First publ. in: Applied Optics, 38 (1999), 8, pp. 1364-1369

Hardware-based rendering of full-parallax synthetic holograms

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

Faculty of Computer Science

Otto-von-Guericke University of Magdeburg, Germany

Abstract

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

Keywords:

Computer-generated hologram, graphics hardware, interference

1. Introduction

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

In the case of stereograms[1], several authors used computer graphics hardware in recent years. Lucente et al.[2,3], as well as Halle and Kropp[4] take advantage of the rendering facilities implemented in graphics workstations for generating the set of 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 work these ideas are extended in order to interfere complex-valued wave patterns by hardware-assisted processing of textured images[6]. These patterns represent arbitrary two-dimensional 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 graphic rendering process results in a significant speed up 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 future work is outlined.

2. Synthetic Holography

In its general form, the information to be recorded on a holographic plate is derived by solving the Kirchhoff diffraction integral for a given object

$$E(\mathbf{p}') = \frac{1}{i\lambda} \int_{\mathbf{p}}^{\mathbf{S}} E(\mathbf{p}) \; \frac{1}{r} \; exp\Big(\frac{2\pi i r}{\lambda}\Big) \; \cos(\alpha) \; d\mathbf{S} \tag{1}$$

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 the distance between $\mathbf{p}$ and $\mathbf{p}'$, and α the angle between $\mathbf{p}' - \mathbf{p}$ and the incident illumination wave.

A typical approximation of Equation 1 is performed by decomposing the object surface into N discrete point sources which all contribute to the hologram in form of Fresnel zone plates [7]:

$$E(\mathbf{p}') = \frac{1}{i\lambda} \sum_{i}^{N} E_i \; \frac{1}{r} \; exp\Big(\frac{2\pi i r}{\lambda}\Big) \; \cos(\alpha). \tag{2}$$

E_i is the field strength emited by the point light source $\mathbf{p}_i$. The evaluation of Equation 2 is still very time-consuming. In the following, we will 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 we call complex texture. This data structure allows performing hardware based interference between different wave patterns.

The hologram is then generated by calculating the interference between complex textures representing the light sources, the interference with a reference wave and by determing the desired intensities utilizing look-up tables. All of these steps are performed as standard graphics hardware operations based on the common graphics language OpenGL[8].

The resulting hologram is reconstructed 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 applying a Rayleigh-Sommerfeld transform.

A. Pre-computing Complex Textures

A complex texture consists of four color channels representing values for both the imaginary and the real component of all values of the complex wave pattern. Such a texture is encoded using a standard image file format of computer graphics called RGBA. In an RGBA file four channels represent the colors red, green and blue as well as an opacity component named 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 positve, and the green channel to store the absolute value if the value is negative. The same is done with the blue and the alpha channel 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 Equation 2 an electrical field strength of the form

$$E(\mathbf{p}') = \frac{1}{r}\ exp\Big(\frac{2\pi ir}{\lambda}\Big)$$

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

The complex numbers of this field are encoded in the four color channels of Figure 1(a)-

(d). Each grey-scale image represents the tonal value of the corresponding color. In Figure 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 which differs in phase. Therefore a set of complex textures representing spherical waves of different phase has to be pre-calculated.

Also, the light sources may differ in their distance to the virtual holographic plate. This fact is reproduced by scaling the pre-calculated complex texture.

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

$$s = \sqrt{z/z_0}. \tag{3}$$

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

The resolution of a complex texture has to be chosen in dependency to the size of the hologram and to what can be handled efficiently by the graphics hardware. Hologramm resolutions needed for visual reconstruction typically exceed the maximum dimensions provided by the graphics hardware. Therefore, the hologram is rendered in tiles which are combined to constite the enire 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 is calculated and an appropriate complex texture is used. The locations of the $\mathbf{p}_i$ are parallel projected on the plate and determine now the center of the complex textures, which are scaled according to Equation 3.

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

B. Interfering Complex Textures

The simplest way of generating holograms is to perform the superposition of grayscale textures like the one shown in Figure 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 et al.[2,3] 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 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 Figure 3. Figure 3(a) shows an object consisting of 155 points. In Figure 3(b) the result of overlaying gray scale textures is given whereas the interference of the corresponding complex textures is shown in Figure 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 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 adding the image values representing positive values of real and imaginary part of the complex numbers.

