Hardware-based rendering of full-parallax synthetic holograms

Alf Ritter, Joachim Böttger, Oliver Deussen, Matthias König, and Thomas Strothotte

We present a method for efficiently calculating the interference of complex-valued two-dimensional wave patterns that is useful during the generation of synthetic holograms. These patterns are represented as a special kind of images (textures), and the interference is calculated in a computer graphics rendering process. This enables us to leverage hardware support for holographic imaging that is implemented in many state-of-the-art computer workstations. Using this approach, we gain a speedup of a factor of 60–90 compared with conventional calculation methods for interfering wave patterns. Our method is evaluated numerically, examples are shown, and the program code is outlined. © 1999 Optical Society of America

OCIS codes: 090.1760, 090.1970.

1. Introduction

With research in synthetic holography a common goal has been to reduce the enormous computational effort inherent to synthetic hologram generation.

In the case of stereograms,[1] several authors have used computer graphics hardware in recent years. Lucente,[2] Lucente and Galyean,[3] and Halle and Kropp[4] take advantage of the rendering facilities implemented in graphics workstations for generating the set of two-dimensional (2-D) views that is needed for stereogram production and for the superposition of basis fringes that serve as diffractive elements in holographic stereograms. Haines and Haines[5] also apply computer graphics rendering to generate 2-D views as a basis for stereograms.

In our study these ideas are extended to interfere complex-valued wave patterns by hardware-assisted processing of textured images.[6] These patterns represent arbitrary 2-D wave fields or, as in the case of synthetic holography, planar cross sections through the emitted wave fields of light sources. Processing these patterns in a computer graphics rendering process results in a significant speedup for calculating interference.

The paper is organized as follows. After a short introduction to synthetic holography, the stages of our holographic rendering process are described, followed by a discussion of methods for texture-based interference simulation. Next, an extension to lines and curves is shown. Finally, the results are reviewed and a future study is outlined.

2. Synthetic Holography

In its general form the information to be recorded on a holographic plate is derived from solution of the Kirchhoff diffraction integral for a given object

$$E(\mathbf{p}') = \frac{1}{i\lambda}\int_{\mathbf{p}}^{\mathbf{S}} E(\mathbf{p})\,\frac{1}{r}\exp\left(\frac{2\pi i r}{\lambda}\right)\cos(\alpha)\mathrm{d}\mathbf{S}, \qquad (1)$$

where $E(\mathbf{p})$ is the electric field strength contributed from point $\mathbf{p}$ of the object surface $\mathbf{S}$, $E(\mathbf{p}')$ is the field strength at point $\mathbf{p}'$ of the holographic plate, λ is the wavelength, r is the distance between $\mathbf{p}$ and $\mathbf{p}'$, and α is the angle between $\overrightarrow{\mathbf{pp}}'$ and the incident illumination wave.

A typical approximation of Eq. (1) is performed by decomposition of the object surface into N discrete point sources that all contribute to the hologram in the form of Fresnel zone plates[7]:

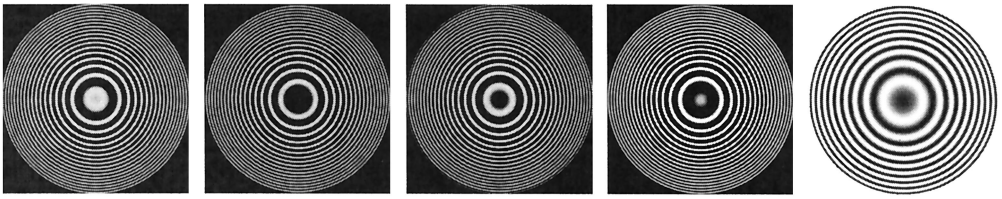
$$E(\mathbf{p}') = \frac{1}{i\lambda}\sum_{i}^{N} E_i\,\frac{1}{r}\exp\left(\frac{2\pi i r}{\lambda}\right)\cos(\alpha). \qquad (2)$$

The authors are with the Faculty of Computer Science, Otto-von-Guericke University of Magdeburg, D-39106 Magdeburg, Germany. O. Deussen's e-mail address is deussen@isg.cs.uni-magdeburg.de.

