Hologram synthesis for photorealistic reconstruction

Martin Janda,[1,*] Ivo Hanák,[1] and Levent Onural[2]

[1]*Department of Computer Science and Engineering, University of West Bohemia, Univerzitní 22, Plzeň, Czech Republic*

[2]*Department of Electrical and Electronics Engineering, Bilkent University, Ankara TR-06800, Turkey*

[*]*Corresponding author: mjanda@kiv.zcu.cz*

Received June 25, 2008; accepted September 21, 2008;
posted October 22, 2008 (Doc. ID 97742); published November 24, 2008

Computation of diffraction patterns, and thus holograms, of scenes with photorealistic properties is a highly complicated and demanding process. An algorithm, based primarily on computer graphics methods, for computing full-parallax diffraction patterns of complicated surfaces with realistic texture and reflectivity properties is proposed and tested. The algorithm is implemented on single-CPU, multiple-CPU and GPU platforms. An alternative algorithm, which implements reduced occlusion diffraction patterns for much faster but somewhat lower quality results, is also developed and tested. The algorithms allow GPU-aided calculations and easy parallelization. Both numerical and optical reconstructions are conducted. The results indicate that the presented algorithms compute diffraction patterns that provide successful photorealistic reconstructions; the computation times are acceptable especially on the GPU implementations. © 2008 Optical Society of America

OCIS codes: 090.1760, 090.1995.

1. INTRODUCTION

Holography is a method to capture the physical nature of light, usually on a planar surface (a hologram), so that when the plate is later illuminated in a proper manner, the originally recorded light is created in a volume [1,2]. If the reconstructed volume filling light is exactly the same as the originally recorded light, an observer will see the original environment, with all of its features including the three-dimensional (3D) nature of the scene, when looking into the reconstructed light. The fidelity of the reconstructed light is directly related to the 3D reconstruction quality. The major difference between holography and photography is the inability of the latter to record properties of light other than its intensity.

Computation of the hologram pattern due to a given object is the primary goal of computer-generated holography, which has a long history [3–6]. Various methods for different situations are reported [3,7–9]. Naturally, there are two major concerns in computer-generated holography: the quality of the eventual optical reconstruction from these holograms and the speed of computations; the latter is especially important in holographic television applications, where a real-time computation at the frame rate is targeted. Fast computation methods to generate the desired holograms are investigated and reported [10–12], and hardware solutions are employed to increase the speed of computations [13,14].

There are numerous methods that are quite fast for computing holograms of planar objects, including a parallel object and image planes as well as slanted planes [15–19]. However, the benefits of holography show themselves best when the original is a 3D object. Therefore, the fast computation of holograms of not only planar [two-dimensional (2D)] objects but especially 3D objects is of prime interest.

Digital simulation of optical diffraction and holography invariably starts with discretization of the problem. That necessitates understanding the effects of sampling and quantization. Such effects are much more complicated and interesting for diffraction and holographic patterns than for more common applications, as analyzed in [20–22].

In this paper, we present a complete discrete computational procedure that yields a planar discrete hologram of a given 3D object (or a 3D environment). The object is described in an abstract manner, as usual in computer graphics, in the form of a 3D geometric structure whose surfaces possess realistic texture and optical reflection properties. Naturally, the problem is similar, in many aspects, to the classical rendering problem in computer graphics [23–25]; however, the holography case involves coherent light and does not possess classical camera with a lens. The simulation of coherent light requires the incorporation of the phase into the calculations. Rendering without a classical camera model requires accumulation of light incident from different directions at the points on the hologram plane.

Our computational approach is fundamentally different from the approaches outlined above since it can handle complicated realistic surface structures with occlusions and realistic light reflectivity properties. Furthermore, the proposed discretization scheme allows easier implementations in different computational environments such as graphics processing units (GPUs) or parallel computers in addition to implementations in classical central processing unit- (CPU-) based architectures. Our

macrogeometric-structure model is a classical triangular mesh; however, we are able to accommodate a very large number of triangles to achieve realistic shapes. Furthermore, we employ a quite complicated surface reflectivity model to render holograms that then provide realistic 3D image reconstructions. A complete mathematical model adopted for diffraction and subsequent holographic recording is presented in Section 2. As commonly utilized in the literature, we also employ the so-called source model in the computations; this is an approximation, much more efficient than the accurate field model, and it provides quite good results for most 3D object shapes [11,26–29]. The discretization of the continuous model is given in Section 3. We employ a photorealistic surface complexity that consists of dense triangular meshes; furthermore, we impose additional restrictions on the surface geometry, as described in Section 4. The details of the algorithm to achieve an efficient computation is presented in Section 5. Implementations on distributed architectures and on GPUs for improved speed are given in Sections 6 and 9, respectively. Fresnel approximation together with its justification is given in Section 7, followed by an associated acceleration method based on a stored and properly zoomed Fresnel kernel given in Section 8.

In addition to numerical reconstructions, we tested our generated holograms also by optical reconstructions using state-of-the-art spatial light modulators (SLMs) [30]. Both numerical and optical results are presented in Section 10. Finally, conclusions are drawn in Section 11.

2. BASIC MODEL FOR HOLOGRAPHIC SETUP

The basic mathematical formulation that we adopt for diffraction and hologram formation due to a 3D object or a scene is presented in this section.