Second, the reference wave (also a complex texture) is added by using an appropriate weight factor. To receive optimal contrast in the final hologram, the maximal absolute value of the so far calculated wave field 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 real and imaginary part (which are stored in the green and alpha channel) are read from the accumulation buffer and subtracted from the corresponding positive values in the red and blue channel.

At this point the accumulation buffer stores the complex-valued wave pattern which results by interfering the waves originating from the input elements.

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

$$A = |E(\mathbf{p})| = \sqrt{re_E^2 + im_E^2}, \tag{4}$$

where re_E and im_E are the real and 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 Equation 4), it is sufficient to compute the intensity[14]

$$I = A^2 = re_E^2 + im_E^2. \tag{5}$$

Equation 5 is implemented by using a so called pixel map operation while reading out the frame buffer 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 which performs a mapping of the values to their square. The sum in Equation 5 is obtained by reading out the overall luminance of the two color channels (red and blue) used for representing real and imaginary components of the complex numbers.

Thus, all postprocessing operations for determining the intensity are facilitated by OpenGL and therefore supported by the graphics hardware. This allows creating holograms of a large number 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. By using a inclined wave field, the generated patterns change their shape appropriately. Figure 5 shows two holograms of three point sources. The hologram in Figure 5(a) was generated using a reference wave which was parallel to the holographic plate, in Figure 5(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 by a number of point sources. This was illustrated in Figure 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]. In our case this was done by pre-computing 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. Figure 3(d)). Speckle patterns are inherent to coherent optics and can be suppressed for example by optimizing the phase distribution over the points[16]. The optimization itself is beyond the scope of this article, but the speed-up 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 is placed sufficiently dense on the object surface such that the reconstruction yields a continuous appearance. Figure 4 shows a triangle approximated by 144 points. In Figure 4(b) the holographic plate is shown whereas Figure 4(c) shows the reconstruction for points with different phase and Figure 4(d) for points in phase.

5. Imaging Lines and Curves

A particularly appealing aspect of the concepts presented thus far is that they can be extended directly in order to represent other primitives like 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 by 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 Figure 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 Equation 3; the resultant hologram is shown in Figure 6(b).

Representing inclined lines by inclined complex textures is possible because if a linear phase distribution 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 Figure 6(c) an object composed out of several hundred lines is shown, while Figure 6(d) shows its hologram using 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 255 intensity levels in each component. As holograms and other diffractive elements usually are constructed out of less than 255 intensity levels this is sufficient. The accumulation buffer has a depth of 25 bit which allows to interfere up to 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 to traditional hologram calculation in which Equation 2 has to be determined for each point of the holo-

gram. Implementations of both methods were tested on a Silicon Graphics Onyx2 computer with two R10000 (195MHz) 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 speed up of the texture-based approach over the traditional method. A factor of 57 to 91 is achieved for all but the trivial cases.

If lines are generated, the speed up is much better as the lines are represented by one or a small number of textures whereas a traditional method still has to evaluate Equation 2 for each line by integration. Even if the opimitzed methods for lines are applied the speed-up 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 using the standard graphics computer language OpenGL. This results in a considerable speed up compared to traditional methods and forms a new brigde 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 3-D wave fields in physics and electrical engineering.

In future work the direct mapping of other geometric primitives, e.g. triangles, to the holographic equivalent needs to be investigated. This transformation enables a more efficient holographic imaging of surfaces than just by assembling them of point sources.

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 itself as a single complex texture. This will allow reusing parts of scenes without recalculation.

