Light Field Rendering using Matrix Optics

Lukas Ahrenberg
MPI Informatik
Stuhlsatzenhausweg 85
D-66123 Saarbrücken, Germany
ahrenberg@mpi-inf.mpg.de

Marcus Magnor
Technical University of Braunschweig
Muehlenpfordtstrasse 23
D-38106 Braunschweig, Germany
magnor@mpi-inf.mpg.de

ABSTRACT

This paper presents a light field rendering framework based on matrix optics. Matrix optics, in contrast to intersection-based methods such as ray-tracing, has the advantage that a generic series of optic operators can be combined into a single matrix. This enables us to realize a "virtual optical bench" where different setups can be easily tested. We introduce the theoretical foundation of matrix optics and define a set of operators suitable for light fields. We then discuss a wavelet compression scheme for our light field representation. Finally we introduce a real-time rendering approach based on matrix optics suitable for both uncompressed and compressed light fields.

Keywords
Light field, matrix optics, ray optics, rendering

1 INTRODUCTION

Since the introduction of image based light fields [GGSC96, LH96] a number of different rendering techniques have been suggested. The standard methods today include both image-space and object-space algorithms. The former is based on ray-tracing, ray-casting or view morphing, while the latter typically involves texture mapping. In this paper we investigate the use of matrix optics for light field rendering. In comparison to other light field rendering methods, no ray-plane intersection test has to be performed for every image pixel. Ray-tracing-based methods can be cumbersome if the imaging system involves a series of optical elements such as lenses or material interfaces causing refraction. In this case the imaging method requires tracing a ray to every element which transforms it in some way. Using matrix optics, a set of operators such as light propagation, thin lenses and dielectric interfaces can be represented using matrices, allowing us to model the whole process as a single matrix. This enables us to model a light field under the influence of an arbitrary series of optical operators by performing a linear transform of its elements. Rendering is a special case of this where a camera model is constructed from lens and propagation operators. Our main contribution in this paper is a matrix optics theory for light field modelling. In addition, we show how the operators can be used when rendering directly from a hierarchical wavelet representation of the light field. The implemented framework can be thought of as a "virtual optical bench", a test environment for optical manipulation of light fields.

The paper is organized as follows: Section 2 presents related work in light field research. Section 3 presents the theory of matrix optics, introduces operators for some optical elements and shows our image formation model. In section 4 we construct a framework for light fields using matrix optics. Results are discussed in 5 before we conclude our work in section 6.

Conference proceedings ISBN 80-86943-03-8
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.

2 RELATED WORK

Several different methods have been proposed for light field rendering in the past few years. Sloan and Hansen have developed a number of methods using ray-light slab intersection tests targeted at parallel architectures [SH99]. In [BBM+01] Buehler et al. present a general method for rendering by blending views from the original light field using a blending field derived from geometrical and view information. Vlasic et al. merge both view-dependent appearance and view-

dependent shape in what they call Opacity Light Fields [VPM+03]. Goldluecke et al. have developed a GPU-based method for dynamic light field rendering using a warping algorithm [GMW02]. The matrix optics method developed in this paper expresses the ray transport from the light field to the image plane as a linear transform. This allows us to easily model the light field under influence of optical elements such as lenses and interfaces.

In addition to a plain light field representation, our framework also has the possibility to handle wavelet compressed light fields. This allows for high compression ratios and efficient storage and rendering. The efficiency for light field encoding has already been reported in several publications; Lalonde and Fournier use wavelets to store light fields in a hierarchical data structure [LF99], while Peter and Straßer introduce a wavelet representation that allows for efficient storage and progressive transmission of light fields [PS01]. In [LSG02] a light field acquisition, compression and representation system based on a hierarchical wavelet structure is presented. Our approach to data representation is similar to the work of Peter and Straßer, while the basic wavelet tree accessing philosophy has ideas in common with the method of Lalonde and Fournier. However, in contrast to both approaches, we perform interactive image reconstruction using matrix optics.

