CHAPTER 4

An Introduction to Light Fields

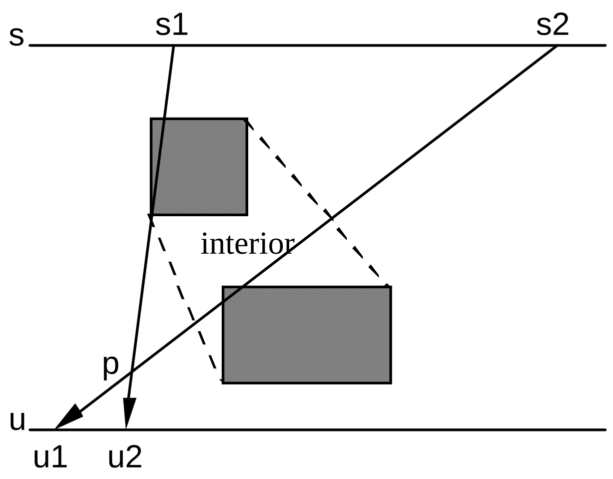
Introduction

Levoy and Hanrahan, 1996 and also Gortler et al., 1996 provided an interesting image-based solution to the radiance equation (EQ 75) under a number of restrictions. The radiance equation is a recursive expression for radiance $L(p, \omega)$ where p is any surface point and ω is the set of all directions. Any specific p and ω together form a ray, hence we can think of $L(p, \omega)$ being defined over the set of all rays with origins at points on surfaces. Hence the domain of L is a five dimensional ray space.

However, radiance is constant along a given direction in ‘free space’, that is, where there are no discontinuities due to intersections with objects. Hence radiance is constant along each direction outside the convex hull of a scene. Figure 17 shows a simple 2D example. Consider consider the set of all rays with origin on line *s* which terminate on line *u*. All such rays intersecting the scene consisting of the two shaded boxes can be parameterised in 2D using (*s*,*u*). Suppose that a radiance value were (somehow) assigned to each such (*s*,*u*), i.e., that *L*(*s*,*u*) was known. Now to form an image, we place an image plane and centre of projection in the scene, and collect all those rays passing through the COP which intersect the image plane.

Clearly this parameterisation is only valid for views outside the *convex hull* of the scene, for it is only for these rays that the radiance is constant. Views from anywhere in the interior would not be possible, since the radiance abruptly changes along such paths.

FIGURE 17. **Parameterising the Space of all Rays in Free Space**

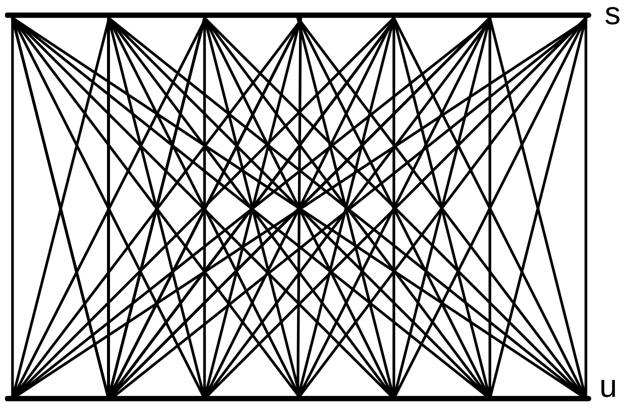


For example, suppose an image for a pin-hole camera were required from point *p* for the view ‘volume’ within the arrowed lines. Then by capturing the radiance from all the rays between the arrowed lines such an image could be constructed.

In order to construct the complete set of rays (for the convex hull of the scene) four copies of the ‘light slab’ defined by *s* and *u* would be needed - one as shown, another with the directions reversed, and two others with, for example, two vertical lines on either side of the scene. Nevertheless, the overall representation of the lumigraph is still two-dimensional, albeit with four such two-dimensional representations.

Such a representation is called a *light field.* A *lumigraph* is the same idea, though with some additional information.

FIGURE 18. Discrete Representation of a Light Field (One Light Slab)



Staying with this two-dimensional example, and with the single 'light slab' for *s* and *u*, the light field radiance would be a continuous function of *s* and *u*, $L(\mathbf{s},u)$. For computational purposes a discrete representation is required. The *two-line parameterisation* is illustrated in Figure 18 (with a 2D analogue). Grids are imposed on the *s* and *u* axes, and then all possible lines between each grid point on one to each grid point on the other forms the discrete representation of the set of all such lines.