Appendix A: Programming Code for Hardware-based Interference

As described in Section 3, all steps during rendering the hologram are performed by using OpenGL and graphics hardware. In the following the 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 the negative real part, blue stores the positive imaginary part and the alpha channel the negative imaginary part.

```
procedure InterfereElements()
{ glBlendFunc (GL_ONE, GL_ZERO);          /* OpenGL settings                    */
  glEnable (GL_BLEND);
  foreach(element) {                      /* For all holographic elements       */
    render(element)                       /* Draw the texture that represents the */
                                          /* complex numbers                    */
    glAccum(GL_ACCUM,+1); }               /* Transfer into the accumulation buffer */
  render(reference);                      /* Interfere with reference wave using an */
  glAccum(GL_ACCUM, refweight);           /* appropriate weight factor "refweight" */
  glReadPixels (GL_GREEN, greenBuffer);   /* read green and alpha values */
  glReadPixels (GL_ALPHA, alphaBuffer);
  glClear();                              /* clear screen                       */
  glDrawPixels (GL_RED, greenBuffer);     /* draw green and alpha to            */
  glDrawPixels (GL_BLUE, alphaBuffer);    /* red and blue                       */
  glAccum (GL_ACCUM, -1.0);               /* subtract them as they represent    */
                                          /* negative values */
```

```
    glPixelMap();                   /* Install pixel map table accord. to Eq.5 */
                                    /* but without using green and alpha val.  */
    glReadPixels();                 /* Map pixels and transfer to memory       */
                                    /* according to Equ.5                      */
}
```

REFERENCES

1. S. A. Benton. Survey of holographic stereograms. *SPIE Processing and Display of Three-Dimensional Data*, 367:15–19, 1983.

2. M. Lucente. Interactive three-dimensional holographic displays: Seeing the future in depth. *ACM Computer Graphics*, 31(2):63–66, May 1997.

3. M. Lucente and T. A. Galyean. Rendering interactive holographic images. In R. Cook, editor, *Proceedings of ACM SIGGRAPH '95 (Los Angeles, August 1995)*, Annual Conference Series, pages 387–394, New York, 1995. ACM Press, New York.

4. M. W. Halle and A. B. Kropp. Fast computer graphics rendering for full parallax spatial displays. In S. A. Benton, editor, *Practical Holography XI*, volume 3011 of *SPIE Proceedings*, pages 105–112, Bellingham, 1997.

5. K. Haines and D. Haines. Computer graphics for holography. *IEEE Computer Graphics and Applications*, 12(1):37–46, January 1992.

6. M. J. Kilgard. Realizing OpenGL: Two implementations of one architecture. In S. Molnar and B.-O. Schneider, editors, *Proceedings of ACM SIGGRAPH/EuroGraphics Workshop on Graphics Hardware*, pages 45–55, Los Angeles, August 1997. ACM Press.

7. J. P. Waters. Holographic image synthesis utilizing theoretical methods. *Applied Physics Letters*, 9(11):405–407, 1966.

8. J. Neider, T. Davis, and M. Woo. *OpenGL Programming Guide: The Official Guide to Learning OpenGL*. Addison-Wesley, Bonn, Paris, Reading, 1993.

9. H. Aagedal, Th. Beth, H. Schwarzer, and S. Teiwes. Design of paraxial diffractive elements with the CAD system DigiOpt. In I. Cindrich and S. H. Lee, editors, *Diffractive and Holographic Optics Technology II*, volume 2404 of *SPIE Proceedings*, pages 50–58, Bellingham, 1994.

10. M. Born and E. Wolf. *Principles of Optics*. Pergamon Press, Oxford, 6 edition, 1980.

11. A. Ritter, Th. Benziger, O. Deussen, Th. Strothotte, and H. Wagener. Synthetic holograms of splines. In H.-P. Seidel, B. Girod, and H. Niemann, editors, *3D Image Analysis and Synthesis '97 (Erlangen, November 1997)*, pages 11–18, Sankt Augustin, 1997. infix-Verlag.

12. A. Ritter, O. Deussen, H. Wagener, and Th. Strothotte. Holographic imaging of lines: A texture-based approach. In P. Storms, editor, *International Conference on Information Visualization IV'97 (London, August 1997)*, pages 272–278, Los Alamitos, 1997. IEEE Computer Society, New York.

13. J. S. Montrym, D. R. Baum, D. L. Dignam, and C. J. Migdal. InfiniteReality: A real-time graphics system. In T. Whitted, editor, *Proceedings of ACM SIGGRAPH '97*, Annual Conference Series, pages 293–302. ACM Press, New York, August 1997.

