Synthetic Holograms of Splines

Alf Ritter[1], Thomas Benziger[2], Oliver Deussen[1], Thomas Strothotte[1], Hubert Wagener[1]

Otto-von-Guericke University of Magdeburg
[1]Department for Simulation and Graphics
[2]Department for Materials Technology and Materials Testing
PSF 4120, D-39016 Magdeburg, Germany
Email: [1]{alf,deussen,wagener,tstr}@isg.cs.uni-magdeburg.de,
[2]thomas.benziger@masch-bau.uni-magdeburg.de

Abstract

In this paper we present an approach for the holographic imaging of objects composed of splines. It is based on an imaging method for lines and extended to fit the requirements for the visualization of 3D curves. The input object consisting of splines is decomposed into line segments. These segments are subsequently transferred into the holographic equivalent—a set of textured rectangles appropriately placed in 3D space. Proper rectangle orientation and texture distortion allow a narrow approximation of holographic patterns generated by cylindrical and conical waves.

Our approach allows faster hologram generation compared to traditional methods. We employ computer graphics rendering, exploiting hardware support for the rendering of textured geometry.

1 Holographic Imaging

An important goal in 3D visualization is to provide the cues known from natural vision which make the perception of depth possible. These phenomena include for instance binocular disparity, ocular accommodation, and motion parallax. A number of display techniques [11] exist to provide a 3D sensation, but holographic displays are the only ones known today to provide *all* the depth cues. Therefore we believe that holography will play an important role in the future of 3D imaging.

The method of holographic imaging of objects was discovered by Dennis Gabor [7]. Holography is a two-step process. It consists of the *recording* and *reconstruction* stage [2]. The hologram is a diffraction screen reconstructing the wave field emitted by the object to be imaged. In the hologram recording step, the output from a laser is separated into two beams.

One beam illuminates the object, the other (reference beam) is directed to the holographic plate. The hologram records the interference pattern caused by the light reflected by the object and the reference beam. During reconstruction, the hologram is again illuminated with the reference beam. The diffracted wave field contains a three-dimensional copy of the original wave reflected by the object. Thus, the viewer looking through the hologram sees the image of the object in depth with changing perspective.

In synthetic holography the processes of optical holographic imaging are simulated. The transmittance of the holographic plate is calculated by a computer. The resulting hologram can be optically reconstructed as described above. The reconstruction also can be performed by means of computer simulation of diffraction [1]. Even the real time display of holographic images is possible [10].

A couple of methods dealing with hologram generation will be briefly described. In most cases the object to be imaged is assembled of a set of radiating points. Each point emits a spherical wave. In *direct simulation* the complex amplitude of the wave originating from each point is calculated for all hologram locations (pixels). This is computationally expensive. One method to reduce the effort is described in [9]. The contributions of each possible 3D point in the image volume are precomputed and stored in a *look-up table*. In the hologram generation step just the contributions of the actual points have to be accumulated. This method was extended to *holographic stereograms* [10].

The methods described so far are point-based. If the 3D input object consists of lines, the contribution of each line to the hologram can be calculated instead of decomposing the object into points. A line represents a large number of points. In computer graph-

ics, line drawings are used to express the essential in the shape of an object [17] with less resources. In our research we developed a method for the efficient holographic imaging of lines [14] and curves in 3D.

2 Holograms of Lines

2.1 The Conventional Approach

In holography, objects to be imaged are assembled of luminous geometric primitives. There are three categories of such primitives: points, lines and planes. These primitives emit waves which generate characteristic patterns on the hologram. A point source emits spherical waves, cylindrical waves originate from infinite luminous lines and plane waves are emitted by infinite planes [4]. In the hologram reconstruction step the waves are again generated and focus in their original objects. Thus the reconstruction of these primitives is accomplished.