3 MATRIX OPTICS

In this section we introduce a theoretical model for light fields based on matrix optics. We will only briefly touch the foundations of matrix optics here, a more general introduction can be found in [Fow75] and [GB94]. Matrix optics defines linear operators for a number of optical elements as well as propagation of light between planes in space. This gives an elegant way of computing propagation and optical manipulations of light fields. Standard ray-casting or ray-tracing based methods must compute the ray-path between the individual optical elements, while in matrix optics all operators can be combined using matrix multiplication. The model introduced here is an extension to the matrix operators in optics, suitable for light field transformations.

We will start by defining a ray space and a light field on this ray space. Then we will introduce a set of matrix optics operators for modelling optical phenomena on the light field. Finally we construct a camera model and show how an image is formed from the light field.

Let $\Pi \subset \mathbb{R}^3$ be a plane with an associated coordinate system in 3D space

$$\Pi = (\mathbf{o}_\Pi, \mathbf{n}_\Pi, \{\mathbf{e}_1^\Pi, \mathbf{e}_2^\Pi\}), \tag{1}$$

where $\mathbf{o}_\Pi$ and $\mathbf{n}_\Pi$ is the origin and normal of Π, and $\{\mathbf{e}_1^\Pi, \mathbf{e}_2^\Pi\}$ are the vectors spanning the plane.

The ray space on Π consists of all light rays intersecting the plane,

$$\mathcal{R}_\Pi = \mathbb{R}^2 \times \mathbb{R}^2. \tag{2}$$

Thus, a ray passing through Π is described as a point in ray space with homogeneous coordinate

$$\mathbf{r} = [\mathbf{x}, \mathbf{d}, 1]^t \in \mathcal{R}_\Pi. \tag{3}$$

Above $\mathbf{x} = [x_1, x_2]$ denote the plane coordinates in the frame $\{\mathbf{e}_1^\Pi, \mathbf{e}_2^\Pi\}$, and $\mathbf{d} = [d_1, d_2]$ the directional component along Π's basis vectors.

A light field on Π is a mapping

$$L : \mathcal{R}_\Pi \to \mathbb{R}. \tag{4}$$

Given a ray $\mathbf{r}$, the light field L yields the radiance along that ray. We can now define a set of matrix operators for transforming ray space.

3.1 Propagation Operators

A propagation $\mathbf{P} : \mathcal{R}_\Pi \to \mathcal{R}_{\Pi_p}$ means a transformation of the ray space $\mathcal{R}_\Pi$ of plane Π to the ray space $\mathcal{R}_{\Pi_p}$ of some other plane

$$\Pi_p = (\mathbf{o}_{\Pi_p}, \mathbf{n}_{\Pi_p}, \{\mathbf{e}_1^{\Pi_p}, \mathbf{e}_2^{\Pi_p}\}). \tag{5}$$

Any point in $\mathcal{R}_\Pi$ and its image in $\mathcal{R}_{\Pi_p}$ under the propagation $\mathbf{P}$ must correspond to the same ray in world space.

We assume that all propagation takes place in free space, i.e. that there are no occluding objects between Π and Π_p.

Mathematically, let $W_\Pi(\mathbf{r})$ be the world space ray of $\mathbf{r} \in \mathcal{R}_\Pi$, given by the base point $\mathbf{o}_\Pi + [\mathbf{e}_1^\Pi, \mathbf{e}_2^\Pi]\mathbf{x}$ and the direction $\mathbf{n}_\Pi + [\mathbf{e}_1^\Pi, \mathbf{e}_2^\Pi]\mathbf{d}$.

Then

$$\mathbf{P} : \mathcal{R}_\Pi \to \mathcal{R}_{\Pi_p} \tag{6}$$

is a propagation operator if and only if

$$\forall \mathbf{r} \in \mathcal{R}_\Pi : W_\Pi(\mathbf{r}) = W_{\Pi_p}(\mathbf{P}\mathbf{r}). \tag{7}$$

We will now introduce two different propagation operators: the transport operator, which propagates between parallel planes, and the rotation operator which propagates between rotated planes.