We consider the geometry shown in Fig. 1. Here we have a surface S that radiates light with a directional amplitude distribution $A(\mathbf{s},\hat{\mathbf{r}})$, where $\mathbf{s}$ represents the coordinates of a point on S, and $\hat{\mathbf{r}}$ is the unit vector indicating the reverse direction of radiation from $\mathbf{s}$; $A(\mathbf{s},\hat{\mathbf{r}})$ is a complex-valued function. The intensity $|A(\mathbf{s},\hat{\mathbf{r}})|^2$ is the light power emanating from the surface per unit area per unit solid angle at the point $\mathbf{s}$ on S. We have the hologram plane H, where we want to compute the optical field due to the monochromatic coherent illumination from the surface S. We chose the hologram plane H as the $z=0$ plane.

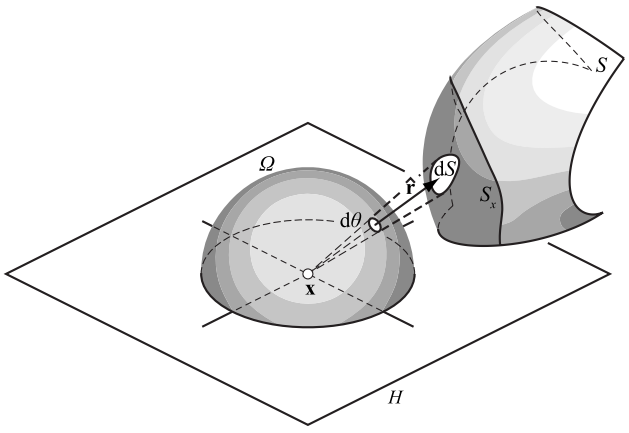


Fig. 1. Relation between a differential solid angle $\mathrm{d}\theta$ and a differential surface element $\mathrm{d}S$. The line crossing the surface S delimits the surface $S_{\mathbf{x}}$ visible from $\mathbf{x}$ and the rest of the surface S.

When we consider monochromatic coherent light propagation from the surface S, the field $u(\mathbf{x})$ at an arbitrary point $\mathbf{x}\in H$ is modeled as

$$u(\mathbf{x}) = c\int_{S_{\mathbf{x}}} A(\mathbf{s},\hat{\mathbf{r}})\frac{\exp(\mathrm{i}kr)}{r}(\hat{\mathbf{n}}_H\cdot\hat{\mathbf{r}})\mathrm{d}S,$$

$$\mathbf{r}=\mathbf{s}-\mathbf{x},\quad r=|\mathbf{r}|,\qquad \hat{\mathbf{r}}=\mathbf{r}/r, \tag{1}$$

where $\hat{\mathbf{n}}_H$ is the unit normal vector to the hologram plane H, $k=2\pi/\lambda$, and λ is the wavelength. The range of integration $S_{\mathbf{x}}$ represents the segments of S that are not occluded when looking from the point $\mathbf{x}$. For convenience, we will drop the complex constant c for the rest of the analysis.

The model adopted above is rather different from conventional hologram representations; however, it is more appropriate for our purposes since it associates the hologram formation process with computer graphics concepts very conveniently. The directional amplitude distribution $A(\mathbf{s},\hat{\mathbf{r}})$ is directly related to local 2D Fourier transform of the local complex texture (amplitude) on S at $\mathbf{s}$, and therefore, the integral gives a 2D space-frequency representation for $u(\mathbf{x})$. Local frequencies at the location $\mathbf{s}$ are directly associated with propagation directions of local plane waves propagating away from $\mathbf{s}$, and the associated complex amplitudes corresponding to those frequencies determine the amplitudes and phase variations of these waves. Therefore, the model violates rigorous mathematical foundations since we need a surface patch, and not a surface point, to describe the local frequency content. We can also state that a point source should have a uniform directional radiation pattern; that violates the nonuniform possibilities implied by the adopted model. Finally, the amplitude decay by $1/r$ is associated with a point source, and other directional radiation patterns as adopted may violate that, as well. A rigorous analysis based on local directional radiation pattern variations, and therefore local frequency content of the surface pattern, inevitably brings the associated uncertainity principle into the model. Such a rigorous analysis is beyond the scope and the purpose of this paper. Instead, in accordance with our goal of bringing widely used computer graphics methods to optical field and hologram computation, we adopt a diffraction model as given by Eq. (1), which is an approximation. However, this model is a quite good approximation that can be successfully used to model frequently encountered real-life situations, and thus it is fully adequate for our purposes.

We convert the surface integral above in Eq. (1) via a change of variables to an integration over the solid angle θ by observing that

$$\mathrm{d}S = \frac{r^2\,\mathrm{d}\theta}{|\cos\varphi|},\qquad \cos\varphi=\hat{\mathbf{n}}_{\mathbf{s}}\cdot\hat{\mathbf{r}}, \tag{2}$$

where $\hat{\mathbf{n}}_{\mathbf{s}}$ is the unit normal to the surface S at the point

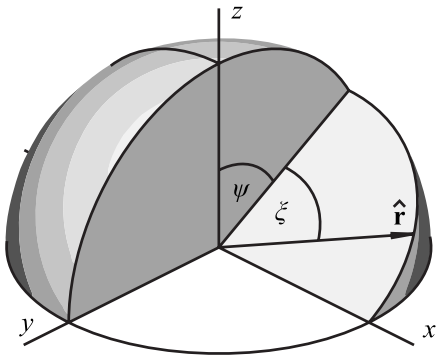


Fig. 2. Parameterization of a ray direction $\hat{\mathbf{r}}$ by two angles ξ and ψ.