The radiance attached to each ray in the light-field would be estimated by choosing each grid point s_i on *s* as a centre of projection with *u* as the image plane (and a fixed interval of *u* as the view plane window). In this way, every ray that intersects the convex hull of the scene would have an associated radiance determined by the 'rendering' of the image formed by the set of perspective projections.

For 3D scenes the 'two-line' parameterisation is replaced by a *two-plane parameterisation* (2PP), the first parameterised by (s,t) and the second by (u,v). The light field (in fact a sub-set of it) is then represented by $L(\mathrm{s},t,u,v)$, and discretised by all possible lines between the two planes defined by a rectangular grid imposed on each. The *st* plane is subdivided into squares with each vertex forming a centre of projection, with a rectangular subset of the *uv* plane as the view plane window to form an associated image. Hence there is an image associated with each grid point of the *st* plane, and a radiance associated with each (s,t,u,v) combination representing lines that intersect the convex hull of the scene. This describes how to form one light slab. In order to cover all possible ray directions, six copies are needed - three orthogonal rotations, and two directions in each case.

The light field once constructed can be used for synthesis of an image from a virtual camera which does not correspond to the set of cameras on the *st* planes. A new image can be formed by sampling the corresponding set of lines through the viewpoint and in the required directions.

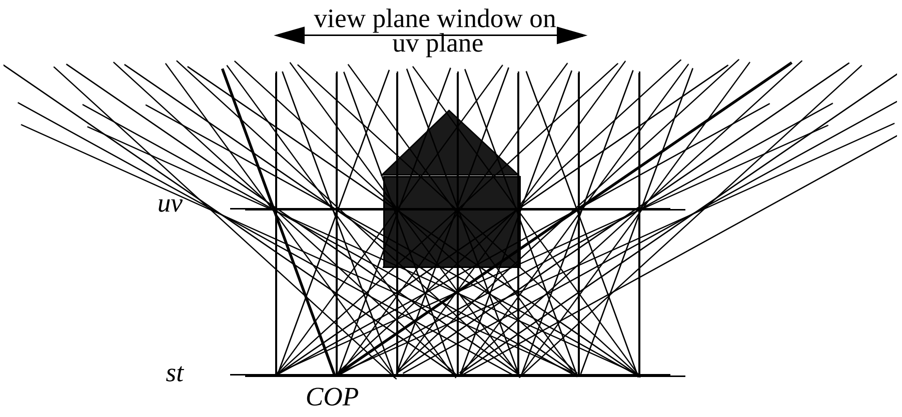
The light field approach was designed as a method of forming new views from images of real scenes. Suppose digital photographs were taken under strict conditions forming the set of viewpoints and directions associated with the st planes. Such a set of images can clearly be used to construct a light field. A virtual light field, that is a light field for virtual scenes, can also be constructed by using some other rendering system in order to form the images. The only possible advantage of this is where the rendering system includes global illumination with a mixture of diffuse and specular surfaces. The light field approach then provides, in principle, a means for real-time walkthrough of a globally illuminated scene - something which is still not possible in any other way.

In the next sessions we consider some issues in more detail. We mainly consider one slight slab parameterised by *st* and *uv.* The extensions to multiple light slabs are obvious.

Rendering: Interpolation

Suppose that the *st* plane is discretised with an $M \times M$ grid, and the *uv* plane with $N \times N$. Then the light field will consist of M^2 perspective images, each of resolution N^2 .This is illustrated in 2D in Figure 19, the *st* point marked COP will have an associated image with view volume between the two thicker lines. All rays converging on this COP will therefore have an associated radiance determined by this image. The image might either have been formed by placing a digital camera in a real scene, or by rendering a virtual scene with the given COP and view volume. The COP is shifted to each point on the *st* plane in turn, thus completing the full light slab. The whole process must be repeated for each of five other light slabs forming the full light field in order to attain a complete ray coverage.

FIGURE 19. **An Image is Created for Each Point on the *st* Plane**



Once the light field has been created it may be used for rendering images from camera positions and orientations outside the convex hull of the scene. First we describe the basic way in which this is carried out.

