Continuous LODs and Adaptive Frame-Rate Control for Spherical Light Fields

Alejandro Domingo, Miguel Escrivá, Francisco Abad, Javier Lluch, Emilio Camahort
{adomingo, mescriva, fjabad, jlluch, camahort}@dsic.upv.es
Computer Graphics Section
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia, Spain

Abstract

Light fields are an image-based representation that represents objects using sets of digital images. Light fields are usually comprised of 4D radiance data that can be used for 3D rendering or 4D display on autostereoscopic and certain types of volumetric displays. We present a modeling and rendering system for spherical isotropic light fields. These are made of images captured with cameras placed on a sphere's surface looking inwards. The system implements a multiresolution representation for both the spatial and the directional domains of the light field. This representation supports continuous levels of detail and adaptive frame rate control.

Keywords— **Light Field Modeling and Rendering, Multiresolution, Continuous Levels of Detail, Adaptive Frame Rate Control**

1 Introduction

3D spatial and autostereoscopic displays provide multiple viewers with a 3D image of the object of interest [9, 10, 12, 13, 14]. These devices require a source of 3D information, that usually comprises huge amounts of data. 4D light fields are an efficient technique to store and query 3D information and can be used to provide those devices with arbitrary views of an object.

Light fields are a Computer Graphics modeling technique that relies on sets of images to represent highly complex geometric objects. Storing and rendering these objects using regular geometric techniques is not efficient. Instead collections of images in different arrangements are used to model them.

High-resolution light-field models may contain thousands of images. Efficient algorithms for storing and retrieving data and rendering have to be carefully designed. On the other hand, they can obtain very realistic images and are the representation used by autostereoscopic and certain volumetric devices.

Current light-field models use different arrangements. Planar light-fields select a planar grid of camera positions and take an image for each camera. They require multiple instances of those planar configurations, called *slabs* to represent entire objects [6]. Spherical light-fields select camera positions on a sphere's surface pointing to the center of the sphere.

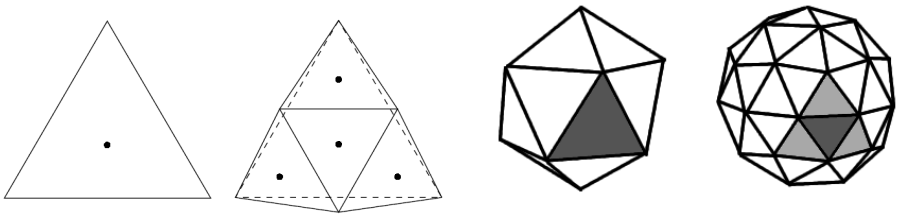
We are concerned with spherical light-field representations. This representation allows us to render models at interactive rates. This interactivity is supported by the multiresolution properties of the data structure, the possibility of using continuous levels of detail (LODs) and the adaptive frame-rate control. Our models can be efficiently displayed using a two-level cache with compression and lazy image retrieval.

In the following section we survey light fields and autostereoscopic and volumetric displays. Then, we introduce our spherical multiresolution representation and describe how to build and render it. In Section 4 we show how the rendering algorithm can be used with the multilevel cache to achieve adaptive frame-rate control. The next section is devoted to our implementation and results. Finally, we present some conclusions and directions for future work.

2 Background

Formally, a light field represents the radiance flowing through all the points in a scene in all possible directions [5, 6]. For a given wavelength, a static light field can be represented as a 5D scalar function $L(x, y, z, \theta, \phi)$ that gives radiance as a function of location (x, y, z) in 3D space and the direction (θ, ϕ) the light is traveling. The light-field function can be simplified to consider only the values it takes in *free space*, thus reducing the 5D domain of the light-field to 4D.

Light-fields are a class of image-based models. They have three main advantages: (i) their rendering complexity depends only on the complexity of the output images, (ii)



(a) Subdivision of a triangle.

(b) The first and second LODs.

Figure 1: Subdivision of directional space: each triangle represents a pencil of directions.

compression and simplification algorithms perform better on image data than in geometric data, and (iii) they can be combined with geometric models to produce hybrid representations.