In our case the patterns generated by cylindrical waves are of special interest [6]. Their description is illustrated in Figures 1(a) and 1(b). The z axis of our
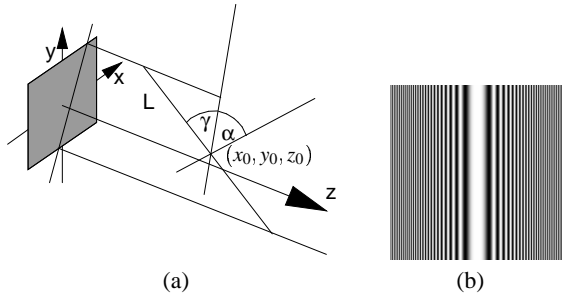

(a)   (b)

Figure 1: Line parameters (a) and holographic pattern of a cylindrical wave (b)

coordinate system is the optical axis. The hologram is in the xy plane. A cylindrical wave emitted by a line L parallel to the x axis and located at a distance $z = R$ to the hologram plane results in a complex amplitude $u(x,y,0)$ which can be described by the Fresnel approximation

$$u(x,y,0) = exp\left(i\frac{\pi}{\lambda R}y^2\right). \qquad (1)$$

λ is the wavelength. Figure 1(b) shows the pattern generated by such a line and computed using the Fresnel approximation from Equation 1. In order to be able to image lines of finite length, a clipping function $rect(x,a)$ is introduced. Clipping is used to restrict the pattern to an area corresponding to the length of the original line projected onto the hologram plane (a
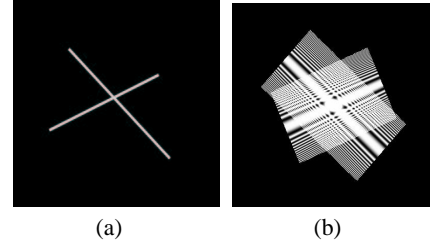

(a)   (b)

Figure 2: Image of two crossing lines, original object (a) and its hologram (b)

denotes the length of the projection). Effects occuring at the edges of these areas are disregarded [6].

If the line L is rotated by an angle α relative to the x axis, the point (x,y) on the hologram plane is rotated to a point (X,Y) with

$$X = x\cos\alpha - y\sin\alpha, \qquad Y = x\sin\alpha + y\cos\alpha. \qquad (2)$$

If, in addition, L is inclined to the hologram plane, the inclination angle γ has to be taken into account

$$R(X,\gamma) = z_0 - X\tan\gamma. \qquad (3)$$

This function determines the distance between the line and the points (pixels) on the hologram. A phase factor

$$exp\left[i\frac{2\pi}{\lambda z_0}(x_0 X + y_0 Y)\right]. \qquad (4)$$

is introduced to reconstruct the line outside the optical axis. These equations together describe the complex amplitude of a line L (recall Figure 1(a)) with a center point (x_0, y_0, z_0), an angle α relative to the x axis, and an inclination angle γ to the hologram plane [6]:

$$u(x,y,0) = exp\left[i\frac{2\pi}{\lambda z_0}(x_0 X + y_0 Y)\right] \times$$
$$exp\left[i\frac{\pi}{\lambda}\frac{Y^2}{R(X,\gamma)}\right] \times rect\left(\frac{X}{2a}\right). (5)$$

The combination of the complex amplitude with the reference wave yields the transmittance of the hologram ([6], [16]).

Figures 2(a) and 2(b) show an example of an input object composed of lines and its hologram. More examples can be found in [13]. In the hologram in Figure 2(b) two overlapping rectangular areas of wave patterns are to be seen. Clipping is performed in the x and y direction to ensure the correct line length in the reconstruction and to prevent high spatial frequencies. The latter cannot be properly transferred to the hologram.