$\mathbf{s}$. Therefore, the complex valued optical field values $u(\mathbf{x})$ over the hologram plane H become

$$u(\mathbf{x}) = \int_{\Omega} A(\mathbf{s},\hat{\mathbf{r}}) \frac{\exp(\mathrm{i}kr)}{r} (\hat{\mathbf{n}}_H \cdot \hat{\mathbf{r}}) \frac{r^2}{|\cos\varphi|} \mathrm{d}\theta, \tag{3}$$

where the integration is over the hemisphere Ω.

For convenience, we define a new function $A'(\mathbf{s},\hat{\mathbf{r}})$ as

$$A'(\mathbf{s},\hat{\mathbf{r}}) = A(\mathbf{s},\hat{\mathbf{r}}) \frac{r^2}{|\cos\varphi|}. \tag{4}$$

In order to evaluate the integral over the solid angle, we further parameterize it in terms of two angles ξ and ψ depicted in Fig. 2. Accordingly, the differential solid angle $\mathrm{d}\theta$ becomes

$$\mathrm{d}\theta = \cos\xi \mathrm{d}\xi \mathrm{d}\psi, \tag{5}$$

and therefore the optical field over H takes the form

$$u(\mathbf{x}) = \int_{-\pi/2}^{\pi/2} \int_{-\pi/2}^{\pi/2} A'(\mathbf{s},\hat{\mathbf{r}}) \frac{\exp(\mathrm{i}kr)}{r} (\hat{\mathbf{n}}_H \cdot \hat{\mathbf{r}}) \cos\xi \mathrm{d}\xi \mathrm{d}\psi, \tag{6}$$

where both $\mathbf{s}$ and $\hat{\mathbf{r}}$ are functions of $(\mathbf{x},\xi,\psi)$. Equation (6) above states the optical field at a point $\mathbf{x}$ on H as a superposition of field contributions received from infinitesimal solid angles associated with different directions $\hat{\mathbf{r}}$ by considering the change in phase and amplitude as a consequence of the distance between the surface element and the optical field point as well as the attenuation due to obliqueness of the incident ray on the hologram plane H from those directions.

Note that forming a hologram is straightforward once the optical field on H due to S is found; e.g., a reference beam can be added, and the intensity of the subsequent field is recorded as the hologram [15].

We will discretize the result given by Eq. (6) in the next section to obtain a form that is suitable for digital processing.

3. DISCRETIZATION

Discretization of Eq. (6) is essential for subsequent digital processing. We start by discretizing the angles ψ and ξ as

$$\psi_l = lD_\psi, \qquad \xi_m = mD_\xi, \tag{7}$$

where l and m are integers and D_ψ and D_ξ are the discretization steps in radians. We also define a regular rectangular grid of discrete positions $\mathbf{x}_{pq}$ on the hologram plane H as

$$\mathbf{x}_{pq} = (pD_x, qD_y, 0) \in H, \tag{8}$$

where p and q are integers and D_x and D_y are the discretization steps in meters. Therefore, we label the optical field value at a position $\mathbf{x}_{pq}$ as u_{pq}. Thus we get the discrete form of Eq. (6) as

$$u_{pq} = \sum_l \sum_m A_{pqlm} \frac{\exp[\mathrm{i}kr_{pqlm}]}{r_{pqlm}} w_{pqlm} \cos\xi_m, \tag{9}$$

where we defined the discrete amplitude variable $A_{pqlm} = A'(\mathbf{s},\hat{\mathbf{r}}) D_\xi D_\psi$ and the attenuation factor $w_{pqlm} = \hat{\mathbf{n}}_H \cdot \hat{\mathbf{r}}$. Note that $\mathbf{s}$ and $\hat{\mathbf{r}}$ are functions of $(\mathbf{x}_{pq},\psi_l,\xi_m)$ and therefore the amplitude, distance, and attenuation factor are all functions of the indices p, q, l, and m.

We assume that the sampling over the hologram plane H using the sampling intervals D_x and D_y does not result in aliasing; this in turn imposes a band-limited $u(\mathbf{x})$ with a spatial radian frequency bandwidth of

$$-\frac{\pi}{D_x} < k_x < \frac{\pi}{D_x}, \qquad -\frac{\pi}{D_y} < k_y < \frac{\pi}{D_y}, \tag{10}$$

where k_x and k_y are the frequency variables along the x and y directions with units in radians per unit length. This bandwidth restriction imposes a limitation on the incidence angles of light rays arriving at the hologram plane H from the source surface S, because a plane wave propagating along direction $\mathbf{k}$ is a 3D Fourier basis function $\exp(\mathrm{i}\mathbf{k}\cdot\mathbf{x})$, whose spatial frequencies along the x, y, and z directions are given by k_x, k_y, and k_z, where $(k_x,k_y,k_z) = \mathbf{k}$. For monochromatic light $|\mathbf{k}| = k = 2\pi/\lambda$, where λ is the wavelength of light. Once k_x and k_y are fixed for a given wavelength λ, $k_z = (k^2 - k_x^2 - k_y^2)^{1/2}$ is also fixed and therefore the direction of propagation is also fixed as $\mathbf{k} = (k_x,k_y,k_z)$. As a result, for a given D_x and D_y and the bandwidth as in Eq. (10) to avoid aliasing, we can get the maximum incidence angles as

