the lens array.

The placement and orientation of the lens array determines if the viewed light field will appear in front or behind the display. If the

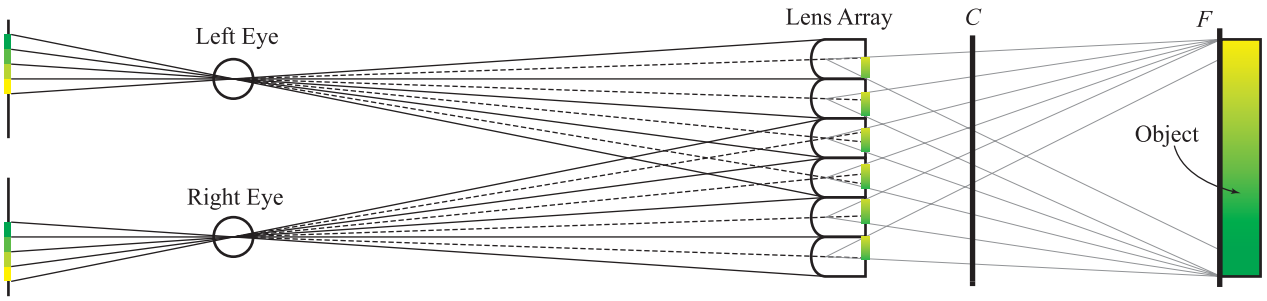

Figure 16: A light field can be reparameterized into a directly viewed light field, which operates on the principals of integral photography.

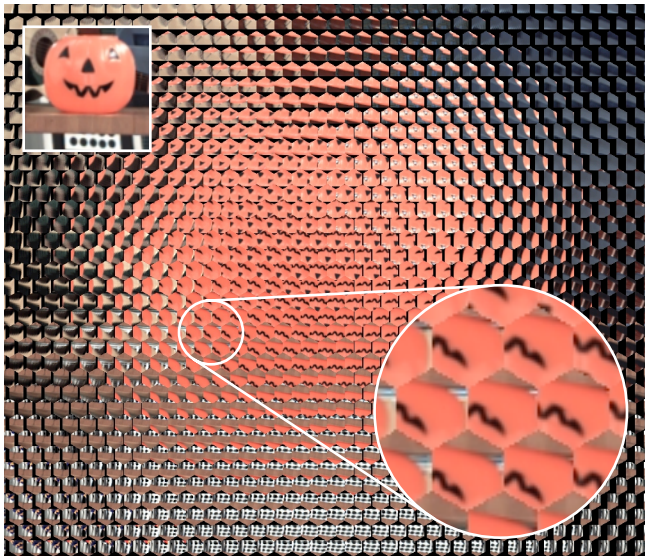

Figure 15: An autostereoscopic light field is drawn in the background. The scene is a three-dimensional version of the small inset picture in the upper left corner. The zoom is 400% magnification.

lens array is placed in front of the object, then the object will appear behind the display. Because the lens array image is rendered from a light field and not directly from an integral camera, we can place the lens array image behind the captured object, and the object will appear to float in front of the display.

9 Results

The light field data sets shown in this paper were created as follows. The tree data set was rendered in Povray 3.1. It is composed of 256 (16 x 16) images with resolutions of 320 x 240. The captured data sets were acquired with an Electrim EDC1000E CCD camera (654 x 496) with a fixed-focal length 16mm lens mounted on an X-Y motion platform from Arrick Robotics (30" x 30" displacement). For each set we captured either 256 (16 x 16) or 1024 (32 x 32) pictures. To calibrate the camera, we originally used a Faro Arm (a sub-millimeter accurate contact digitizer) to measure the spatial coordinates of targets on a three-dimensional calibration pattern. Using an image of the targets, we used the Tsai-Lenz camera calibration algorithm [26] which reported focal length, CCD sensor element aspect ratio, principle point, and extrinsic rotational orientation. The radial lens distortion reported for our lens was less than 1 pixel per 1000 pixels, and we decided not to correct for it. Finally, we resampled the raw 654 x 496 images down to 327 x 248 before using them as input to the renderer.