FIGURE 20. **Synthesising an Image for a New View**

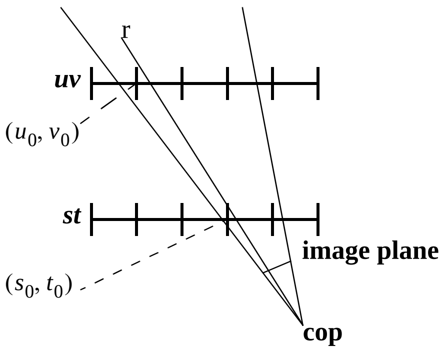
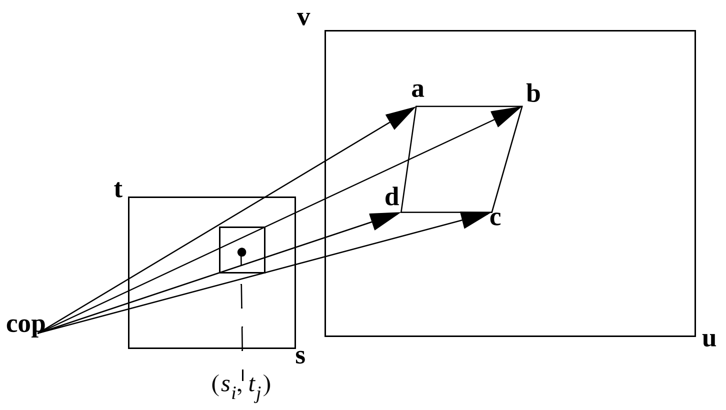


Figure 20 shows the *st* and *uv* planes and a new camera position and orientation. Each ray through a pixel of the virtual camera will intersect the *st* and *uv* planes. For example, the ray *r* intersects the *st* plane at (s_0, t_0) and the *uv* plane at (u_0, v_0). Hence the pixel corresponding to that ray would be set with the value of

$L(s_0, t_0, u_0, v_0)$. In other words, each primary ray generated from the camera, is used to look up the nearest ray in the 4D space of rays corresponding to the light field.

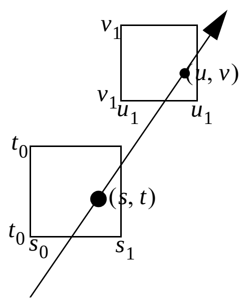
This method can be implemented very efficiently using texture mapping. Figure 22 shows one grid point (s_i, t_j) on the *st* plane, and a square containing all the points on the plane that would be approximated by this particular grid point. Now the square projects to points *a*, *b*, *c*, *d* on the *uv* plane. Corresponding to (s_i, t_j) there is an image on the *uv* plane. This image may be used as a texture to render the *st* square for (s_i, t_j), with texture coordinates given by *a*, *b*, *c*, *d*. In this approach therefore the square corresponding to each *st* grid-point is rendered using as a texture map the image corresponding to that grid-point. This involves finding the set of texture coordinates for each square, but since neighbouring squares will share grid points the number of projections required is approximately the number of squares drawn. Of course not all M^2 squares would be rendered, depending on the view volume of the camera.

FIGURE 21. Using Texture Mapping



In practice this simplistic technique will lead to significant aliasing. An alternative scheme is to use quadrilinear interpolation. There are four nearest neighbours to the intersection point on the *st* plane, and four nearest neighbours on the *uv* plane, as shown in Figure 22. Consider first interpolation on s only, with all other parameters fixed. (EQ 76) shows an identity expression s as a barycentric combination of s_0 and s_1.

FIGURE 22. Quadrilinear Interpolation



$$s = \left(\frac{s_1 - s}{s_1 - s_0}\right)s_0 + \left(\frac{s - s_0}{s_1 - s_0}\right)s_1$$
$$\text{or} \qquad \textbf{(EQ 76)}$$
$$s = \alpha_0 s_0 + \alpha_1 s_1$$
$$\text{where } \alpha_0 + \alpha_1 = 1$$

Quadrilinear interpolation makes the assumption that the *L* function is itself affine (of course it will not be so). Hence applying this to (EQ 76) we obtain:

$$\begin{aligned} L(s, t_0, u_0, v_0) &= \left(\frac{s_1 - s}{s_1 - s_0}\right)L(s_0, t_0, u_0, v_0) + \left(\frac{s - s_0}{s_1 - s_0}\right)L(s_1, t_0, u_0, v_0) \\ &= \alpha_0 L(s_0, t_0, u_0, v_0) + \alpha_1 L(s_1, t_0, u_0, v_0) \end{aligned} \qquad \textbf{(EQ 77)}$$