2.1 Light-Field Models

Planar light fields were first introduced by Levoy and Hanrahan [6]. A similar approach with geometric proxies was proposed by Gortler *et al*. [5]. Both representations are anisotropic because they require multiple slabs in order to represent an object and multi-slab rendering produces artifacts when crossing slab boundaries [8].

Typical arrangements of planar light fields contain 32×32 or 64×64 cameras arranged in a regular planar grid. Each image typically contains 256×256 or 512×512 pixels each. To reduce the number of images to retrieve and render, Sloan et al. triangulate the camera plane [7]. They remove all the cameras that fall within a triangle. Then, they use the cameras located at the triangles' vertices to render their images using image warping.

Spherical light fields were proposed by Camahort *et al*. [8]. They are isotropic because they sample the light field of an object in a (nearly) uniform fashion. This reduces the number of artifacts when rendering the object's light field from different viewpoints. We have implemented a light-field modeling and rendering system based on spherical isotropic light fields [16]. The system can be extended to support modeling of multiple objects and geometric information [18].

2.2 Autostereoscopic and Volumetric Displays

3D spatial and autostereoscopic displays are becoming the subject of recent research efforts in the Computer Graphics hardware community [10, 13]. They provide one or more viewers with a 3D image of an object without the need of goggles or other invasive devices.

In practice autostereoscopic devices display a 4D version of the light-field function. This version is typically based on the two-plane parameterization (2PP) that was originally inspired by holography and, specifically, by holographic stereograms [2, 3]. This parameterization simplifies rendering by avoiding the use of cylindrical and spherical projections during the light-field reconstruction process.

Autostereoscopic devices typically display planar light fields [9, 14]. Volumetric displays are also becoming common in Computer Graphics applications [1, 4]. They are able to display volumetric data, 3D geometric data or spherical light fields [11, 15, 17].

3 Multiresolution Spherical Light Fields

A static 4D spherical light field is a scalar function that returns radiance along the lines intersecting the unit sphere [8]. The object of interest is located inside the sphere. In our representation each line is parameterized by its direction and its intersection point with a 2D plane orthogonal to the line direction. To obtain a discretization of this line domain we discretize the 2D set of all 3D directions. Then, we discretize each plane by superimposing a regular grid to obtain the light-field images. These images are parallel projections along the discrete directions.

3.1 Multiresolution Representation

A multiresolution representation is built as follows. We start approximating the sphere using an icosahedron and choosing as initial directions those passing through the centers of the icosahedron's faces. Then, we subdivide each triangle into four sub-triangles as shown in Figure 1(a). The first two resolution levels are shown in Figure 1(b). They have 20 and 80 triangles respectively, corresponding to 20 and 80 directional samples. Each sample approximates the pencil of directions passing through its associated triangle and the center of the sphere. If we further subdivide we obtain LODs with $20 * 4^{level}$ pencils where $level \geq 0$. For the positional samples we use standard mipmaps of the images taken along each direction (see Figure 2).

The DPP rendering algorithm is an adapted version of the Lumigraph algorithm [5, 16]. Rendering using a spherical light field is done by placing the camera frustum inside of the tessellated sphere. The rendering algorithm de-

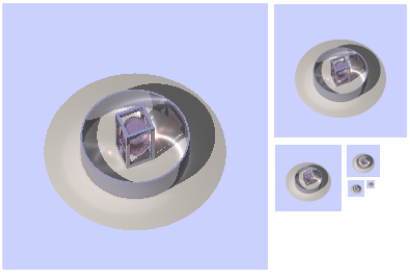


Figure 2: Mipmap pyramid for one of the directional samples of the dataset *phot*

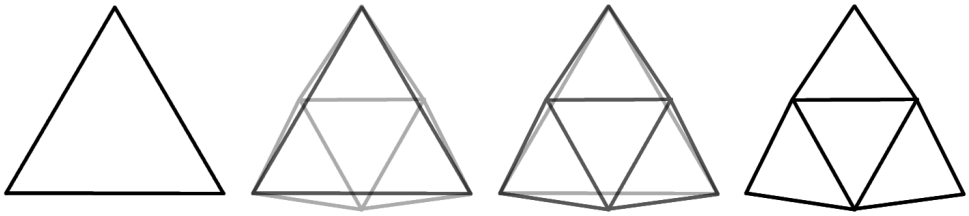


Figure 3: A triangle and its children with LODs $n, n + 1/3, n + 2/3$, and $n + 1$. Gray levels denote the $alpha$ weights.