3.1.1 The Transport Operator

The transport operator, $T_\mathbf{v}$, propagates the light to a plane parallel to Π offset by some vector, $\mathbf{v} = [v_1, v_2, v_3]^t \in \mathbb{R}^3$

$$\Pi_\mathbf{v} = (\mathbf{o}_\Pi + \mathbf{v}, \mathbf{n}_\Pi, \{e_1^\Pi, e_2^\Pi\}). \tag{8}$$

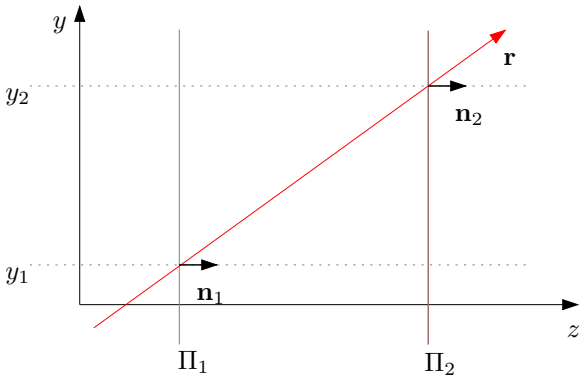

Figure 1: An example of ray propagation. At the plane Π_1, the ray $\mathbf{r}$ has coordinate $\mathbf{r}_{\Pi_1} = [y_1, d]^t$. d is the directional deviation of $\mathbf{r}$ from the normal $\mathbf{n}_1$, and can intuitively be thought of as the slope of r. For an empty space propagation to plane Π_2, the spatial y-component is updated by the transport along the ray direction, while the directional component is unaffected.

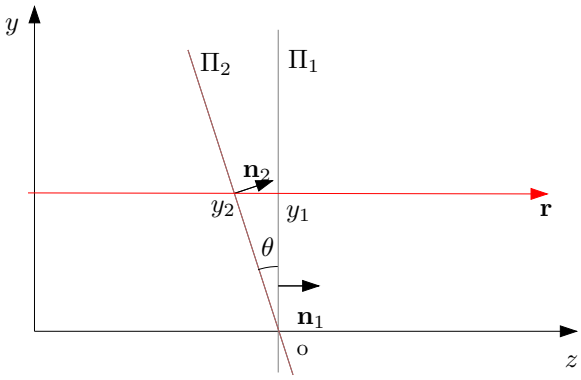

Figure 2: Π_2 is a plane rotated by θ around the origin of Π_1, and $\mathbf{r}$ is a ray traveling in the normal direction of Pi_1, intersecting the planes in y_1 and y_2 respectively. To find the ray coordinate in Π_2, observe that y_2 is scaled proportionally to y_1, given by the triangle $\overline{oy_1y_2}$. As $\mathbf{r}$ intersects Π_2 at an angle of $-\theta$, the direction will be offset by $-\tan\theta$.

It can be written as a 5×5 matrix

$$\mathbf{T_v} = \begin{bmatrix} 1 & 0 & v_3 & 0 & v_1 \\ 0 & 1 & 0 & v_3 & v_2 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{9}$$

which has the desired properties. Intuitively, this corresponds to a transportation along the ray direction and a translation in the plane. Figure 1 shows an example of a pure transportation.

3.1.2 The Rotation Operator

The rotation operator maps the ray space of a plane Π to a ray space of a rotated plane $\Pi_{\mathbf{S}_j^\theta}$, where $\mathbf{S}_j^\theta$ denotes rotation of θ around basis vector $\mathbf{e}_j^\Pi, j \in [1, 2]$:

$$\Pi_{\mathbf{S}_j^\theta} = (\mathbf{o}_\Pi, \mathbf{S}_j^\theta \mathbf{n}_\Pi, \{\mathbf{S}_j^\theta \mathbf{e}_1^\Pi, \mathbf{S}_j^\theta \mathbf{e}_2^\Pi\}). \tag{10}$$

The matrix for ray-space transformation corresponding to the plane rotation of Π around $\mathbf{e}_1^\Pi$ is

$$\mathbf{R}_1^\theta = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1/\cos\theta & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -\tan\theta \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{11}$$

Rotation around $\mathbf{e}_2^\Pi$ follows by symmetry.
A general rotation does not yield a linear operation in ray space. However, it can be adequately approximated if the *paraxial approximation* of ray optics is applied. A linear approximation can be used for rays which lie close to the optical axis. We will use this approximation both for rotations and for elements with curved surfaces in the next section. Figure 2 shows an example where the plane has been rotated around the origin of the plane p_1.