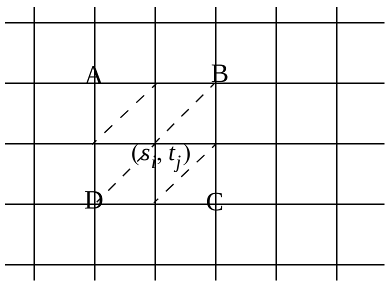
Repeating this argument for each of the parameters we obtain:

$$L(s, t, u, v) = \sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1}\sum_{l=0}^{1} \alpha_i \beta_j \gamma_k \delta_l L(s_i, t_j, u_k, v_l) \qquad \textbf{(EQ 78)}$$

where each of the β_j, γ_k, δ_l are defined in the same way as α_i in (EQ 76), and correspond to *t*,*u*, and *v* respectively. Now of course all the interpolated *s*, *t*, *u*, *v* values can be easily found using a method similar to finding interpolated texture coordinates during the process of rendering the *st* squares and *uv* squares as polygons.

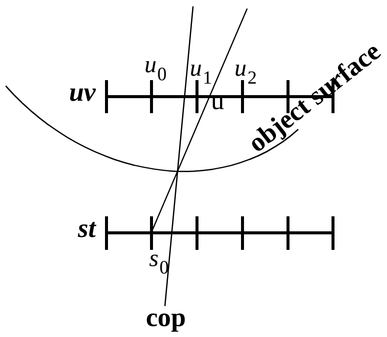
Gortler et al., 1996 show that this method cannot exactly be implemented using the texture mapping hardware. The support for the quadrilinear interpolation at (s_i, t_j) extends out to ABCD in Figure 23, and hence neighbouring grid-points will have overlapping supports. Consider the six triangles surrounding the grid-point. Gortler shows that if each of these 6 triangles is rendered with texture mapping as before plus alpha blending, where $\alpha = 1$ at the grid point and $\alpha = 0$ at the other triangle vertices, then this is equivalent to linear interpolation on *st* and bilinear interpolation on *uv*. It is suggested that results are not visually distinguishable from full bilinear interpolation.

FIGURE 23. **Support for the Quadrilinear Basis**



Gortler et al., 1996 show that keeping some geometric information together with the light field rays can lead to an improvement in radiance estimation. Figure 24 shows a viewing ray with origin at the COP intersecting an object surface. Now the nearest grid-point on the st plane is shown as s_0, and normally u_0 and u_1 would be used for interpolation. However, it is clear that more accurate information can be obtained from the ray s_0 through u, and therefore the better interpolation would be between u_1 and u_2. Gortler suggests storing a polygonal mesh approximating the scene geometry in order to allow for such depth corrections. This approach can result in less blurred images for the same *st* resolution, since a compensation is introduced for the errors involved in the interpolation. A further analysis concerned with speed-up to the rendering process, in particular tradeoffs between quality and time can be found in (Sloan, Cohen and Gortler, 1997).

FIGURE 24. **Depth Correction**



Representing Light Fields

The 2PP has obvious computational advantages, especially with respect to storage (rectangular images) and the use of rendering (especially texture) hardware for reconstruction. However, the representation does not provide a uniform distribution of rays in 4-space - in other words, if we think of any arbitrary ray in space, not all rays are equally likely to have the same degree of approximation of representation in the light field space. In practice, what this means is that the quality of the rendered image will be a function of the viewpoint and viewing direction. Viewpoints near the centre of the *st* plane, and directions orthogonal to this plane will result in better images than viewpoints off to one side with oblique viewing directions - simply because the number and distribution of rays will be different at such positions. A thorough analysis of the 2PP as a 4D space can be found in (Gu and Cohen, 1997).

Camahort and Fussel, 1999 have carried out a theoretical study of three alternative light field representations - the 2PP as above, a two-sphere representation (2SP), and a direction and point representation (DPP). These were considered by (Camahort, Lerios, and Fussell, 1998). The 2SP involves placing a bounding sphere around the scene, and partitioning the sphere into a uniform grid. Then all connections between all vertices on the sphere form the parameterisation. The DPP similarly involves placing a sphere around the scene. Now choose a point uniformly at random on the surface of the sphere. This defines a vector from the origin through that point. Consider the plane orthogonal to that vector and through the centre of the sphere, and bounded by the sphere (i.e., a disc at the origin of the sphere). Now choose a uniformly distributed set of points on the disc, as the origin of rays in the same direction as the original vector. The collection of all such rays will form a uniformly distributed set of rays in ray space. Each ray is represented by its direction, and its intersection on the disc through the origin. A different two sphere parameterisation was introduced in (Ihm, Park and Lee, 1997) which puts a bounding sphere around the scene, and then a set of small spheres on the surface of the bounding spheres to represent directions. They also introduce an efficient wavelet compression scheme for this data structure.