Recently, we have experimented with an alternative calibration method that requires no 3D measurements and uses the actual light field images rather than images of a special calibration object. This approach has some similarities with a family of techniques known as "self-calibration" [5]. However, our light field capture system, where a non-rotating camera is translated in a plane, is a degenerate case for true self-calibration [25]. Instead, we align the epipolar geometries of the source images rather than compute an actual calibration. Our method amounts to a two-axis rectification [1]. If the camera's optical axis is roughly perpendicular to the plane of motion and the X-Y platform's motion is reasonably accurate, then such a rectification is easily found. We find the epipolar planes induced by the horizontal and vertical camera motion by tracking a few corresponding image points. This can be done either automatically or by hand. The images are then rectified by a rotating their epipoles onto the line at infinity. The final rectified images will have valid horizontal and vertical EPI structures and can be immediately used in our light field viewer. However, when a user navigates within the scene they might notice a projective distortion of the space. This distortion can be ameliorated by allowing the user to interactively adjust the focal length of the data cameras. Otherwise, the focal length can be estimated by measuring a few points in the scene. In our experiments, it has been easy to create light fields using this method, and the final results are comparable to our strictly calibrated data sets.

Our autostereoscopic images were printed at 300dpi on a Tektronics Phaser 440 dye-sublimation printer and use a Fresnel Technologies #300 Hex Lens Array, with approximately 134 lenses per square inch [6].

10 Future Work

Light fields contain a large amount of redundancy which we would like to exploit. We would like to develop an algorithm for optimally selecting a focal plane, perhaps using auto-focus techniques similar to those used in consumer camcorders. Currently, the focal plane must be placed manually. In addition, we believe there is some promise in using our reparameterization techniques for passive depth-from-focus or depth-from-defocus vision algorithms [18]. In the left column of Figure 17, we have created two images with different focal surfaces and a large aperture. We then apply a gradient magnitude filter to these images, which give us the output to the right. These edge images tell us where in-focus, high-frequency energy exists. We would also like to experiment with depth-from-defocus by comparing two images with slightly different focal planes or apertures.

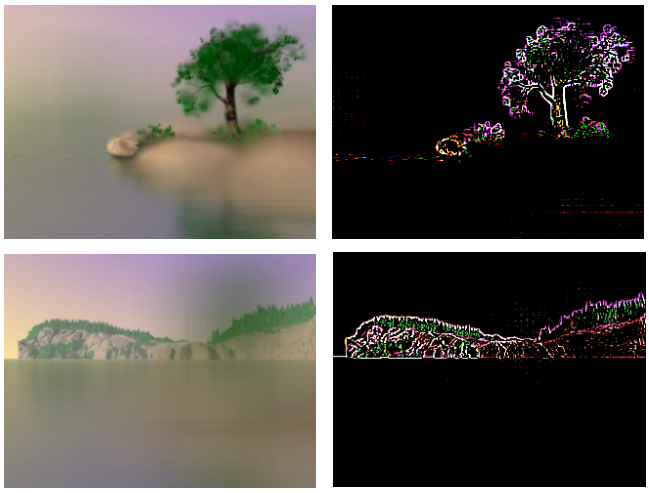

Figure 17: We believe our techniques can be used in a depth-from-focus or depth-from-defocus vision system. By applying a gradient magnitude filter on an image created with a wide aperture, we can detect in-focus regions.

Figure 18: Moving the focal plane within a deep scene.

11 Conclusion

Previous implementations have tried to solve focusing problems in undersampled light fields by 1) using scenes that were roughly planar, 2) using aperture filtering to bandlimit the input data, or 3) using approximate geometry for depth-correction. Unfortunately, most scenes can not be confined to a single plane, aperture filtering can not be undone or controlled at run time, and proxy surfaces can be difficult to obtain. We have presented a new parameterization that enables dynamic, run-time control of the sample reconstruction. This allows the user to modify focus and depth-of-field dynamically. This new parameterization allows light fields to capture deep scenes at moderate sampling rates. In addition, we have presented a strategy for creating directly-viewable light fields. These passive, autostereoscopic light fields can be viewed without head-mounted hardware by multiple viewers simultaneously under variable lighting conditions.

12 Acknowledgments

This work was supported by NTT, Fresnel Technologies, Hughes Research, and an NSF fellowship. Thanks to Ramy Sadek and Annie Sun Choi for illustrations, assistance, support, and proofreading. Thanks to Robert W. Sumner for his LaTeX experience and to Chris Buehler for many helpful light field discussions and DirectX help. Thanks to Neil Alexander for the "Alexander Bay" tree scene and to Christopher Carlson for permission to use Bumpé in our scenes. Finally, thanks to the entire MIT LCS Computer Graphics Group, especially Bryt Bradley for her administrative support.