3.2 Lens and Interface Operators

In this section we present operators which map the ray space on Π onto itself.

$$\mathbf{I} : \mathcal{R}_\Pi \to \mathcal{R}_\Pi. \tag{12}$$

This kind of operator changes the ray direction, and can be used to model elements such as interfaces and thin lenses.

3.2.1 Interfaces

Interfaces are used to model light transition from one medium to another. If the materials have different refraction indices, a perturbation of the ray direction will occur when passing through the material boundary.
The matrix for a planar interface is

$$\mathbf{P}_{n_1,n_2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{n_1}{n_2} & 0 & 0 \\ 0 & 0 & 0 & \frac{n_1}{n_2} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

where n_1 and n_2 are the refractive indices of the source and destination materials.
For a circularly curved interface the perturbation of the directional component depends on the plane coordinate component of the ray. The operator matrix is

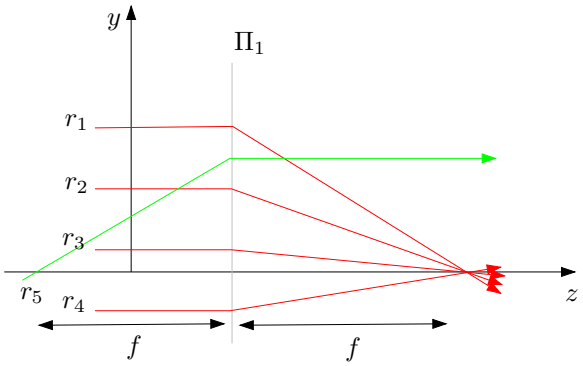

Figure 3: A thin lens in plane Π_1. The rays r_1 to r_4 arrive perpendicular to p_1. Their directions are perturbed depending on the distance from the origin of Π_1 so that all intersect at a distance of f in front of Π_1. r_5 on the other hand, intersects the z-axis a distance of f *in front* of Π_1 and will emanate normal to the plane. A general ray will have its directional component offset by $-y_1/f$.

written as

$$\mathbf{C}_{n_1,n_2,r} = \begin{bmatrix} 1 & 0 & 0 & 0 & \\ 0 & 1 & 0 & 0 & \\ \frac{1}{r}(\frac{n_1}{n_2}-1) & 0 & \frac{n_1}{n_2} & 0 & \\ 0 & \frac{1}{r}(\frac{n_1}{n_2}-1) & 0 & \frac{n_1}{n_2} & \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{14}$$

As with the planar interface n_1, n_2 denotes the refractive indices, while r is the radius of curvature. A positive r yields a convex interface, while negative values results in a concave one.

3.2.2 The Thin Lens Operator

A lens is considered "thin" if the light propagation within the lens material can be neglected. Thus, the thin lens acts as a perturbator of ray directions, and has the matrix

$$\mathbf{H}_f = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ -1/f & 0 & 1 & 0 & 0 \\ 0 & -1/f & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

for a focal length of f.
The matrix varies the directional component of a light field coordinate as a linear function of the plane component. Figure 3 depicts an intuitive example of a thin lens.

3.3 Image Formation

We will now show how a two dimensional image can be formed from a 4D light field.

Let

$$\Gamma = (\mathbf{o}_{\mathbb{R}^3}, \mathbf{e}_3, \{\mathbf{e}_1, \mathbf{e}_2\}) \tag{16}$$

be the image plane located at the world space origin, where $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ is the standard basis in $\mathbb{R}^3$. To form an image on Γ the intensity of the image plane at a point $\mathbf{x}$ is computed using the general camera model

$$I_\Gamma(\mathbf{x}) = \int_{A_\mathbf{x}} \omega(\mathbf{d}) L(\mathbf{M}[\mathbf{x}, \mathbf{d}, 1]^t) d\mathbf{d}. \tag{17}$$