2.2 Rendering Holographic Equivalents

The conventional approach demonstrated above includes computationally expensive operations. For

each hologram pixel, 2D rotations have to be performed and trigonometry functions are involved [6] in computing the hologram transmittance. In order to cope with this complexity, we introduce a new approach to hologram generation [14] using computer graphics methods to speed up hologram generation compared to the conventional computation procedure. As stated in Section 2.1, lines emit cylindrical waves. In this case there is a constant phase distribution along the line ([6], [8]). If the line is inclined to the hologram, a pattern as to be seen in Figure 3(a) results which is very complicated to simulate in our approach. Instead, a linear phase distribution is chosen (see [14] for details) which results in a conical wave (see Figure 3(b)).
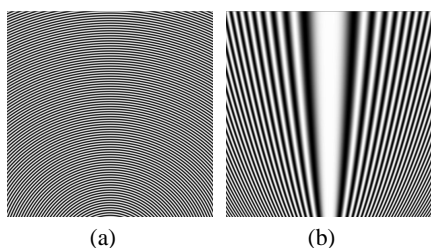

(a) (b)

Figure 3: Hologram of an inclined line with constant phase(a), a linear phase distribution (b)

Our approach is based on the rendering of textured geometry. Taking the input object consisting of line segments, we build up a second 3D object which we call the *holographic equivalent* [14] of the input object. The holographic equivalent consists of one *element* per input line. Each element is a textured 3D rectangle. The position (center point coordinates (x_0, y_0, z_0), see Figure 1(a)), and rotation (angles α and γ) of the input line are mapped to the referring element in the equivalent. An example of an object and its holographic equivalent is shown in Figure 4. The
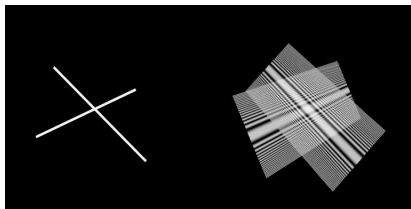

Figure 4: Two crossing lines (left) and their holographic equivalent (right)

elements (rectangles) are "centered" around the original lines. The original line represents the middle axis of the rectangle (determined by α and γ). However, the rectangles are restricted in their azimuthal orientation. They are rotated around their middle axis such that they are normal to the viewing plane (hologram plane) in this direction. This is necessary to ensure consistent reconstruction depths.

The *reference texture* mapped to the rectangles of the equivalent is precomputed using the Fresnel approximation (Equation 1). Under a perspective projection, the textured rectangles appear distorted depending on the angle γ. A converging pattern results which is already similar to a pattern generated by a conical wave. For an exact simulation, though, a quadratic distortion function needs to be applied along the rectangles (Figure 4). In order to accomplish this distortion, the rectangles are subdivided into strips. For the distortion the texture coordinates along the strip vertices are altered [14].

The hologram is gained just by rendering the equivalent (projection of the equivalent to the image/hologram plane). The equivalent is rendered in tiles to fit the hologram resolution requirements [14]. If the input object consisting of lines is transformed, (e.g. rotated) the transformation is directly applied to the holographic equivalent. This makes the fast generation of holographic views of objects from different observer positions possible. The rectangles are rendered translucent. Thus, lines in the background are also transferred to the hologram.
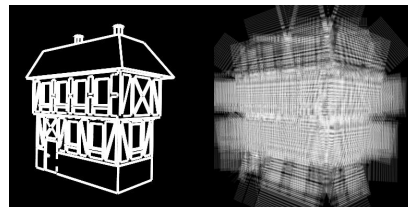

Figure 5: A house composed of line segments (left), translucent rendering of the textured rectangles in the holographic equivalent (right)

3 Extending the Approach to Splines

3.1 Imaging of Splines

The method introduced in Section 2.2 and described in detail in [14] is sufficient for objects consisting of straight line segments. There are two ways of extending our approach to be able to image curves in 3D:

1. As in the "line approach", each spline in the input object is represented by one element in the holographic equivalent. Simulate the holographic patterns for the imaging of curves by

morphing these elements.

2. Approximate splines by straight line segments and transfer these segments into elements of the holographic equivalent.