$$\Xi_d = \arcsin\left(\frac{k_{x_{\max}}}{k}\right) = \arcsin\left(\frac{\lambda}{2D_x}\right),$$

$$\Psi_d = \arcsin\left(\frac{k_{y_{\max}}}{k}\right) = \arcsin\left(\frac{\lambda}{2D_y}\right), \tag{11}$$

where $k_{x_{\max}} = \pi/D_x$ and $k_{y_{\max}} = \pi/D_y$ [see Eq. (10)]. This observation further limits the range of l and m indices in Eq. (7) and therefore improves the computational efficiency. Note that since the propagation direction of a wave does not change (free-space propagation) as it propagates away (or toward) H, the frequency content, and therefore the bandwidth, will be the same over any hypothetical surface parallel to H. However, in the case of a slanted plane, the same 3D propagation direction will impose other 2D frequency components on the complex field on the plane; the same angle limitations will then impose

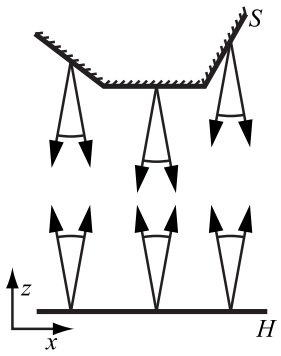


Fig. 3. The propagation angle restrictions associated with aliasing-free sampling of the optical field on H due to D_x (lower cones) determine the propagation angle restrictions associated with the surface S (upper cones). The same restrictions are also in place for the y direction due to D_y.

band-limited patterns with different center frequencies over slanted planes [18]. Therefore, for arbitrary S, the corresponding restriction on the surface texture owing to limitations imposed on the pattern over H can be summarized as locally band-limited functions whose local center frequencies are related to the local slope of S. (See Fig. 3.) If the slope is zero (parallel to H) the center frequency is zero (low-pass signal), and as the slope increases, the center frequency also increases. All these statements should be interpreted in the sense of space-frequency representations, in line with the model adopted in the previous section, and therefore the associated uncertainity principle is in effect.

It is possible to impose further restrictions on the range of l and m in Eq. (7) based on the spatial extent of a finite $S_{\mathbf{x}_{pq}}$. Considering the bounding box around $S_{\mathbf{x}_{pq}}$, we see that the range of angles is restricted to $[\Psi_b,\Psi_B]$ and $[\Xi_b,\Xi_B]$. (See Fig. 4.)

Therefore, combining all restrictions above, we fix the range of l and m as

$$l \in [L_{\min},L_{\max}] : \psi_l \in [-\Psi_d,\Psi_d] \cap [\Psi_b,\Psi_B],$$

$$m \in [M_{\min},M_{\max}] : \xi_m \in [-\Xi_d,\Xi_d] \cap [\Xi_b,\Xi_B]. \quad (12)$$

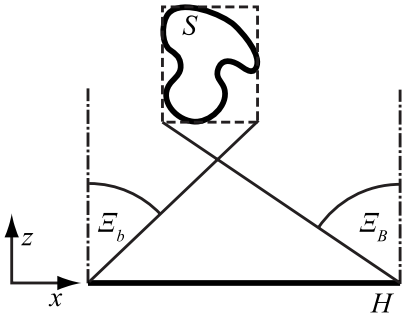


Fig. 4. Range $[\Xi_b,\Xi_B]$ inferred from the bounding box of the surface S. The situation for $[\Psi_b,\Psi_B]$ is similar.

4. RESTRICTIONS ON THE SURFACE GEOMETRY

In this section we will describe the restrictions we choose to impose on the surface geometry S for both analytical and computational convenience.

As we stated in Section 1, our goal is to compute holograms for subsequent photorealistic reconstructions. The formulation presented in Section 2 is readily applicable to represent realistic texture reflectance properties since we adopted a space-frequency based description that allows spreading of radiated light within a solid angle. In addition, we target a realistic geometry that would allow the level of detail we expect in a realistic case. To this end, we may choose a triangular mesh that is commonly used in computer graphics, with a large number of triangles. However, as described in Section 3, the adopted aliasing-free sampling strategy imposes restrictions on the complex-valued texture A_{pqlm} over S. Even though it is possible to satisfy these restrictions, the management of such restrictions will be cumbersome and thus a computational burden. Therefore, we choose to restrict the geometry of S further, in such a way that the management of the complex-valued-texture restrictions becomes easier.

To this end, we restrict S to consist only of planar segments that are always parallel to H. However, the segments are discontinuous, since each one may have a different distance from H. Such a geometry will ensure that the texture on S will be a slowly varying (low-frequency content) signal as described in Section 3. It is not difficult to implement such restrictions on the texture. However, the discontinuities between the segments could be problematic since they result in phase discontinuities, and that in turn would generate high-frequency components on the texture; this would then violate the restrictions described in Section 3. Therefore, we restrict the discontinuities to be integer multiples of the monochromatic light wavelength λ. Actually, for convenience, we restrict the distance z from H to a point $\mathbf{s}$ on S to be always an integer multiple of λ. Such a restriction does not create any visual degradation since the introduced step size λ is 0.4–0.6 μm for visible light, whereas the practical distances between the hologram and the object surface is of the order of tens of centimeters in most applications.