Received 8 June 1998; revised manuscript received 23 November 1998.

0003-6935/99/081364-06$15.00/0

© 1999 Optical Society of America



(a) (b) (c) (d) (e)

Fig. 1. Textures representing wave patterns of point sources: (a)–(d) together form a single complex texture: red channel (positive real part), green channel (negative real part), blue channel (positive imaginary part), and alpha channel (negative imaginary part) are shown as gray-scale images. (e) The Fresnel zone plate that is obtained by interference of the complex texture with a plane wave.

Here E_i is the field strength emitted by the point light source $\mathbf{p}_i$. The evaluation of Eq. (2) is still quite time consuming. Below, we suggest our solution for reducing the computational effort by means of computer graphics hardware.

3. Hardware-Based Generation of Holograms

Our method of holographic imaging involves the following steps. First, a set of special images is precomputed. To represent complex numbers, several color channels of the image are combined; the result is what we call complex texture. This data structure allows for performing hardware-based interference between different wave patterns.

We then generate the hologram by calculating the interference between complex textures that represent the light sources, by calculating the interference with a reference wave, and by determining the desired intensities with lookup tables. All of these steps are performed as standard graphics hardware operations based on the common graphics language OpenGL.[8]

We reconstruct the resulting hologram either optically by recording it on film and reconstructing the image in a laser setup, or by computer simulation. The examples in this paper were reconstructed with the optical simulation system DigiOpt introduced by Aagedal *et al.*[9] by application of a Rayleigh–Sommerfeld transform.

A. Precomputing Complex Textures

A complex texture consists of four color channels representing values for both the imaginary and the real components of all values of the complex wave pattern. Such a texture is encoded with a standard computer graphics image file format called RGBA. In an RGBA file four channels represent the colors red, green, and blue as well as an opacity component called the alpha channel.

Each complex number of the wave field is now encoded as a colored pixel in the complex texture. Because negative values cannot be stored as colors, and for numerical reasons, we use the red channel if the real part is positive and the green channel to store the absolute value if the value is negative. The same is done with the blue and the alpha channels for storing the imaginary part.

An example may clarify this: A cross section through a spherical wave originating from point $\mathbf{p}$ yields, according to Eq. (2), an electrical-field strength of the form

$$E(\mathbf{p}') = \frac{1}{r}\exp\left(\frac{2\pi i r}{\lambda}\right),$$

where $\mathbf{p}'$ is a point on the cross section, λ is the wavelength, and r is the distance between $\mathbf{p}$ and $\mathbf{p}'$.

The complex numbers of this field are encoded in the four color channels of Figs. 1(a)–1(d). Each gray-scale image represents the tonal value of the corresponding color. In Fig. 1(e) the resulting Fresnel zone plate is shown, which is the result of interfering the complex texture with a plane reference wave.

Each point light source of a virtual object emits a spherical wave that differs in phase. Therefore a set of complex textures representing spherical waves of different phase has to be precalculated.

Also, the light sources might differ in their distance to the virtual holographic plate. This fact is reproduced with scaling of the precalculated complex texture.

We assume that the original texture was calculated for a point source at distance z_0. If this texture should now represent a point at distance z, it has to be scaled by[10]

$$s = (z/z_0)^{1/2}. \tag{3}$$

Fortunately, such scaling of textures is also implemented by graphics hardware and can be used inside OpenGL.

The choice of the resolution of a complex texture has to be dependent on the size of the hologram and on the capacity that can be handled efficiently by the graphics hardware. Hologram resolutions needed for visual reconstruction typically exceed the maximum dimensions provided by the graphics hardware. Therefore the hologram is rendered in tiles that are combined to constitute the entire hologram.