The first idea is based on the method for the holographic imaging of curves introduced in [5] which extends the conventional approach as described in [6] and [8]. Equation 1 (Fresnel approximation) is modified

$$u(x,y,0) = exp\left(i\frac{\pi}{\lambda}\frac{y^2}{R+f(x)}\right). \qquad (6)$$

R identifying the line distance in Equation 1 is augmented by a function $f(x)$ in Equation 6. This function describes the depth variation of the curve, making for instance a sinusoidal shape possible (see [5] for examples). Such a behavior can be simulated by morphing the elements of the holographic equivalent. Morphing is defined as a simultaneous interpolation of shape and texture [3]. However, a morphing procedure turns out to be time-consuming.

The second approach does not involve expensive operations like morphing and is therefore rather feasible. Approximations like this are common in computer graphics: at the lowest level, graphics hardware draws points, line segments and polygons (triangles, quadrilaterals). Smooth curves are drawn by approximating them with a more or less large number of line segments [12]. Another advantage is that the procedure of building the holographic equivalent shown in Section 2.2 and in [14] does not need to be changed in general. A spline is represented by a number of line segments. For each of these segments one element (textured rectangle) is built into the holographic equivalent. Figure 6 shows a simple example of a curve and its holographic equivalent.
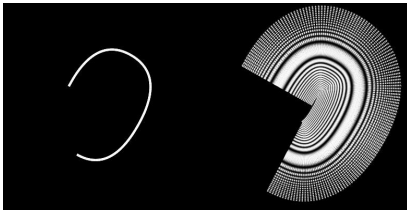

Figure 6: A spline curve approximated by straight lines (left), the holographic equivalent (right) consisting of one element per input line segment

3.2 Problems of Adapting the Line Method

Using the equivalent as described in Section 3.1 works, in principle. However, undesired diffraction effects occur at the rectangle borders. They are disregarded (see [6]) for objects consisting of a small number of line segments but splines are approximated by a relatively large number of these segments. The more line segments of which the original object consists, the more elements are included in the holographic equivalent.

It can be seen in Figure 6 that there are gaps between the rectangles of the holographic equivalent. Furthermore, an overlap of adjacent rectangles occurs. These effects are schematically illustrated in Figure 7. Sections 3.3 and 3.4 show solutions to this problem.
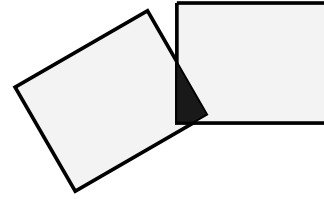

Figure 7: Two adjacent rectangles in the holographic equivalent showing a gap and overlap (marked dark)

3.3 Stretching Elements

In Figure 6 it can be seen that there is a high contrast between the gaps (in hologram background color) and the adjacent rectangles. This contrast is higher than the difference in brightness imposed between the regions of overlap and the non-overlapping part of the adjacent rectangles.

A first attempt at solving the "gap problem" is to stretch the elements (rectangles) of the holographic equivalent such that the gaps between adjacent rectangles are closed. Figure 8 illustrates the principle. For implementing the stretching, first the size of the
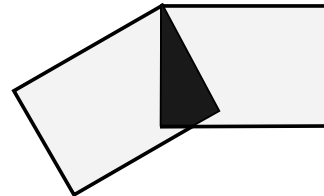

Figure 8: Stretching adjacent elements to close the gap

gap has to be determined. Then the adjacent rectangles are scaled such that the gap is closed. Figure 9 shows an example in which the gaps are closed by stretching adjacent elements.
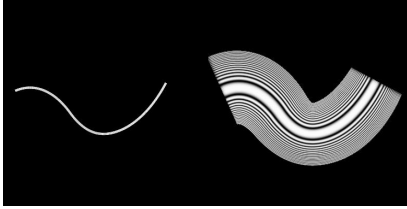
Figure 9: Curve (left) with elements stretched in the holographic equivalent (right)

Figure 11: Curve (left), holographic equivalent with stitching applied (right)

The gaps, as the areas of the highest contrast, can be removed with the solution demonstrated in this section. However, the other problem of overlapping adjacent rectangles is even made worse. Therefore we developed a second solution which is more suitable to our problem.