L is a light field defined on $\mathcal{R}_\pi$. $A_\mathbf{x}$ is the set of all ray directions intersecting $\mathbf{x}$ through the camera aperture, $\mathbf{M}$ is the matrix transforming from $\mathcal{R}_\Gamma$ to $\mathcal{R}_\Pi$ via any optical elements present, and ω is a weighting function used to grade rays dependent on their direction.
However general, this model is computationally expensive. A common practice in real-time computer graphics rendering is to use the pinhole camera model. This is a special case of the general model where the aperture of the camera is considered a single point in space, yielding only a single ray per point in the image plane. If we let the weight $\omega = \delta_0$, so that only rays with $\mathbf{d} = \mathbf{0}$, i.e. perpendicular to the image plane, are considered the camera integral reduces to

$$I_\Gamma(\mathbf{x}) = L(\mathbf{M}[\mathbf{x}, \mathbf{0}, 1]^t). \tag{18}$$

For $\mathbf{M} = \mathbf{1}$ or $\mathbf{M} = \mathbf{T}_\mathbf{v}$ this will render an orthographic view of the light field. Perspective views can simply be rendered by including a lens matrix in $\mathbf{M}$. Thus, a perspective camera with focal length f, viewing a light field from a distance of t, would have the matrix

$$\mathbf{M} = \mathbf{T}_\mathbf{v}\mathbf{H}_f \tag{19}$$

where $\mathbf{v} = [0, 0, t]$.

4 IMPLEMENTATION

In the previous section we have shown how matrix operators can be used to transform light fields. In this section we will discuss our implemented framework.

4.1 Light Field Representation

We have chosen to implement two different ways of representing the light field. The first is a raw light field table, which we will call the *direct* representation, the second is a *wavelet compressed* representation suitable for huge light fields. The former is faster as it basically is a four-dimensional image representation. For each spatial coordinate x_1, x_2, we can look up the RGB value of any ray of direction d_1, d_2. However, as the data size of light fields often gets large, we have also implemented the alternative to use a wavelet compressed representation as described below.

4.1.1 A Wavelet-based Representation

For huge light fields it may be necessary to sacrifice speed and compress the data. We have implemented a wavelet compression scheme where the light field is wavelet-compressed and the coefficients are stored in a space-partioning hexadecary tree structure. This structure is similar to the one presented by similar to the ones presented by Peter and Straßer [PS01]. The data can easily be sent as a progressive stream, making it an attractive choice for data transmission.

Wavelet compression is a well-known approach to data reduction. In the discussion below, we assume the reader to have a basic knowledge of wavelet theory and point to other sources, such as [Dau92] and [SDS96], for a thorough introduction.

Let L be a light field on $\mathcal{R}_\Pi$, and B_i a set of wavelet basis functions. L can then be written as a linear combination of the basis functions

$$L = \sum_{i=0}^{N-1} c_i B_i \tag{20}$$

where N is the total sampled resolution of L, i.e. the number of wavelet basis functions. c_i is called the wavelet coefficients, and for basis functions with good interpolating properties many of these can be dropped or quantized without introducing major visual errors. Thus, lossy compression can be achieved by only storing the important coefficients.

In our wavelet representation of light fields we have made use of two important properties of the basis functions: that they have local support, and space subdivision. A wavelet basis function has local support and is used to refine the value of a basis function on a coarser scale. That is, for a basis function, B_k, of support $s > 0$, one can find basis functions, B_j, $j < k$, of a coarser scale $t > s$, so that their support contains the support of B_k. The support of the basis functions divide the original support of the data set into sub-sets. The basis functions form a space partitioning tree analogous to binary-, quad- and octrees in 1D, 2D and 3D. In analogy we will call this a *hexadeca-tree*, as the children subdivide a parent's support into 16 regions. Using this tree we let each node represent all basis functions of a specific scale and translation. The child-nodes will be those basis functions refining the value along their parent-node's support, subdividing them. In the 1D case of the situation, a binary tree is formed; each node contains only one basis function and the corresponding coefficient, but higher-dimensional trees will have more functions of the same support which are stored in the same tree node. In our 4D case, there will be 16 different basis functions defined having the same support. These will be represented in the same node.