14. J. W. Goodman. *Introduction to Fourier Optics*. McGraw-Hill, New York, 2 edition, 1968.

15. C. Frère, D. Leseberg, and O. Bryngdahl. Computer-generated holograms of three-dimensional objects composed of line segments. *Journal of the Optical Society of America (JOSA)*, 3(5):726–730, 1986.

16. W. Lauterborn, Th. Kurz, and M. Wiesenfeldt. *Coherent Optics, Fundamentals and Applications*. Springer-Verlag, Berlin, Heidelberg, New York, 1995.

17. D. Leseberg. Computer generated holograms: Cylindrical, conical, and helical waves. *Applied Optics*, 26(20):4385–4390, 1987.

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

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

Fig. 3. Construction of a hologram out of 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.

Fig. 4. 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.

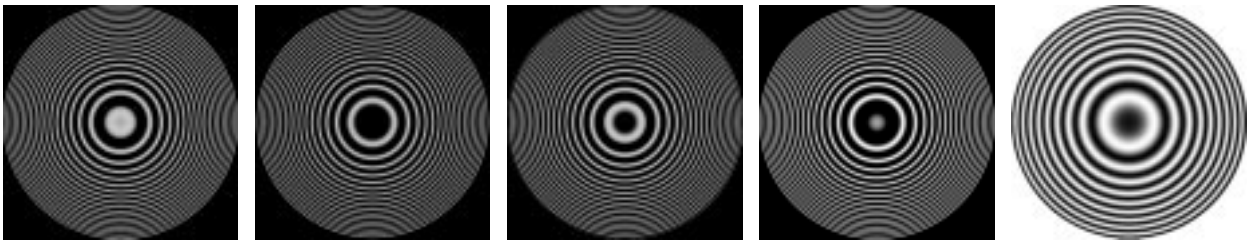
Fig. 5. Off-axis reconstruction: (a) Hologram of three point sources, the reference wave is parallel to the virtual holographic plate; (b) an inclined wave was chosen.

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

TABLES

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 speed-up of the texture-based approach over the traditional method.

	Traditional			Texture-based			Speed up		
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



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

Figure 1

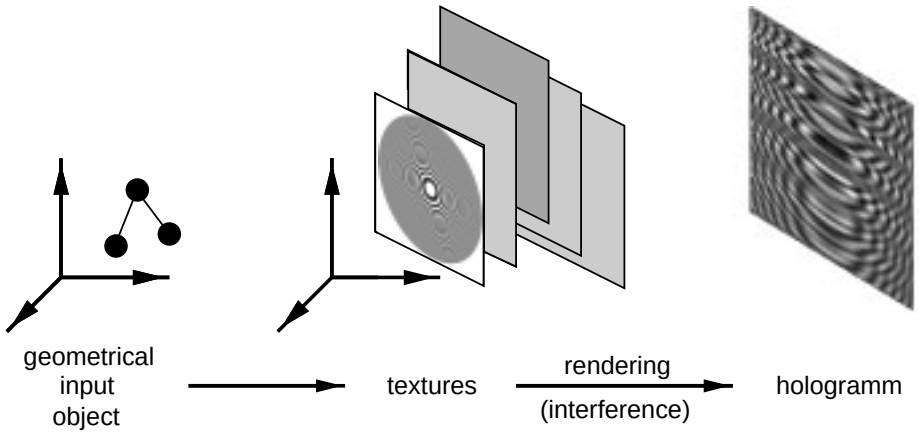
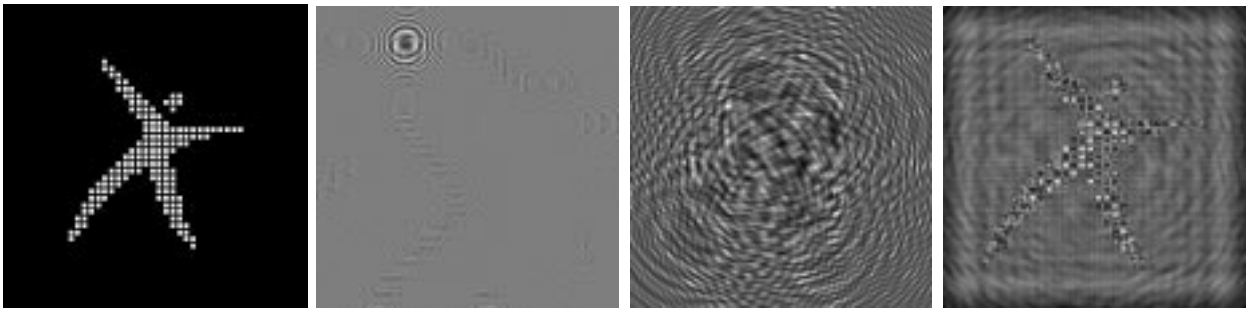
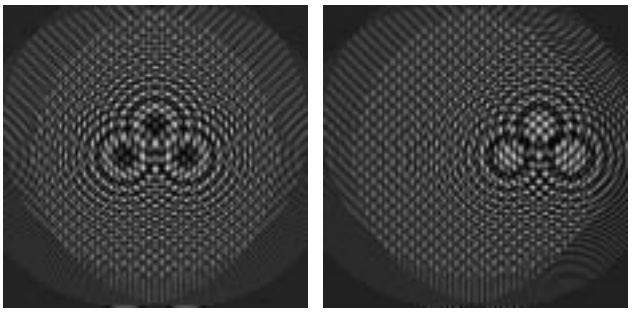