The restriction outlined above eases the overall computational burden while conveniently ensuring the conditions associated with discretization. However, in order to keep the benefits of working with triangular meshes and still impose the outlined stepwise-discontinuous surface, we adopt a two-step approach. First, we start with a triangular mesh S' to represent S. Then, to get the actual S from S' we quantize the distance from H to be an integer multiple of λ so that the deviation from the triangular mesh is minimum. (See Fig. 5.)

5. EFFICIENT COMPUTATION

The basic model for the continuous case and the subsequent discretization issues, as well as the restrictions on the surface geometry, have already been discussed in previous sections. Therefore, we assume that now the problem is the actual implementation of the relation given by Eq. (9).

One of the issues during the computation is the accuracy of the distances r_{pqlm} in Eq. (9) since the distance affects the phase and the hologram computations are sensitive to phase errors. The texture on S that is represented by A_{pqlm} in Eq. (9) and the geometry S are input data. We assume they both comply with the restrictions imposed on them as described in Sections 3 and 4. As a consequence of the already discussed restrictions on the signal $u(\mathbf{x})$ and the surface S, it is possible to further restrict A_{pqlm} to be a real-valued function. This in turn restricts the angular propagation pattern to be symmetric around its center. (See Fig. 3.) Effects of such an additional restriction on the variety of surface textures is not important for our purposes. Computation of other factors in Eq. (9) is rather straightforward. Therefore, we emphasize only the accurate computation of the distances, which also includes the accurate computation of the intersection points of rays with S.

Another primary issue is the efficiency of the computations and their suitability for specific implementations, such as an implementation using a commercially available GPU. The actual order of the summations in Eq. (9) and the exploitation of the associated parallelism in computation are important in that regard. In this section we describe computational procedures that address both of the issues outlined above.

An important variable of Eq. (9) is the distance to the intersection of a ray and a surface that is nearest to the starting point of the ray on H. This is known as the *ray casting* method [24,31]. The evaluation of the distance r_{pqlm} between the point $\mathbf{x}_{pq}$ on H and the intersection $\mathbf{s}_{pqlm}$ on S is repeated many times, and therefore it is computationally demanding. We describe an efficient way to find the nearest intersection and then compute the distance r_{pqlm} for the surface S that complies with the restrictions imposed on it as described in Section 4.

The ray from $\mathbf{x}_{pq}$ toward S is

$$R_{pqlm} = \{\mathbf{x} : \mathbf{x} = \mathbf{x}_{pq} + r\hat{\mathbf{r}}_{lm}\}. \tag{13}$$

As a consequence, the intersection point $\mathbf{s}_{pqlm} = R_{pqlm} \cap S_{\mathbf{x}_{pq}}$ corresponds to the point of R_{pqlm} with a parameter $r = r_{pqlm}$.

As we outlined in Section 4, it is convenient to describe the surface S in two steps: first as a triangular mesh S' and then conversion of mesh structure to a discontinuous surface S whose segments are parallel to H. (See Fig. 5.) Therefore, we start with a triangular mesh $G = (\mathcal{V}, \mathcal{T})$ as our surface S', which consists of a set of vertices $\mathcal{V}$ and a set of triangles $\mathcal{T}$. A triangle $t_{ABC} \in \mathcal{T}$ specifies a triangular section of a plane between vertices $\mathbf{v}_A$, $\mathbf{v}_B$, and $\mathbf{v}_C$. Normal of the plane is $\mathbf{n}_{ABC} = (\mathbf{v}_B - \mathbf{v}_A) \times (\mathbf{v}_C - \mathbf{v}_A)$. Our definition of the mesh G reflects a data structure that is commonly used in computer graphics [25].

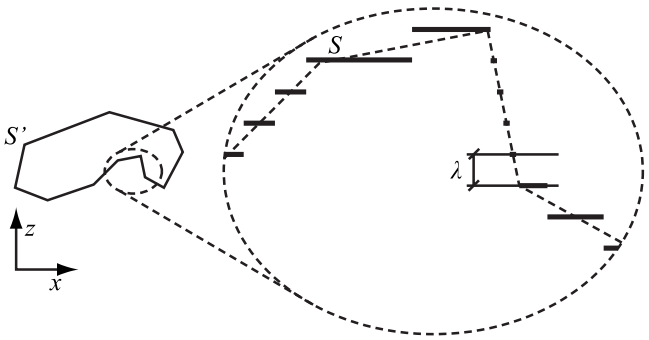


Fig. 5. Cross section of a surface S' and its decomposition into a stepwise surface S.

In the case of multiple intersections of a ray with the mesh, we need to find the one nearest to $\mathbf{x}_{pq}$. If the set of intersection points is empty, then we assume that $r_{pqlm} = \infty$.

As the number of triangles is expected to be high, testing each triangle for an intersection might significantly reduce the overall performance. However, we can detect triangles that are completely invisible and omit them. For us a triangle t_{ABC} is completely invisible from a point $\mathbf{x}_{pq}$ if $(\mathbf{x}_{pq} \cdot \hat{\mathbf{n}}_{ABC} - \mathbf{v}_A \cdot \hat{\mathbf{n}}_{ABC}) \leqslant 0$. We denote such triangles as back facing for $\mathbf{x}_{pq}$.