3.4 Stitching

The gaps and the areas of overlap have to be removed at the same time. This cannot be accomplished just by scaling the elements (rectangles) up or down. Instead, the coordinates of the rectangle vertices have to be altered. Figure 10 illustrates the idea (note the difference in the vertex coordinates compared to Figure 7). As stated in Section 2.2, the rectangles in
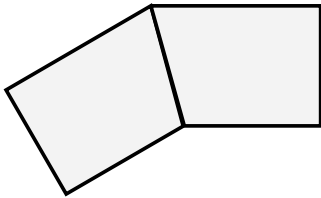

Figure 10: Stitching by altering vertex coordinates

the holographic equivalent are subdivided into strips to be able to implement a quadratic distortion function along the rectangle. Since the splines are approximated by a large number of short line segments, there is also a high number of short rectangles in the holographic equivalent. There is a sufficient number of elements representing the original curve. Therefore, the elements do not have to be subdivided into strips. Otherwise the modification of vertex coordinates would have to be performed for each strip of which the rectangle consists.

Figure 11 shows an example without gaps and overlap. Due to the missing overlap the holographic equivalent appears not as bright as the equivalent after stretching (compare Figure 9). However, the gaps can only be completely closed if antialiasing is ap-
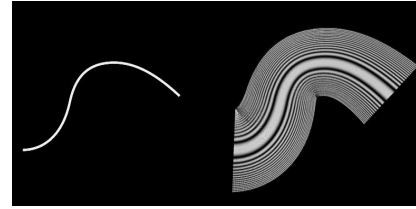
plied. Otherwise "holes" of pixel size remain which influence the hologram quality.

All the operations described in this section are necessary to ensure the desired reconstruction results. It has to be noted that the hologram rendering speed is decreased due to stitching and antialiasing. However, we consider that the improved hologram quality justifies this compromise.

4 Implementation

The approach presented in Sections 2 and 3 is implemented in our HoloRenderer system. The HoloRenderer is written in C++ on Silicon Graphics Workstations using the graphics libraries IRIS Performer [15] and OpenGL [12]. This environment was chosen because IRIS Performer as a high level graphics library allows the rapid development of computer graphics applications. It includes for instance facilities for loading and managing geometry data in several formats. State-of-the-art graphics workstations of the SGI family implement features like texture mapping and multisampling for antialiasing purposes in hardware which we exploit when rendering holographic equivalents to holograms.

The HoloRenderer has a modular structure. The *viewer module* loads the geometry data of the object to be imaged, builds the holographic equivalent, and displays both the object and its equivalent in a preview window (see Figures 9 and 11). The rectangle in each element of the equivalent is wrapped by a Performer billboard. Billboards are special Performer geometry nodes ensuring a certain orientation to the camera. Billboards are adapted to our purposes to cope with the orientation constraint (see Section 2.2 and also [14] for a detailed discussion). The *viewer* is furthermore responsible for rendering the hologram in tiles in order to assemble a hologram of high resolution (see [14] for details).

All functions necessary for overlaying the quadratic distortion function (see Section 2.2 and

[14]) are implemented in the *distortion module*. The *generator module* implements several other methods of hologram generation (see [6], [8], and [9]). Thus, we are able to compare the results gained with different approaches (for instance the conventional "line method", see Figure 2(b)). The *output module* takes the results of hologram rendering performed by the *viewer* or of the *generator's* computation and stores it as pixel data in a file (formats TIFF, BMP, XBM, Postscript). The next step is the image reconstruction which will be described in Section 5.2.