To generate a hologram of N point sources $\mathbf{p}_i$, for each of the sources the phase and the distance to the virtual holographic plate are calculated and an appropriate complex texture is used. The locations of the $\mathbf{p}_i$ are parallel projected on the plate and now determine the center of the complex textures, which are scaled according to Eq. (3).

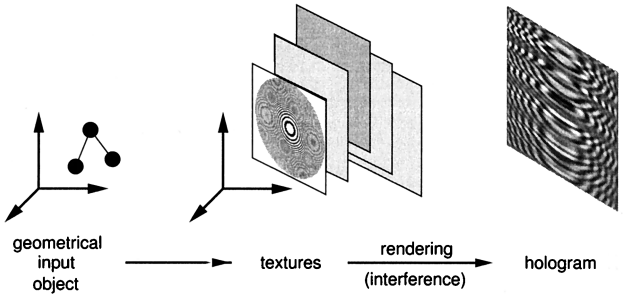


Fig. 2. Generation of a hologram: A geometric object is transformed into a set of complex textures. These textures are interfered to generate the hologram.

In Fig. 2 the process is shown. A geometric object is transformed into a set of complex textures. These textures are interfered to generate the hologram.

B. Interfering Complex Textures

The simplest way of generating holograms is to perform the superposition of gray-scale textures such as the one shown in Fig. 1(e). In this case the textures represent holograms of single graphic primitives and thus have only positive values.

This method was performed by Lucente,[2] Lucente and Galyean,[3] and Kropp[4] to superimpose basis fringes of stereograms and by Ritter *et al.*[11,12] to create full parallax holograms out of a moderate number of textures. For a larger number of textures and for an accurate simulation of the holographic imaging process, complex wave patterns have to be processed. As mentioned above, in this paper this is accomplished by interfering complex textures.

The difference between superposition and interference of textures is illustrated in Fig. 3. Figure 3(a) shows an object consisting of 155 points. In Fig. 3(b) the result of overlaying gray-scale textures is given, whereas the interference of the corresponding complex textures is shown in Fig. 3(c). Figure 3(d) shows the simulated reconstruction of the hologram.

For interfering the textures, a special hardware unit of high-end graphics workstations is used: the accumulation buffer.[13] It enables one to perform hardware-based arithmetic operations on whole images. The corresponding programming code is given in Appendix A.

The algorithm works as follows: First, all the complex textures are rendered to the accumulation buffer by addition of the image values representing positive values of the real and the imaginary parts of the complex numbers.

Second, the reference wave (also a complex texture) is added by use of an appropriate weight factor. To receive optimal contrast in the final hologram, the maximal absolute value of the wave field calculated thus far is determined and used as weight. In practice we use a heuristically determined weight $w = 0.5 * N$ for N given light sources.

Third, the negative values for the real and the imaginary parts (which are stored in the green and the alpha channel) are read from the accumulation buffer and subtracted from the corresponding positive values in the red and the blue channel.

At this point the accumulation buffer stores the complex-valued wave pattern, which results from interference of the waves that originate from the input elements.

In the final step the amplitude A of the electrical field in the hologram plane is calculated. The corresponding operation for each hologram pixel is described by

$$A = |E(\mathbf{p})| = (\mathrm{Re}_E{}^2 + \mathrm{Im}_E{}^2)^{1/2}, \tag{4}$$

where Re_E and Im_E are the real and the imaginary components of the electric field in the hologram plane. Since the film material used for hologram recording is sensitive to the intensity I as the square of the amplitude A [see Eq. (4)], it is sufficient to compute the intensity[14]

$$I = A^2 = \mathrm{Re}_E{}^2 + \mathrm{Im}_E{}^2. \tag{5}$$

Equation (5) is implemented with a so-called pixel map operation while the frame buffer is read out with the accumulated interference pattern. These operations are normally used for gamma correction and map each value of the frame buffer to a value given in the pixel map table.