termines which pencils of directions intersect the viewing frustum. Triangles falling within the frustum are texture mapped with the images stored in the light field. A specific texture matrix is computed for each triangle depending on the relative position of its image with respect to the viewing parameters. In our system, the rendering hardware is in charge of the reprojection and display tasks. Each visible triangle T_k is rendered with a texture map containing a portion of the texture. That portion is determined by the geometric relationship between the viewer's position and the light-field model as given by its center and orientation. We determine the correct texture coordinates by casting three rays from the viewpoint, one through each of the vertices of T_k.

3.2 Continuous Levels of Detail

The main problem of discrete LOD techniques is the *popping* effect. This effect appears as artifacts in the image when an object suddenly changes its LOD. Our system creates smooth transitions between LODs in both the positional and the directional domains. For the positional domain we interpolate between consecutive mipmap levels. For the directional domain we do a linear interpolation between the triangle associated to a pencil and the triangles associated to its children. We α-blend the parent with a value $t \in [0, 1]$, that determines the weight associated to the parent's LOD. Likewise, we α-blend the childrens using an α value of $1 - t$. This is illustrated in Figure 3.

4 Adaptive Frame Rate Control

Our multiresolution representation supports adaptive frame rate control. The number of pencil triangles to render can be decided on-line by choosing which LOD to display. The rendering time depends on the number of triangles inside the rendering window. On the other hand, the number of triangles is usually small. The real bottleneck of the algorithm is the image caching process. Images need to be loaded into the GPU to texture-map the triangles. So, first we will explain how the images are cached and then how to compute the number of pencil triangles to render.

4.1 Light-Field Image Caching

Light-field images are stored in a three-level cache hierarchy. As the light-field models can require up to several gigabytes of memory, they must be stored in a hard drive. Images that are not stored in the cache, but are needed to render the pencils inside the frustum are loaded on demand into main memory, and then into the GPU.

Our caching algorithm uses out-of-core techniques. These techniques support lazy image caching. Upon initialization all 20 images of the first LOD are loaded into the GPU memory. Then images are loaded from main memory or retrieved from disk and then loaded into the GPU for texture mapping the pencil triangles. When an image needed to render a given triangle is not available in memory, its parent image can be rendered instead. Since the first LOD of images is always stored in the GPU, there is always a parent image ready to use. Later, as images associated to children triangles are available, the display can be updated. The frames rendered using parent images are usually less

(a) 3

(b) $3\frac{1}{3}$

(c) $3\frac{2}{3}$

(d) 4

Figure 4: Test scene with LODs $3, 3\frac{1}{3}, 3\frac{2}{3}, 4$

detailed than they should, but their quality improves as the children images are loaded into the GPU.

4.2 Selecting the Number of Pencil Triangles

To achieve adaptive frame-rate control the maximum number of triangles to render in each frame has to be computed. This is done as follows. The solid angle of a square viewing window with a given fov is $\Omega(\text{fov}) = 2\text{fov}\sin(\text{fov}/2)$. The solid angle subtended by each triangle is the solid angle of the sphere (4π) divided by the total number of triangles at the given LOD. The number of triangles projected by a window with fov field of view is:

$$p = D * \text{fov}/(2\pi) * \sin(\text{fov}/2)$$

where the number of pencils in the current LOD $level$ is given by $D = 20 * 4^{level}$ For non-integer LODs D is four times larger than the lowest integer LOD's, namely $D' = 1.25 * 20 * 4^{\lceil level \rceil}$ Still, only three times more images are needed because center sub-triangles use the same