To keep the memory size at a minimum, the node structure is dynamic, storing only the non-zero coefficients and the existing children. We can reduce the memory usage of the hexadeca-tree even more by pruning the tree in a bottom-up approach after compressing the coefficients. As no leaf-node with zero-coefficients will contribute to the reconstructed signal, it can be removed. If all children of a node are removed, it becomes a leaf and the same test can be applied again until we find a node that can not be removed.

The nodes are stored breadth-first in an array. This facilitates progressive decoding so that time-, transmission- or memory-critical applications need only read and decode a part of the tree to obtain approximate rendering results. The approach is similar to the spatial orientation trees in the SPIHT codec for images [SP96].

4.2 Rendering

As seen in Section 3.3, Eq. (18) can be used to render an image from a light field, L. Our framework implements this equation, and computes the image formation matrix $\mathbf{M}$ by having the user specifying a chain of optical elements.

For the direct light field representation, a value lookup is straightforward, and Eq. (18) can be implemented directly. However for the wavelet-compressed representation, the light field must be reconstructed before it can be evaluated. Instead of reconstructing the full light field we have developed a an access method that takes the hierarchical structure of the wavelet tree into consideration. This method is similar to the the work of Lalonde and Fournier [LF99], but integrates it with our operator framework.

Given some image formation matrix $\mathbf{M}$, and an image plane coordinate $\mathbf{u} \in \Gamma$, let $\mathbf{v} = \mathbf{Mu}$. Observe that if $\mathbf{v}$ is located in the support of a specific node in the hexadeca-tree it is bound to be located in the support of one of that nodes children. As the wavelet functions have a value of 0 outside their support, the only nodes that will affect $L(\mathbf{v})$ are the single parent-child chain of nodes whose support contains $\mathbf{v}$. Thus,

$$L(\mathbf{v}) = \sum_{i \in \Omega} c_i B_i(\mathbf{v}), \tag{21}$$

where Ω is the set of all indices of basis functions B with support containing $\mathbf{v}$.

From (18) and (21) we then have the reconstruction expression

$$I_\Gamma(\mathbf{x}) = \sum_{i \in \Omega} c_i B_i(\mathbf{M}[\mathbf{x}, \mathbf{0}, 1]^t). \tag{22}$$

As the children of a node sub-divide the support, it is simple to compute Ω from the root node. Given that a

node contains **v** it is only necessary to check in which of the node's children the point lies until a leaf node is reached. The reconstruction sum will thus take the form of a traversal through the space partitioning tree. This is depicted in Fig. 4.

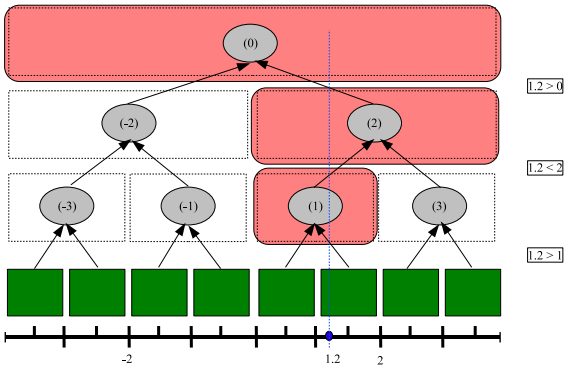

Figure 4: Traversal of a basis node tree, to reconstruct the value at position 1.2. The method starts at the root node, and checks in which child the coordinate lies. Nodes visited in the traversal are highlighted.

We know that the support of the next basis functions in the sum will be contained within the support of the current. The number of summations needed to reconstruct a value in a direct approach will thus be the depth of the tree, which is logarithmically dependent on the resolution of the data. This straightforward method can be directly implemented as an image-space algorithm.

4.3 Antialiasing

A common problem when reconstructing novel views from a light field is aliasing artifacts due to undersampling. This is usually resolved with some kind of interpolation. However, given the 4 degrees of freedom in a normal light field, direct interpolation would be computationally expensive. As most light fields tend to have a higher sampling rate in the spatial dimensions, we use a nearest neighbor interpolation there while performing bi-linear filtering in the directional dimensions, as they tend to have a lower sampling rate, and thus be more prone to aliasing.