References

[1] Nicholas Ayache and Charles Hansen. Rectification of images for binocular and trinocular stereovision. In *ICPR88*, pages 11–16, 1988.

[2] R.C. Bolles, H.H. Baker, and D.H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *IJCV*, 1(1):7–56, 1987.

[3] Emilio Camahort, Apostolos Lerios, and Donald Fussell. Uniformly sampled light fields. *Eurographics Rendering Workshop 1998*, pages 117–130, 1998.

[4] J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum. Plenoptic sampling. In *SIGGRAPH 2000*, 2000.

[5] Olivier D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *ECCV92*, pages 563–578, 1992.

[6] Fresnel Technologies. *#300 Hex Lens Array*, 1999. http://www.fresneltech.com.

[7] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. *SIGGRAPH 96*, pages 43–54, 1996.

[8] Xianfeng Gu, Steven J. Gortler, and Michael F. Cohen. Polyhedral geometry and the two-plane parameterization. *Eurographics Rendering Workshop 1997*, pages 1–12, June 1997.

[9] Paul E. Haeberli and Kurt Akeley. The accumulation buffer: Hardware support for high-quality rendering. *SIGGRAPH 90*, 24(4):309–318, 1990.

[10] Michael Halle. The generalized holographic stereogram. Master's thesis, Program in Media Arts and Sciences, Massachusetts Institute of Technology, 1991.

[11] Wolfgang Heidrich, Philipp Slusalek, and Hans-Peter Seidel. An image-based model for realistic lens systems in interactive computer graphics. *Graphics Interface '97*, pages 68–75, 1997.

[12] Aaron Isaksen, Leonard McMilland, and Steven J. Gortler. Dynamically reparameterized light fields. Technical Report LCS-TR-778, Massachusetts Institute of Technology, May 1999.

[13] Marc Levoy and Pat Hanrahan. Light field rendering. *SIGGRAPH 96*, pages 31–42, 1996.

[14] Microsoft Corporation. *Microsoft DirectX 7.0*, 1999. http://www.microsoft.com/directx.

[15] Don P. Mitchell and Arun N. Netravali. Reconstruction filters in computer graphics. *SIGGRAPH 88*, 22(4):221–228, 1988.

[16] Takanori Okoshi. *Three-Dimensional Imaging Techniques*. Academic Press, Inc., New York, 1976.

[17] Paul Haeberli. A multifocus method for controlling depth of field. Technical report, SGI, October 1994. http://www.sgi.com/grafica/depth/index.html.

[18] A.P. Pentland. A new sense for depth of field. *PAMI*, 9(4):523–531, July 1987.

[19] Matthew J.P. Regan, Gavin S.P. Miller, Steven M. Rubin, and Chris Kogelnik. A real time low-latency hardware light-field renderer. *SIGGRAPH 99*, pages 287–290, 1999.

[20] Hartmut Schirmacher, Wolfgang Heidrich, and Hans-Peter Seidel. Adaptive acquisition of lumigraphs from synthetic scenes. *Computer Graphics Forum*, 18(3):151–160, September 1999. ISSN 1067-7055.

[21] Heung-Yeung Shum and Li-Wei He. Rendering with concentric mosaics. *SIGGRAPH 99*, pages 299–306, 1999.

[22] Peter-Pike Sloan, Michael F. Cohen, and Steven J. Gortler. Time critical lumigraph rendering. *1997 Symposium on Interactive 3D Graphics*, pages 17–24, April 1997. ISBN 0-89791-884-3.

[23] Warren J. Smith. *Practical Optical System Layout*. McGraw-Hill, 1997.

[24] R. F. Stevens and N. Davies. Lens arrays and photography. *Journal of Photographic Science*, 39(5):199–208, 1991.

[25] Peter Sturm. Critical motion sequences for monocular self-calibration and uncalibrated euclidean reconstruction. In *CVPR97*, pages 1100–1105, 1997.

[26] R.Y. Tsai. An efficient and accurate camera calibration technique for 3-d machine vision. In *CVPR*, pages 364–374, 1986.

[27] L. Yang, M. McCormick, and N. Davies. Discussion of the optics of a new 3d imaging system. *Applied Optics*, 27(21):4529–4534, 1988.

[28] Z.Y. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *ICCV*, pages 666–673, 1999.