In our case a table is installed that performs a mapping of the values to their square. We obtain the sum in Eq. (5) by reading out the overall luminance of the two color channels (red and blue) used

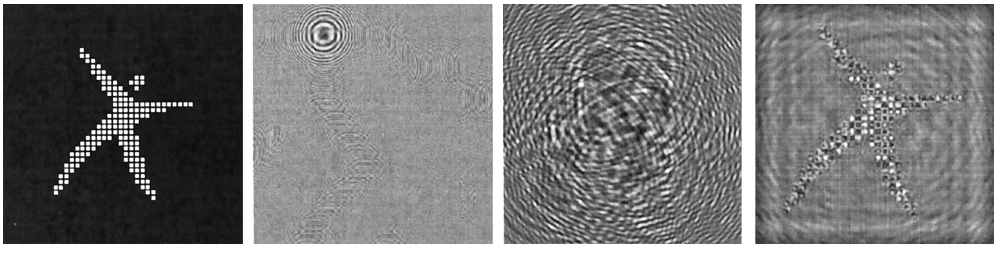
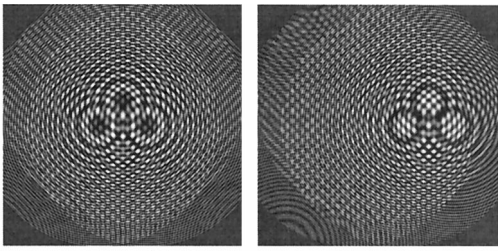


Fig. 3. Construction of a hologram from a set of points: (a) input object, (b) result obtained by superposition of gray-scale textures, (c) interference of complex textures for point sources with random phase, (d) reconstruction of (c) including typical speckles.



(a) (b)

Fig. 4. Off-axis reconstruction: (a) hologram of point sources, with the reference wave parallel to the virtual holographic plate; (b) an inclined wave was chosen.

for representing real and imaginary components of the complex numbers.

Thus all postprocessing operations for determining the intensity are facilitated by OpenGL and are therefore supported by the graphics hardware. This allows for creating holograms of large numbers of textured elements with high performance on workstations with a hardware accumulation buffer.

C. Off-Axis Holograms

To generate off-axis reconstructions, the complex texture for the reference wave is changed. With an inclined wave field, the generated patterns change their shape appropriately. Figure 4 shows two holograms of three point sources. The hologram in Fig. 4(a) was generated with a reference wave that was parallel to the holographic plate; in Fig. 4(b) an inclined reference wave was used. In this case the center of the resulting zone plates in the hologram move out of the center of the textures, which causes an off-axis reconstruction.

4. Point-Based Holograms

As described above the most straightforward method of generating a hologram is to represent the input object with a number of point sources. This is illustrated in Fig. 3.

For display purposes it is important to have holograms with diffuse surfaces. This is achieved by point sources that emit light with random phase. Otherwise the object would appear glossy.[15] We did this by precomputing a set of 32 textures that represent zone plates at different phases. These textures are used randomly for the point light sources.

One problem in combination with a number of objects that emit light with differing phase is the occurence of speckle patterns [cf., Fig. 3(d)]. Speckle patterns are inherent to coherent optics and can be suppressed, for example, by optimization of the phase distribution over the points.[16] The optimization itself is beyond the scope of this paper, but the speedup of the proposed method should also speed up the optimization process in order to get the right phase factors.

To assemble surfaces, a number of point sources are placed with sufficient density on the object surface such that the reconstruction yields a continuous appearance. Figure 5(a) shows a triangle approximated by 144 points. In Fig. 5(b) the holographic plate is shown, and Figs. 5(c) and 5(d) show reconstructions.