5 Results

5.1 Hologram Generation

Comparing the method shown in this paper to other approaches in terms of hologram generation speed reveals a couple of advantages of our approach. First, we use line segments instead of points as the basic primitives. This is more efficient for the holographic imaging of objects consisting of lines or curves as already shown in [6] and [8]. Second, the generation of the cylindrical wave pattern as the computationally most expensive step is performed in advance. Therefore, it is not included in the actual hologram generation, which is the major difference to the traditional "line approach" ([5], [6], [8]). Third, the ability to make use of graphics hardware is a decisive advantage of our approach to rendering holograms. The holographic rendering of complex objects (thousands of line segments) takes a couple of seconds for a high output resolution (10000×10000 pixels). The same hologram would take minutes if generated using the traditional method for lines.

5.2 Image Reconstruction

Reconstruction is necessary to display the image in its three-dimensional extent and to verify the holograms generated with our approach. Reconstruction can be performed by optical means. The hologram stored in a file is transferred onto photographic film which can be reconstructed in a special laser setup (see [13] for details). This procedure, however, is time-consuming. On the other hand, the process of diffraction can be simulated on a computer.

In order to simulate hologram reconstruction we use the *DigiOpt* system [1] which was developed at the University of Karlsruhe, Germany. The simulation just can be performed for distinct depths (distances to the hologram). There is no 3D sensation as in optical reconstruction but the results gained with this method are sufficient for a general verification of the hologram. All the examples in Section 5.3 were simulated with the DigiOpt system. The objects are rather simple because complex objects (with a large number of elements in the holographic equivalent) cannot be recognized in a simulation with DigiOpt which reads holograms of reduced resolution. Complex objects have to be optically reconstructed.

5.3 Examples

In this section we present some example holograms of reduced resolution together with the original object and the simulated reconstruction. Since all the
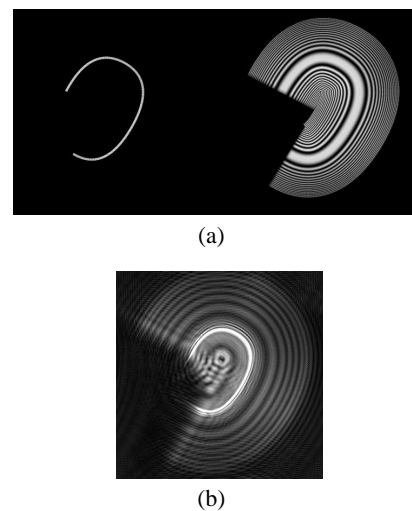

(a)

(b)

Figure 12: Spline seen from the top together with the equivalent (a), simulated reconstruction (b)

3D spline curves of which an input object consists are approximated by line segments, complex and simple objects are processed the same way when generating the holographic equivalent. The only difference is that the equivalent of a complex object contains more elements than that of a simple object. Therefore it is sufficient to show that the hologram generation works for single 3D spline curves. All other objects can be assembled of splines like for instance those shown in Figures 12, 13, 14, 15, and 16.

Figures 12 and 13 show the same object. It is a simple, almost circular 3D curve. In Figure 12(a) it is viewed from the top. The holographic equivalent of the curve is presented to the right of the original spline. The equivalent shows the same circular shape as the input curve. The reconstruction using the DigiOpt system results in a bright curve (Figure 12(b)). Since the original spline has no "depth", one simula-
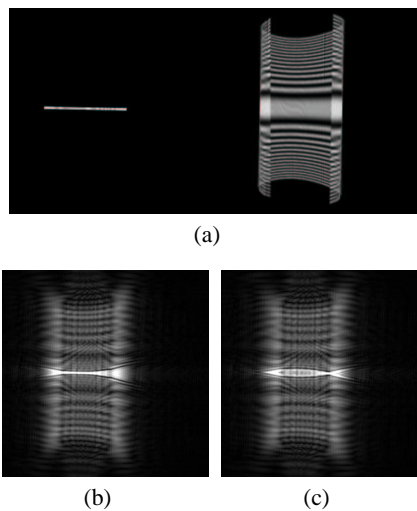
Figure 13: The same spline imaged from the side

tion run (simulated reconstruction performed for one distance to the hologram) is sufficient. The whole curve is "in focus".