Furthermore, we observe that in Eq. (9) for $l = l_c$, all rays R_{pql_cm} are coplanar with a plane ρ_{ql_c}. Considering the left-handed coordinate system, the plane ρ_{ql_c} is

$$\rho_{ql_c} : (qD_y - y)\cos\psi_{l_c} + z\sin\psi_{l_c} = 0. \tag{14}$$

This observation leads to efficient evaluation of the intersection points by adopting a two-step algorithm that is based on slicing the surface. We start by intersecting the mesh G with the plane ρ_{ql_c}, and we obtain a slice $W_{ql_c} = (\mathcal{V}_{ql_c}, \mathcal{E}_{ql_c})$, where $\mathcal{V}_{ql_c}$ is a set of vertices and $\mathcal{E}_{ql_c} = \{E_{AB} : E_{AB} \in \mathcal{T} \cap \rho_{ql_c}, \mathbf{v}_A \in \mathcal{V}_{ql_c}, \mathbf{v}_B \in \mathcal{V}_{ql_c}\}$ is a set of edges. An edge E_{AB} is

$$E_{AB} = \{\mathbf{x} : \mathbf{x} = \mathbf{v}_A + e(\mathbf{v}_B - \mathbf{v}_A), e \in [0,1]\}, \tag{15}$$

where the vertex $\mathbf{v}_A$ is the beginning of the edge and the vertex $\mathbf{v}_B$ is the end of the edge. Considering only the slice W_{ql_c} reduces the complexity of intersection search significantly. Let us consider the simplest case where $l_c = 0$ and therefore ρ_{q0} is parallel with the plane $\eta : y = 0$. This special case is of great significance because each case where $l \neq 0$ can be converted by a geometric transformation to a case where $l = 0$. The transformation is described later in this section.

Now that we have a set of edges $\mathcal{E}_{q0}$ and a ray R_{pq0m}, the goal is to find $\mathcal{E}_{q0} \cap R_{pq0m}$. According to Eqs. (13) and (15) the ray parameter r and the edge parameter e, corresponding to the intersection point are calculated from

$$\mathbf{x}_{pq} + r\hat{\mathbf{r}}_{0m} = \mathbf{v}_A + e(\mathbf{v}_B - \mathbf{v}_A). \tag{16}$$

When reorganized and assuming $\hat{\mathbf{r}}_{0m} = (x_{r_{0m}}, y_{r_{0m}}, z_{r_{0m}})$, Eq. (16) becomes a set of two linear equations that can be easily solved for r and e. Since edges are finite line segments, the solution is valid only if $e \in [0,1]$. Otherwise the edge is not considered to be intersected.

When searching for the intersections, it is not necessary to test all edges $\mathcal{E}_{q0}$ for each m. Since m is ascending, the angle ξ_m is also ascending due to Eq. (7). We use an easy framework to quickly select for each m only those edges that are intersected by the ray R_{pq0m}. For each vertex $\mathbf{v}_i \in \mathcal{V}_{q0}$ and for a fixed q we define an angle

$$\alpha_{ip} = \arctan\left(\frac{x_{pq} - x_{\mathbf{v}_i}}{z_{\mathbf{v}_i}}\right), \tag{17}$$

and we sort edges $E_{AB} \in \mathcal{E}_{q0}$ ascendantly by α_{Bp}. All edges are oriented so that $\alpha_{Ap} > \alpha_{Bp}$. Then, as long as $\xi_m \in [\alpha_{Bp}, \alpha_{Ap}]$, the edge E_{AB} is intersected by the ray R_{pq0m}.

Further, let $\mathcal{Q} = \{E^i_{AB}\}$, $\mathcal{Q} \subset \mathcal{E}_{q0}$ be a set of active edges, i.e., edges that are known to intersect with the specific ray R_{pq0m}. Every time m is increased, edges E_{AB}, $\alpha_{Bp} < \xi_m$ are added to $\mathcal{Q}$ while edges E_{AB}, $\alpha_{Ap} < \xi_m$ are removed from $\mathcal{Q}$. Since edges $E_{AB} \in \mathcal{E}_{q0}$ are sorted by α_{Bp} the evaluation of the condition $\alpha_{Bp} < \xi_m$ is usually performed only once per edge. We can further improve the algorithm efficiency by employing proper sorting algorithms at different stages. Observing that for each edge E_{AB} the angles α_{Ap} and α_{Bp} corresponding to samples u_{pq} and u_{p+1q} differ only a little and therefore the order of edges do not change much, we use the Bubble Sort algorithm [32] for reordering the edges according to the new values of α_{Bp+1} since it is known that this algorithm is efficient for sorting partially sorted data. Nevertheless, for the first sample in a row, the edges are considered irregular and therefore the Quick Sort algorithm [32] is used in this case.

Up to now, we have described the evaluation of a single row for a fixed q and fixed l. The rest of the rows for different q are processed in the same fashion. When we move from the row q to the row $q-1$, we can exploit the fact that l is kept constant and rows are spaced uniformly. Because of this an efficient triangle scan-line conversion algorithm [33] can be employed to obtain a slice $W_{q-1,l}$ from a slice W_{ql}. Thus we complete the computation of all intersection points for fixed l.

To find all intersection points, we repeat the above procedure for all $l \in [L_{\min}, L_{\max}]$. Since computing the slice W_{q0} is more convenient than computing arbitrary slice W_{ql}, we proceed with a transformed mesh $G_{\mathcal{M}_l} = \mathcal{M}_l\{G\}$ such that the slice $W_{ql} = G \cap \rho_{ql}$ equals the slice $W_{q0} = G_{\mathcal{M}_l} \cap \rho_{q0}$; see Fig. 6.