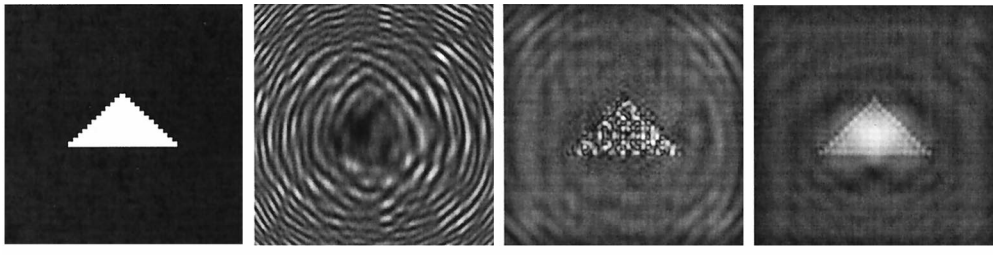
5. Imaging Lines and Curves

A particularly appealing aspect of the concepts presented thus far is that they can be extended directly to represent other primitives such as line and curve segments. This has a marked effect on the overall complexity by decreasing the number of textured elements to be processed during hologram rendering. Instead of having a texture for each point on a line or curve, we are able to represent an entire line segment with one element.

A point source emits a spherical wave. Lines emit cylindrical and conical waves, depending on the phase distribution along the line.[15,17] These wave patterns can also be stored as complex textures. This enables assembling holograms of a set of lines and curves. As for points, the elements are projected by parallel projection on the virtual holographic plate.

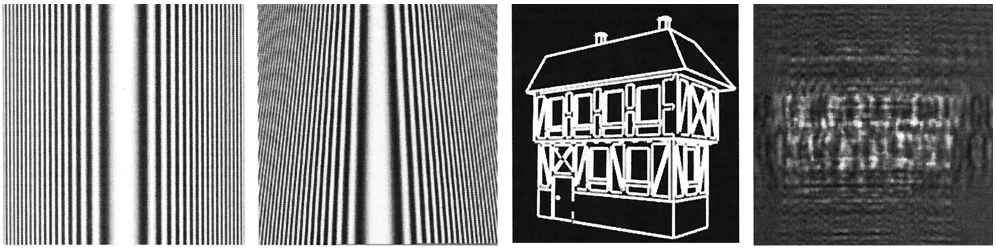
An example can be seen in Figure 6. In Fig. 6(a) the hologram of a single vertical line parallel to the hologram plane is shown. If the line is inclined to the hologram, the corresponding complex texture is also inclined and scaled according to Eq. (3); the resultant hologram is shown in Fig. 6(b).

Representing inclined lines by inclined complex textures is possible, because if a linear phase distri-



(a) (b) (c) (d)

Fig. 5. Holographic image of a surface: (a) input object assembled out of 144 points, (b) corresponding hologram with random phase, (c) reconstruction of (b), (d) reconstruction of a hologram of the same object with points in phase.



(a) (b) (c) (d)

Fig. 6. Holograms of lines: (a) cylindrical wave originating from a line, (b) conical wave pattern of an inclined line generated by transformation of the texture, (c) input object assembled from lines; (d) hologram of (c) with cylindrical and conical wave patterns.

Table 1. Time Consumed for Calculating Holograms of Different Input Complexity (Number of Points) and Size (512 × 512, 1024 × 1024, 5120 × 5120 Pixels) in Seconds and Speedup of the Texture-Based Approach Compared with the Traditional Method

	Traditional			Texture Based			Speedup		
Points	512	1024	5120	512	1024	5120	512	1024	5120
1	2.5	9.9	255	0.19	0.75	18	13	13	14
10	4.8	19.3	483	0.21	0.84	20	23	23	44
100	28.5	113	2850	0.5	2.0	42	57	57	68
1000	250	1004	25000	3.6	15	275	69	67	91

bution along the line is assumed, a conical wave is emitted.[17] Unfortunately, this cannot be done if random phases are used. In this case the textures of a set of lines with different inclination angles has to be precalculated.