The situation is different in Figure 13. Here the same spline is rotated and viewed from the side. In Figure 13(a) the curve appears as a horizontal line. The equivalent on the right shows again the circular shape, but this time viewed from the side. In contrast to Figure 12 the curve has a depth extent. Therefore the reconstructions show different foci. The reconstructions were performed for distinct depths. In Figure 13(b) the focus is in the background around the center of the spline. In Figure 13(c) the focus is shifted to the side. Two foci appear, one at each side.

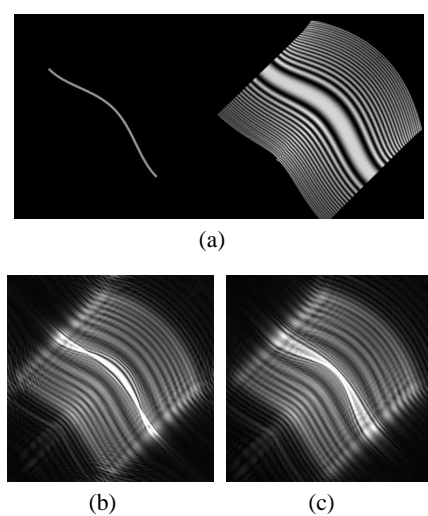An example of a curve of sinusoidal shape can be

Figure 14: A spline of sinusoidal shape

found in Figure 14. It shows that our approach is su-

perior to the method shown in [5], since we are able to image curves of arbitrary orientation in 3D space. The examples in [5] are restricted to one-dimensional functions (see Equation 6). The reconstruction in Figure 14(b) shows two foci, one on either side. In Figure 14(c) there is just one focus on the "top" of the curve.

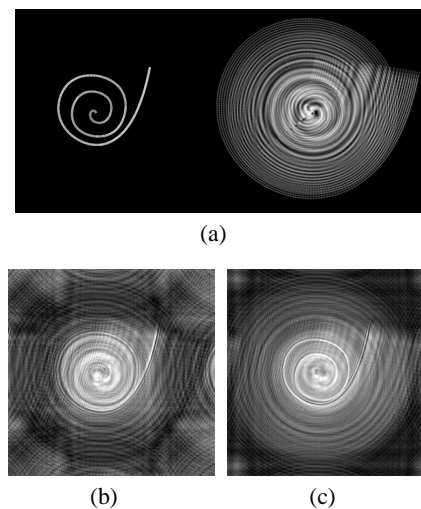The final example is that of a helix. In Figure 15 it

Figure 15: Helix viewed from the top

is viewed from the top. The point in the center of the helix is closer to the image plane than the end of the helix. In the reconstruction in Figure 15(b), the lower part (the end of the helix) appears in focus. The upper part is to be seen in Figure 15(c). In Figure 16,
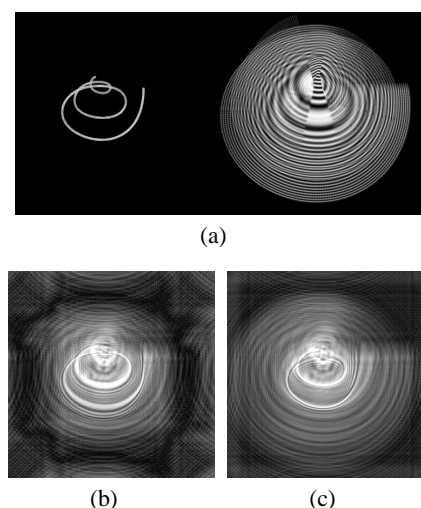

Figure 16: The helix from the side

similar to Figures 12 and 13, the helix is rotated. The reconstructions in Figures 16(b) and 16(c) show foci in the lower and upper part, respectively.