The transformation operator $\mathcal{M}_l$ skews the geometry $G = (\mathcal{V}, \mathcal{T})$ by modifying vertices $\mathbf{v} \in \mathcal{V}$ in the direction of the y axis accordingly, i.e. $G_{\mathcal{M}_l} = (\mathcal{M}_l\{\mathcal{V}\}, \mathcal{T})$. If $\mathbf{v} \in \mathcal{V}$ then $\mathbf{v}_{\mathcal{M}_l} \in \mathcal{M}_l\{\mathcal{V}\}$ is

$$x_{\mathbf{v}_{\mathcal{M}_l}} = x_{\mathbf{v}},$$

$$y_{\mathbf{v}_{\mathcal{M}_l}} = y_{\mathbf{v}} - z_{\mathbf{v}} \tan \psi_l,$$

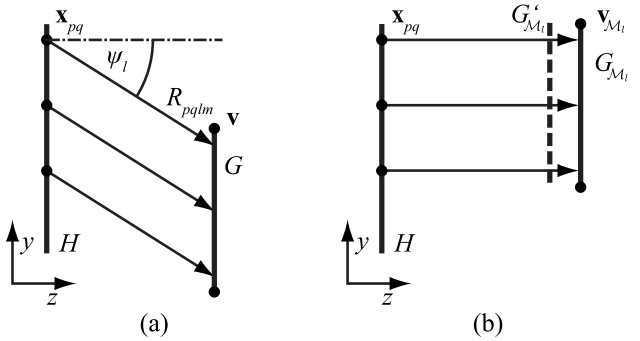


Fig. 6. (a) A mesh G and (b) its modified version $G_{\mathcal{M}_l}$ transformed by Eq. (18). The dashed mesh $G'_{\mathcal{M}_l}$ is the skewed mesh G without correction on the distance $z_{\mathbf{v}}$.

$$z_{\mathbf{v}_{\mathcal{M}_l}} = z_{\mathbf{v}} \frac{1}{\cos \psi_l}. \tag{18}$$

Finally, we modify each intersection point and the corresponding distance by moving the already computed intersection with the triangular mesh S' to the nearest intersection with the stepwise-discontinuous S. We move the intersection along the ray R_{pqlm}. As a consequence, r_{pqlm} becomes an integer multiple of λ'_{lm}, where

$$\lambda'_{lm} = \frac{\lambda}{\hat{\mathbf{n}}_H \cdot \hat{\mathbf{r}}_{lm}} \tag{19}$$

and $\hat{\mathbf{n}}_H = (0,0,1)$ is the normal to the plane H. When r_{pqlm} is rounded to the nearest integer multiple of λ'_{lm}, it is ensured that the corresponding distance to $\mathbf{s}_{pqlm}$ on S' is rounded to the nearest integer multiple of λ to imply an S as we described in Section 4.

Having the geometry of S, the intersection points of our rays with S together with the associated distances, and the texture A_{pqlm} on S, we are ready to complete the evaluation of Eq. (9). The algorithm actually follows the steps for the computation of the intersection points we have outlined above: as we evaluate the intersection points and the corresponding distances in the presented order, we immediately compute the associated partial field. First, we compute the partial result u^l_{pq} for all p and q but for fixed l and then get u_{pq} of Eq. (9) as $u_{pq} = \Sigma_l u^l_{pq}$. Not that each u^l_{pq} can be interpreted as a "horizontal parallax only" component of u_{pq} [34]. Thus we achieve an efficient and suitable algorithm, which we call Alg. 1, for evaluating the complete discrete optical field value u_{pq}; naturally, the discrete variables p and q run over a specified finite range. However, there are opportunities for further acceleration, as presented in the following section.

Algorithm 1. Skeleton of the algorithm for diffraction pattern computation. See the referred sections in the text for details.

```
for l = K_min to L_max do                                   ▷ Sec. 3
  Transform mesh G to mesh G_Ml                             ▷ Sec. 5
  for q = −Q to Q do
    Compute slice W_q0 = G_Ml ∩ ρ_q0                        ▷ Sec. 5
    for p = −P to P do
      Compute α_Ap and α_Bp for each edge E_AB ∈ E_q0        ▷ Eq. (17)
      Sort edges in E_q0 according to corresponding α_Bp
      Let Q = 0
```

for $m=M_{\min}$ to $M_{\max}$ **do** ▷ Sec. 3
  Add edge $E_{AB} \in \mathcal{E}_{q0}$ to $\mathcal{Q}$, if $\alpha_{Bp} < \xi_m$
  Remove edge $E_{AB} \in \mathcal{Q}$ from $\mathcal{Q}$, if $\alpha_{Ap} < \xi_m$
  if $\mathcal{Q} \neq 0$ **then**
    Compute all intersection $\{r^i_{pq0m}\} = R_{pq0m} \cap E_{AB}$, $E_{AB} \in \mathcal{Q}$ ▷ Sec. 5
    Obtain the nearest intersection $r_{pq0m} = \min_i\{r^i_{pq0m}\}$
    Add contribution $\frac{A_{pqlm}}{r_{pq0m}} \exp\left(i\frac{2\pi}{\lambda} r_{pq0m}\right) w_{pqlm} \cos \xi_m$ to u_{pq} ▷ Eq. (9)
  end if
 end for
end for
end for
end for

6. DISTRIBUTED COMPUTING

One strategy to accelerate digital optical field computation is distributing the task on a cluster of computers. A linear speedup is desirable; however, it is practically impossible to reach such a speedup owing to the sequential parts of the algorithm and the communication necessary to coordinate the computation [35]. In this section we present a distribution scheme that utilizes the algorithm described in the previous section. We aim on minimizing the communication load through an efficient task decomposition.