In Fig. 6(c) an object composed of several hundred lines is shown, and Fig. 6(d) shows its hologram with a linear phase distribution along each line.

6. Evaluation of the Proposed Method

It is instructive to analyze the proposed method with respect to numerical accuracy and statistics about the time consumption.

The graphics hardware allows textures to have a range of 8 bit for each color channel, which corresponds to 256 intensity levels in each component. As holograms and other diffractive elements are usually constructed from less than 256 intensity levels, this is sufficient. The accumulation buffer has a depth of 25 bit, which allows for interference as great as 2^{17} complex textures without numerical problems. The result is obtained with an accuracy of 8 bit.

To analyze the time consumption, we compared the texture-based approach with traditional hologram calculation in which Eq. (2) has to be determined for each point of the hologram. Implementations of both methods were tested on a Silicon Graphics Onyx2 computer with two R10000 (195 MHz) processors and InfiniteReality graphics.

Table 1 shows the timing results in seconds for three hologram resolutions and a varying number of input points as well as the speedup of the texture-based approach over the traditional method. A factor of 57–91 is achieved for all but the trivial cases.

If lines are generated, the speedup is much better, because the lines are represented by one or a small number of textures, whereas a traditional method still has to evaluate Eq. (2) for each line by integration. Even if the optimized methods for lines are applied, the speedup factor remains in the same level.

7. Conclusions

In this paper we have described a method that uses computer graphics hardware for rapidly generating interference between complex wave patterns for use in computer-generated full-parallax holograms.

All steps of the process have been implemented on the basis of hardware supported operations with the standard graphics computer language OpenGL. This results in a considerable speedup compared with traditional methods and forms a new bridge between computer graphics and synthetic holography.

Our timing results show that for moderate resolutions and object complexities the generation of full-parallax holograms can achieve interactive rates.

Furthermore, the complex textures as defined in this paper can be used in various places where discrete fields of complex values are to be treated. This involves operations on 2-D and three-dimensional wave fields in physics and in electrical engineering.

In a future study the direct mapping of other geometric primitives, e.g., triangles, to the holographic equivalent needs to be investigated. This transformation enables a holographic imaging of surfaces that is more efficient than by assembly of point sources alone.

One important property of our method is that complex wave patterns resulting from the interference of

a number of elements of the holographic equivalent can be stored themselves as a single complex texture. This will allow us to reuse parts of scenes without recalculation.

Appendix A: Programming Code for Hardware-Based Interference

As described in Section 3, all steps during the rendering of the hologram are performed with OpenGL and graphics hardware. Below, an outline of the algorithm is given.

The image is encoded as follows: The red channel stores the positive real part of the values, the green channel stores the negative real part, the blue channel stores the positive imaginary part, and the alpha channel stores the negative imaginary part.

```
procedure InterfereElements()
{ glBlendFunc (GL_ONE, GL_ZERO);
  glEnable (GL_BLEND);
  foreach(element) {
     render(element)

     glAccum(GL_ACCUM,+1.0); }
  render(reference);
  glAccum(GL_ACCUM, refweight);
  glReadPixels (GL_GREEN, greenBuffer);
  glReadPixels (GL_ALPHA, alphaBuffer);
  glClear( );
  glDrawPixels (GL_RED, greenBuffer);
  glDrawPixels (GL_BLUE, alphaBuffer);
  glAccum (GL_ACCUM, −1.0);

  glPixelMap();

  glReadPixels();

}
```

```
/* OpenGL settings */

/* For all holographic elements */
/* draw the texture that represents the */
/* complex numbers */
/* Transfer into the accumulation buffer */
/* Interfere with reference wave, using an */
/* appropriate weight factor refweight */
/* read green and alpha values */

/* Clear screen */
/* Draw green and alpha to */
/* red and blue */
/* Subtract them as they represent */
/* negative values */
/* Install pixel map table accord. to Eq. (5) */
/* but without using green and alpha val. */
/* Map pixels and transfer to memory */
/* according to Eq. (5) */
```