5 RESULTS

We have implemented a software framework of our matrix optics representation of light fields. If required, the light field can be stored in a progressive, wavelet compressed data structure. The software renderer computes the current view in a texture and uses OpenGL to map it onto a polygon filling the screen.
For testing we have used both a synthetic light field and the 'buddha4' data set freely available from the Stanford light fields archive [Arc05]. The synthetic data set have a resolution of $128^2 \times 64^2$ samples, each point on the sampling plane covering an angular region of 120×120 degrees. This has proven to be a good balance between spatial and directional resolution while still keeping the original data size manageable. The buddha data set has a resolution of $256^2 \times 32^2$ samples.
The framework lets us implement and test a range of different optical setups. The most interesting application in computer graphics is of course to construct a 'camera' that lets the user interactively view a light field. Figure 5 shows a set of views from one of our synthetic light field. Figure 6 shows four images rendered from the buddha data set. The camera was constructed using two thin lenses operators offset by propagation operators. The camera transform was modeled by a rotation and yet a propagation operator. Aside from light field viewing, we also believe that the availability of other operators, such as interfaces, will allow for easy testing of a range of optical configurations.
We have performed renderings from both a direct and a wavelet compressed representation of the test data set. Rendering speeds are presented in Table 1. The machine used is a Linux-PC with an Intel Xenon 2.8GHz CPU and 2 Gigabytes of RAM.

	No AA	**Bi-Linear AA**
Direct	417 fps	228 fps
Wavelet	79 fps	21 fps

Table 1: Frame rates for rendering a $128^2 \times 64^2$ synthetic light field.

As can be seen from Table 1 the rendering speeds for the uncompressed data representation greatly exceeds those of the wavelet compressed data, making it a preferred choice if the whole data set can be fit into main memory. However, many light field data sets are huge, and may require compression. Nevertheless, our rendering algorithm achieves interactive framerates.

6 CONCLUSIONS

We have presented a light field a set of transformations inspired by matrix optics. This allows us to model optical elements such as lenses and interfaces into the image formation process. On its basis, we have implemented a real-time light field rendering framework. The framework can handle uncompressed light fields as well as wavelet compressed The wavelet compression scheme builds a hierarchical representation of the light field, and we have presented a fast way of accessing the data that integrates well with the presented transformations.
We believe matrix optics proves an elegant solution to modeling optical phenomena for light field rendering, and should provide an interesting complement to

intersection-based methods. The ability to freely combine the operators of different optical elements into one single matrix, results in a lot of flexibility to testing. It should also be noted that the framework is not restricted to pure image-based rendering. Many computer graphics problems can be posed as a sampling or transport of a light field. In such situations this framework can be used to model mappings of light fields through optical elements.

We plan to continue our work by investigating a range of questions and future possibilities. The core work of this paper, the matrix optics operators, is quite general. However, we wish to examine how well they behave for non-linear properties. In addition, aperture stops and ray integration will be implemented via Eq. 17 to allow for effects such as depth of field. It would also be interesting to see if the matrix operators presented here could be used in Fourier Slice Photography as presented in [Ng05].

7 ACKNOWLEDGEMENTS

This work is supported by the EC within FP6 under Grant 511568 with the acronym 3DTV.

References

[Arc05] The Stanford Light Fields Archive. http://graphics.stanford.edu/software/lightpack/lifs.html. World Wide Web, December 2005.

[BBM+01] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 425–432. ACM Press, 2001.

[Dau92] Ingrid Daubechies. *Ten lectures on wavelets*. Society for Industrial and Applied Mathematics, 1992.

[Fow75] Grant R. Fowles. *Introduction to Modern Optics*, chapter 10. Dover Publications, second edition, 1975.

[GB94] A. Gerrard and J. M. Burch. *Introduction to Matrix Methods in Optics*. Dover Publications, 1994.

[GGSC96] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In *Computer Graphics (SIGGGRAPH'96 Conf. Proc.)*, pages 43–54. ACM SIGGRAPH, August 1996.

[GMW02] Bastian Goldlücke, Marcus Magnor, and Bennett Wilburn. Hardware-accelerated dynamic light field rendering. In G. Greiner, H. Niemann, T. Ertl, B. Girod, and H.-P. Seidel, editors, *Proceedings Vision, Modeling and Visualization VMV 2002*, pages 455–462, Erlangen, Germany, November 2002. aka.