Camahort and Fussel, 1999 provide an in-depth analysis of these different schemes (as well as a thorough discussion of uniformity), and show that the direction and point approach results in less rendering bias, and corrections that are simple to apply. A ‘rectangularisation’ of this type of approach is adopted for virtual light fields in the next chapter.

Practical Issues

There are a number of practical issues to consider. First, what is the ideal placement of the *st* and *uv* planes? Clearly *st* is outside of the scene, whereas *uv* should be through the centre of the scene. Ideally the geometry of the scene should be as close as possible to the *uv* plane, so that the grid points are (very rough!) approximations to surface geometry. In practice this means that light fields cannot adequately represent scenes of significant depth. A distribution of rays which is adequate for the representation of near surfaces will not be sufficient for the representation of far away surfaces.

What resolutions should be used for *M* and *N*? The latter is more important since, as suggested, it determines the degree of approximation of scene geometry. The higher the resolution on the *uv* plane the greater the accuracy of this representation. Hence in practice $N > M$, and values of *N*=256 and *M*=32 have been shown to be sufficient for images of size 256×256.

Assume that this resolution is used, then one light slab will require 2^{26} rays. Suppose each ray carries 3Bytes, then 192MB is needed, and therefore over 1GB for a full light field (6 light slabs). Remember that this will only enable static scene walkthrough and relatively low resolution images. Clearly a compression scheme is needed! Levoy and Hanrahan, 1996 used vector quantization (Gersho and Gray, 1992). This is a compression scheme, where the data set is partitioned into clusters, each being represented by one vector found during a training phase. Each vector corresponds to a codebook which stores indices of the members of the cluster which it represents. Decoding is a very fast operation - especially useful in the context of rendering. Levoy and Hanrahan demonstrated compression ratios of more than 100:1 using this scheme.

Further Developments

The original light field / lumigraph papers were published in 1996. Since then a great deal of interest has been generated, and we briefly summarise some of the significant developments.

As mentioned earlier, one of the drawbacks of the light field scheme is that it is most appropriate for scenes without significant depth. (Isaksen, McMillan, and Gortler, 2000) discuss how to overcome this problem, in addition to supporting depth-of-field effects and showing how light fields can be used for the basis of auto-stereoscopic display systems. This latter development has also taken up in detail in (Perlin, Paxia and Kollin, 2000).

The light field approach is clearly for static scene walkthrough rather than interaction with objects in the scene. Nevertheless (Seitz and Kutulakos, 1998) allow some degree of interaction by showing how it is possible to edit one image of a scene and propagate the changes to all other images, maintaining overall consistency amongst all the images.

A surface light field is a light field consisting of rays emanating from surfaces - in principle for any point on a surface the set of all rays leaving that point with their associated radiances. Wood, et al., 2000 show how to construct, edit and display such light fields for real scenes. An important general contribution is to show how such compressed light fields may be directly rendered from the compressed representation.

The theory of light field representations was considered in (Camahort and Fussel, 1999) as discussed earlier. Chai, Tong, Chan, and Shum, 2000 provide a thorough analysis of sampling requirements. This includes a minimum sampling curve which provides the relationship between scene complexity, output resolution, and the image sampling.

Summary

This chapter has introduced the idea of light fields, as an image based approach to providing a ‘solution’ to the radiance equation. The LF approach is almost ‘brute force’ for this purpose - it works with a discrete representation of all possible rays covering a scene, and then assigns radiance to those rays by image rendering. Based on a finite sample of images, which have been used to colour rays, it was shown how images from new viewpoints can be constructed. There was some discussion of efficient rendering using the texture mapping hardware in the context of interpolation schemes. Alternative schemes for representing a uniform set of rays were considered, followed by a brief discussion of recent developments.

In this chapter we have concentrated on the basic ideas of LFs, without considering the source of the images from which they are constructed. The most widespread use of light fields is for virtual walkthroughs of real scenes. In the next chapter we consider the issues involved in using light fields for virtual scenes, and consider one approach, still in its early stages, in more detail.