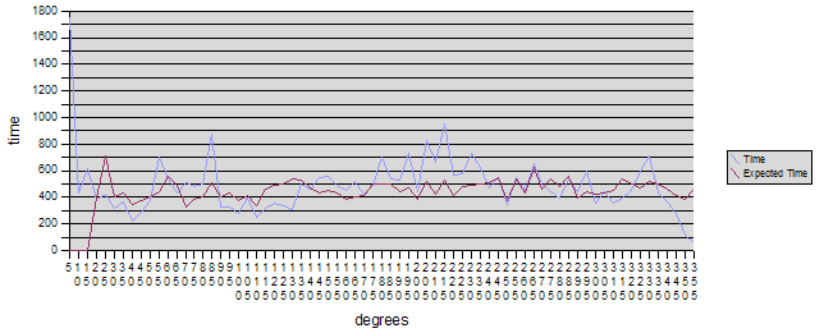


Figure 5: Rendering time (in ms) and expected rendering time using level 4

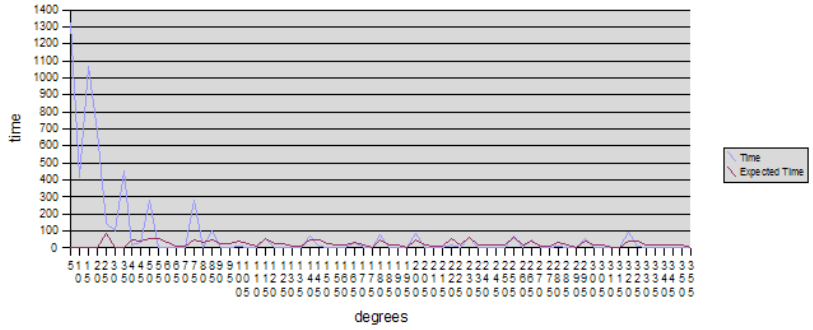


Figure 6: Rendering time with adaptive frame control starting with level 4, limiting the frame rendering time to 100ms

image as their parent.

A better approximation can be obtained by simulating the rendering of each level of the geode and computing the number of images stored in the cache for each level. This is more precise but requires preprocessing the data before rendering, and it can not predict how the user will interact with the system.

We computed the rendering times for each LOD in several experiments and we obtained an approximation of the coefficients of the following linear function:

$$t = t_{main}N_{main} + t_{gpu}N_{gpu} + t_{disk}N_{disk} + t_{setup}$$

where t_{main}, t_{gpu} and t_{disk} are the times it takes to render a triangle depending on where its associated texture image is located (main memory, video memory or disk). N_{main}, N_{gpu} and N_{disk} are the number of triangles rendered in each situation. t_{setup} is the constant time it takes to setup the rendering of any frame.

When running the simulation, the first several frames do not provide enough samples to obtain good estimates about the rendering times. Two methods can be used to solve this problem. In the initial moments of the rendering lower LODs can be used, incrementing them as the simulation proceeds. Or some initial estimates for the t_i can be obtained from the data and machine specifications (e.g.: the hard drive's latency and data transfer rates, and the memory to GPU transfer time). Memory to GPU time depends on polygon transfer rate and, most importantly, on the texture transfer rate.

When transfer rates are available, transfer times per triangle can be computed using the transfer rates and the size of the data to be transferred. For example, a single pencil triangle image typically takes 40 KBytes in disk, and 256 KBytes of GPU memory in mipmap format. The initial set of 20 images loaded for the lowest LOD takes 5 MBytes of GPU memory. Time estimates for our test machine are given in the next section.

5 Implementation and Results

We tested our system on a Pentium IV at 3Ghz with 1GB of main memory, equipped with an ATI Mobility Radeon X600 graphics card with 128MB of memory. Figure 5 shows the expected rendering time versus the measured rendering time, forcing the level of detail to 4. Note the accuracy of the estimated values. In the test the camera is moved along a circular path in steps of 5 degrees per frame and the expected time is calculated using linear regression. The measured times in our system are: setup rendering time is $t_{setup} = 272$ms which depends on the LOD and the processing and memory speed of the test computer,

disk access time is $t_{disk} = 19.1$ms and main memory to GPU transfer time is $t_{transfer} = 0.01$ms.

Figure 6 shows the rendering time and LOD using frame rate control.For each frame, the highest LOD that satisfies the frame rate required is selected.

Videos of our system can be found on the web page `http://www.sig.upv.es/ALF/papers/gmai2007/`

6 Conclusions and Future Work

We have presented a method to interpolate between different levels of detail in spherical light fields, and a method to compute the time that an LOD takes to render. From these methods we have built a rendering application that changes continuously the resolution of a light field to keep the frame rate at interactive rates.