References

1. S. A. Benton, "Survey of holographic stereograms," in *Processing and Display of Three-Dimensional Data*, J. J. Pearson, ed., Proc. SPIE **367,** 15–19 (1983).
2. M. Lucente, "Interactive three-dimensional holographic displays: seeing the future in depth," ACM (Assoc. Comput. Mach.) Comput. Graphics **31**(2), 63–66 (1997).
3. M. Lucente and T. A. Galyean, "Rendering interactive holographic images," in *Proceedings of ACM SIGGRAPH '95*, R. Cook, ed., Annual Conference Series (ACM Press, New York, 1995), pp. 387–394.
4. M. W. Halle and A. B. Kropp, "Fast computer graphics rendering for full parallax spatial displays," in *Practical Holography XI*, S. A. Benton, ed., Proc. SPIE **3011,** 105–112 (1997).
5. K. Haines and D. Haines, "Computer graphics for holography," IEEE Comput. Graphics Appl. **12,** 37–46 (1992).
6. M. J. Kilgard, "Realizing OpenGL: two implementations of one architecture," in *Proceedings of ACM SIGGRAPH/EuroGraphics Workshop on Graphics Hardware*, S. Molnar and B.-O. Schneider, eds. (ACM Press, New York, 1997), pp. 45–55.
7. J. P. Waters, "Holographic image synthesis utilizing theoretical methods," Appl. Phys. Lett. **9,** 405–407 (1966).
8. J. Neider, T. Davis, and M. Woo, *OpenGL Programming Guide: The Official Guide to Learning OpenGL* (Addison-Wesley, Bonn, Germany, 1993).
9. H. Aagedal, Th. Beth, H. Schwarzer, and S. Teiwes, "Design of paraxial diffractive elements with the CAD system DigiOpt," in *Diffractive and Holographic Optics Technology II*, I. Cindrich and S. H. Lee, eds., Proc. SPIE **2404,** 50–58 (1994).
10. M. Born and E. Wolf, *Principles of Optics*, 6th ed. (Pergamon, Oxford, 1980).
11. A. Ritter, Th. Benziger, O. Deussen, Th. Strothotte, and H. Wagener, "Synthetic holograms of splines," in *3D Image Analysis and Synthesis '97*, H.-P. Seidel, B. Girod, and H. Niemann, eds., (infix Verlag, Sankt Augustin, Germany, 1997), pp. 11–18.
12. A. Ritter, O. Deussen, H. Wagener, and Th. Strothotte, "Holographic imaging of lines: a texture-based approach," in *Proceedings of the International Conference on Information Visualization IV '97*, P. Storms, ed. (IEEE Computer Society Press, Los Alamitos, Calif., 1997), pp. 272–278.
13. J. S. Montrym, D. R. Baum, D. L. Dignam, and C. J. Migdal, "InfiniteReality: a real-time graphics system," in *Proceedings of ACM SIGGRAPH '97*, T. Whitted, ed., Annual Conference Series (ACM Press, New York, 1997), 293–302.
14. J. W. Goodman, *Introduction to Fourier Optics*, 2nd ed. (McGraw-Hill, New York, 1968).
15. C. Frère, D. Leseberg, and O. Bryngdahl, "Computer-generated holograms of three-dimensional objects composed of line segments," J. Opt. Soc. Am. A **3,** 726–730 (1986).
16. W. Lauterborn, Th. Kurz, and M. Wiesenfeldt, *Coherent Optics, Fundamentals and Applications* (Springer-Verlag, Berlin, 1995).
17. D. Leseberg, "Computer generated holograms: cylindrical, conical, and helical waves," Appl. Opt. **26,** 4385–4390 (1987).