[LF99] Paul Lalonde and Alain Fournier. Interactive rendering of wavelet projected light fields. In *Proceedings of the 1999 conference on Graphics interface '99*, pages 107–114. Morgan Kaufmann Publishers Inc., 1999.

[LH96] M. Levoy and P. Hanrahan. Light field rendering. In *Computer Graphics (SIGGGRAPH'96 Conf. Proc.)*, pages 31–42. ACM SIGGRAPH, August 1996.

[LSG02] Reto Lütolf, Bernt Schiele, and Markus H. Gross. The light field oracle. In *Pacific Conference on Computer Graphics and Applications*, pages 116–126, 2002.

[Ng05] Ren Ng. Fourier slice photography. *ACM Trans. Graph.*, 24(3):735–744, 2005.

[PS01] Ingmar Peter and Wolfgang Straßer. The wavelet stream: Interactive multi resolution light field rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 127–138. Springer-Verlag, 2001.

[SDS96] Eric J. Stollnitz, Tony D. Derose, and David H. Salesin. *Wavelets for computer graphics: theory and applications*. Morgan Kaufmann Publishers Inc., 1996.

[SH99] Peter-Pike Sloan and Charles Hansen. Parallel lumigraph reconstruction. In *PVGS '99: Proceedings of the 1999 IEEE symposium on Parallel visualization and graphics*, pages 7–14. ACM Press, 1999.

[SP96] Amir Said and William A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:243–250, 1996.

[VPM+03] Daniel Vlasic, Hanspeter Pfister, Sergey Molinov, Radek Grzeszczuk, and Wojciech Matusik. Opacity light fields: interactive rendering of surface light fields with view-dependent opacity. In *SI3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 65–74, New York, NY, USA, 2003. ACM Press.

Figure 5: Five views of a test light field as taken by a perspective camera. The camera was constructed by combining lens and propagation operators Placement relative to the light field is controlled by a rotation and a propagation operator.

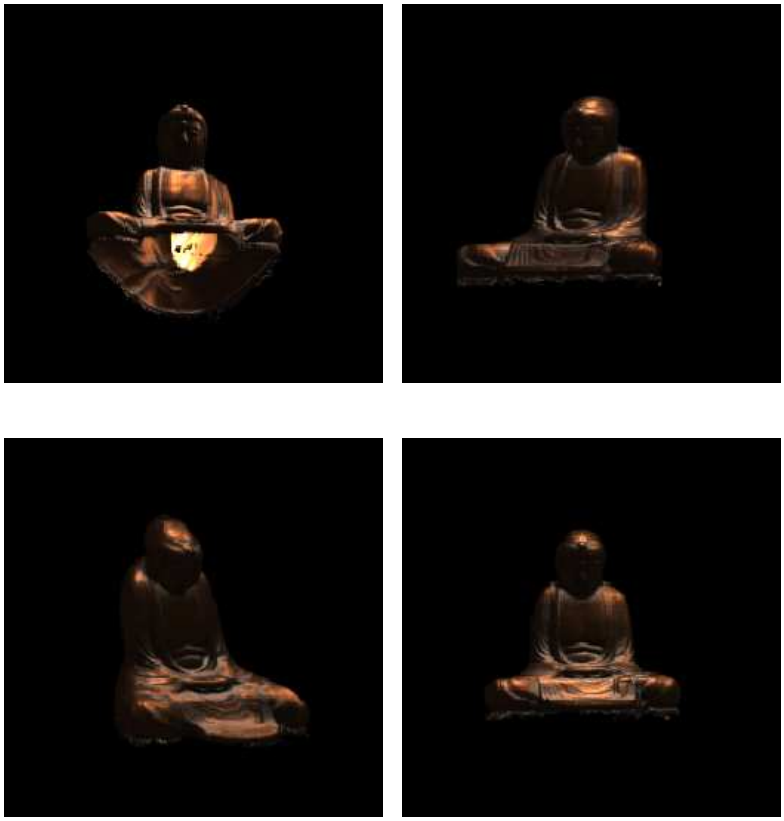

Figure 6: Four views of the buddha light field.