geometrical
input
object
textures
rendering
(interference)
hologramm

Figure 2



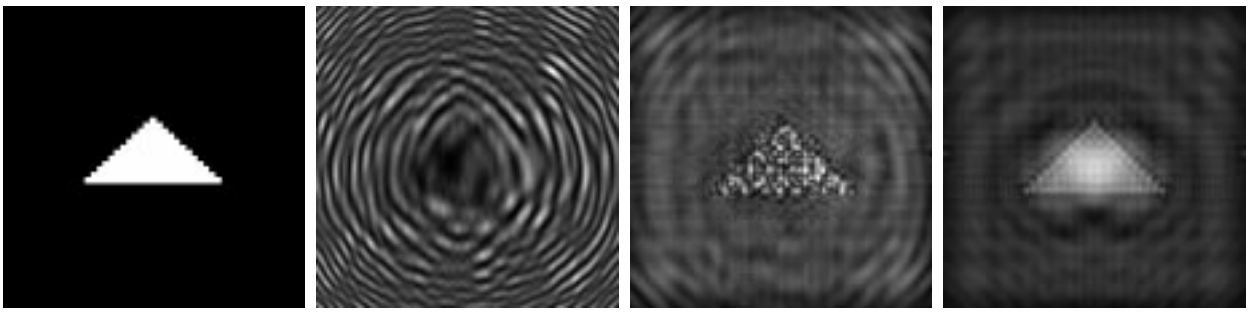
(a) (b) (c) (d)

Figure 3



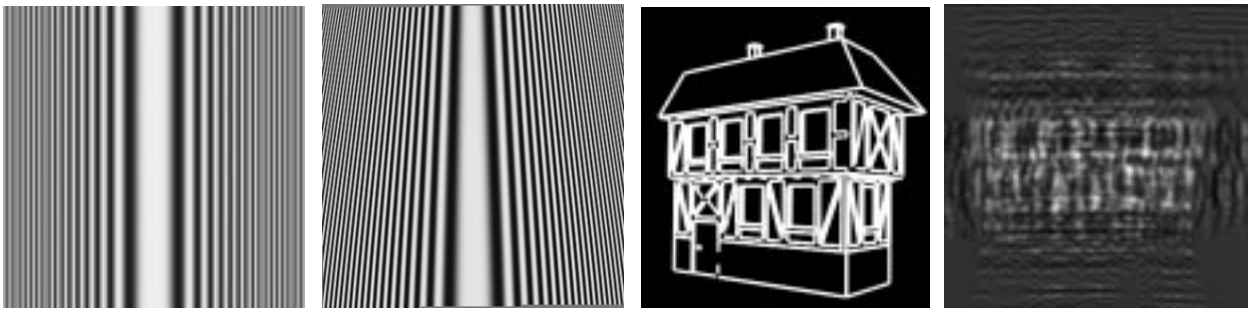
(a) (b)

Figure 4



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

Figure 5



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

Figure 6