We are planning on extending our light-field models with depth information. This will allow us to display multiple light fields and to combine our models with geometry to display annotated/augmented light fields on autostereoscopic and certain types of volumetric displays. We are also planning to add interpolation in the directional domain.

Acknowledgements

This work was partially supported by grants TIN2004-203303-E and TIN2005-08863-C03-01 of the Spanish Ministry of Education and Science and by the GAMETOOLS STREP project IST-004363 of the 6th Framework Program of the European Union.

References

[1] Parker, E., and Wallis, P A. Three-dimensional cathode-ray tube displays, J. IEEE, 1948, 95, pp.371–390

[2] S. A. Benton. Survey of holographic stereograms. In Processing and Display of Three-Dimensional Data, volume 367, pages 15–19, 1982.

[3] K. Haines and D. Haines. Computer graphics for holography. IEEE Computer Graphics and Applications, pages 37–46, January 1992.

[4] Blundell, B.G.; Schwarz, A.J.; Horrell, D.K., Volumetric three-dimensional display systems: their past, present and future. Engineering Science and Education Journal , vol.2, no.5pp.196–200, Oct 1993

[5] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski and Michael F. Cohen. The Lumigraph. In Proc. SIGGRAPH '96

[6] Marc Levoy and Pat Hanrahan. Light Field Rendering. In Proc. SIGGRAPH '96

[7] Peter-Pike Sloan, Michael F. Cohen and Steven J. Gortler. Time Critical Lumigraph Rendering. In Proc. 1997 Symposium on Interactive 3D Graphics

[8] Camahort E., Lerios A., Fussel D. Uniformly Sampled Light Fields. In Proc. Eurographics Rendering Workshop '98 (1998), pp. 117–130

[9] A. Isaksen, L. McMillan, and S. J. Gortler. Dynamically reparameterized light fields. In Proc. SIGGRAPH '00, pages 297–306, 2000.

[10] K. Perlin, S. Paxia, and J. S. Kollin. An autostereoscopic display. In Proc. SIGGRAPH '00, pages 319–326, 2000.

[11] Favalora, G. E., Napoli, J., Hall, D. M., Dorval, R. K., Giovinco, M. G., Richmond, M. J., Chun, W. S. 100 million-voxel volumetric display. Proceedings of the SPIE, vol 4712, 300–312, (2002)

[12] Sullivan, Allan. 3-Deep. IEEE Spectrum, pages 2227, April 2005.

[13] T. Balogh, T. Forgcs, O. Balet, E. Bouvier, F. Bettio, E. Gobbetti, and G. Zanetti. A scalable holographic display for interactive graphics applications. IEEE VR 2005 Workshop on Emerging Display Technologies, March 2005.

[14] R. Yang, S. Chen, X. Huang, S. Li, L. Wang, and C. Jaynes. Towards the light field display. IEEE VR 2005 Workshop on Emerging Display Technologies, March 2005.

[15] Won-Suk Chun, Joshua Napoli, *et al.* Spatial 3-D Infrastructure: Display-Independent Software Framework, High-Speed Rendering Electronics, and Several New Displays. In Proceedings of SPIE-IS&T Electronic Imaging, SPIE Vol. 5664, pp. 302-312 (2005).

[16] Escrivá M., Domingo A., Abad F., Vivó R., Camahort E. Modeling and Rendering of DPP-Based Light Fields. In Procededings of GMAI'2006 (London, UK, July 2006), IEEE Computer Society.

[17] Takafumi Koike, Michio Oikawa, Nobutaka Kimura, Fumiko Beniyama, Toshio Moriya, and Masami Yamasaki, Integral videography of high-density light field with spherical layout camera array. In Proceedings of SPIE Int. Soc. Opt. Eng. 6055, 605510 (2006)

[18] A. Domingo, M. Escrivá, F.J.Abad, R.Vivó, E.Camahort, Introducing Extended and Augmented Light Fields for Autostereoscopic Displays, 3rd Ibero-American Symposium in Computer Graphics, SIACG 2006, Short Papers, pp. 64-67, July 2006