6 Summary

We successfully extended our approach for the holographic imaging of lines to curves. Our method is based on the mapping of an input object consisting of lines or curves to a second 3D object—the holographic equivalent. The equivalent consists of textured rectangles. In order to simulate holographic patterns induced by cylindrical and conical waves, the texture coordinates are altered depending on the line orientation. The hologram is generated by rendering the equivalent.

Splines to be imaged are approximated by short line segments. In the equivalent, one rectangle per line segment is generated. Undesired diffraction effects occuring at the element (rectangle) borders are eliminated by stretching and stitching adjacent elements. The holograms have been verified by means of optical reconstruction and computer simulation.

Future work includes an extension to holographic imaging of triangles, which allows the visualization of a much wider variety of objects than just those consisting of line segments and splines.

References

[1] H. Aagedal, T. 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 IIt*, Proc. SPIE 2404, pages 50–58, 1995.

[2] L. Bergmann and C. Schaefer. *Lehrbuch der Experimentellen Physik*, volume III, Optik. de Gruyter, Berlin, New York, 1993.

[3] S. C. Chen and L. Williams. View interpolation for image synthesis. In *Proceedings of SIGGRAPH '93*, pages 279–288, Anaheim, August 1993.

[4] C. Frère. *Verallgemeinerte Konfiguration der Rekonstruktions- und der Hologrammfläche in der digitalen Holografie*. PhD thesis, Fachbereich Physik der Universität-Gesamthochschule Essen, 1988.

[5] C. Frère and O. Bryngdahl. Computer-generated holograms: Reconstruction of curves in 3-D. *Opt. Comm. 60*, pages 369–372, 1986.

[6] 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)*, A3(5):726–730, 1986.

[7] D. Gabor. A new microscopic principle. *Nature*, 161(4098):777–778, May 1948.

[8] D. Leseberg and C. Frère. Free positioned and oriented focal lines from computer-generated holograms. *SPIE Proceedings*, 812:113–118, 1987.

[9] M. Lucente. Interactive computation of holograms using a look-up table. *Journal of Electronic Imaging*, 2(1):28–34, 1993.

[10] M. Lucente and T.A. Galyean. Rendering interactive holographic images. In *Proceedings of SIGGRAPH '95*, pages 387–394, Los Angeles, August 1995.

[11] M. McKenna and D. Zeltzer. Three dimensional display systems for virtual environments. *Presence: Teleoperators and Virtual Environments*, 1(4):421–458, 1992.

[12] J. Neider, T. Davis, and M. Woo. *OpenGL Programming Guide: the Official Guide to Learning OpenGL*. Addison-Wesley, Reading, MA, 1993.

[13] A. Ritter and T. Benziger. Synthetische Holografie und Computergrafik. In O. Deussen and P. Lorenz, editors, *Simulation und Animation '97*, pages 76–86, Magdeburg, March 1997. SCS – Society for Computer Simulation Int.

[14] A. Ritter, O. Deussen, H. Wagener, and T. Strothotte. Holographic imaging of lines: a texture based approach. In *International Conference on Information Visualization IV'97*, pages 272–278. IEEE Computer Society, 1997.

[15] J. Rohlf and J. Helman. IRIS Performer: A high performance multiprocessing toolkit for real-time 3D graphics. In *Proceedings of SIGGRAPH '94*, pages 381–394, Orlando, August 1994.

[16] D. Schreier. *Synthetische Holografie*. Physik-Verlag, Weinheim, 1984.

[17] T. Strothotte, B. Preim, A. Raab, J. Schumann, and D.R. Forsey. How to render frames and influence people. *Computer Graphics Forum*, 13(3):455–466, 1994.