We assume a homogeneous cluster of nodes connected via a network; i.e., each node is a stand-alone computer with identical configuration. In order to maintain a minimal communication load, we exploit the homogeneous cluster environment and we employ a static workload balancing. We algorithmically assign subtasks to nodes prior to the evaluation, and after assigning the subtasks no other communication is necessary until all nodes process their assigned subtasks.

The key feature of the proposed distribution is a decomposition of the algorithm into subtasks. Since we evaluate each sample u_{pq} independently of other samples, we can partition the desired optical field into arbitrary segments, i.e., tiles on H. There are two factors that control the tile selection.

First, the algorithm Alg. 1, which is used for computing the diffraction pattern, exploits preprocessing to speed up the synthesis. It preprocesses the mesh G to speed up the calculation of slices W_{q0}, and it preprocesses each slice W_{q0} to speed up intersection evaluations. In order to limit the communication load and to avoid synchronization, each node needs to repeat the preprocessing procedure for each subtask. This has a negative effect on the efficiency. In order to minimize the number of repetitions of the preprocessing step for each slice W_{q0} the tiles are chosen to be composed of the optical field rows.

Second, the static load balancing requires tasks to finish at the same time to be efficient. Therefore we assign each N_nth row to one tile, where N_n is the number of nodes. This is based on the assumption that a sequence of N_n slices (W_{q0}'s) will have a similar number of edges and vertices. This is justified by the small sampling step D_y, which is much smaller than the minimal details in commonly used meshes. Hence, the processing times of all subtasks are similar. To be efficient, the entire subtask has to fit into physical memory of a node. If the subtask does not fit into node physical memory, we have to use a lower number of rows so that we can generate $n_s N_n$ subtasks where $n_s \in \mathbb{N}$. In the following text, we keep $n_s = 1$ for the sake of simplicity.

Once the subtasks are established, they are transferred to the corresponding computation nodes for processing. The total number of processed rows in each subtask is equal to the number of rows processed by sequential computation. As a consequence, the total time spent on distributed evaluation of the rows is reduced by almost $1/N_n$ compared with the sequential computation.

The distribution described scheme can also be used in conjunction with the GPU-based algorithm by replacing the algorithm used with the one given in Section 9.

7. FRESNEL APPROXIMATION

As already discussed in Section 3 and as summarized by Eq. (12), the propagation angles can be quite small. This is true, in particular, for implementations targeted for optical reconstructions using currently available SLMs. Under these conditions, the Fresnel approximation is a valid choice, and the term $\exp(\mathrm{i}kr_{pqlm})/r_{pqlm}$ in Eq. (9) is approximated as [14,36]

$$\frac{\exp(\mathrm{i}kr_{pqlm})}{r_{pqlm}} \approx \frac{1}{z_{pqlm}} \exp(\mathrm{i}kz_{pqlm}) \times \exp\left[\mathrm{i}\frac{k}{2z_{pqlm}}(x^2_{pqlm} + y^2_{pqlm})\right], \quad (20)$$

where $(\mathbf{s}_{pqlm} - \mathbf{x}_{pq}) = (x_{pqlm}, y_{pqlm}, z_{pqlm})$.

The benefits of the Fresnel approximation are even more significant in our case due to restrictions imposed on S as described in Section 4. First of all, since the distance from H to S is always restricted to be an integer multiple of λ, the phase factor $\exp(\mathrm{i}kz_{pqlm})$ in Eq. (20) is always equal to 1. Therefore, we define a generic kernel

$$h_z(x,y) = \frac{1}{z} \exp\left[\mathrm{i}\frac{k}{2z}(x^2 + y^2)\right], \quad (21)$$

where z is a parameter. Observing that

$$h_{z_{pqlm}}(x,y) = \sigma^2 h_z(\sigma x, \sigma y), \quad (22)$$

where $\sigma = (z/z_{pqlm})^{1/2}$, we can easily compute the needed kernel for any z_{pqlm} from the precomputed and stored ge-

neric kernel by properly zooming the kernel and by modifying the gain factor. Please note that usually σ is close to 1, and therefore the associated effect on the amplitude of the kernel is negligible. However, the effect on phase due to scaling of the variables (x,y) is important. We exploit this property in the implementation given in Section 8 by computing the generic kernel once and then by zooming it using noninteger interpolation techniques.

8. ACCELERATION BY REDUCED OCCLUSION

Synthesis of a diffraction pattern, as described in previous sections, is a computationally demanding task. Therefore, it is worthwhile to investigate alternative methods that sacrifice quality somewhat in return for a gain in computational performance. We expect that the loss in quality will be minimal, whereas the gain is significant.

As described in Section 5, our method computes many intermediate horizontal parallax only (HPO) optical fields and then combines those to get the full parallax photorealistic reconstructions. There is an alternative procedure that needs a fewer number of HPO optical fields and still yields full-parallax but reduced-occlusion reconstructions.

The approach is based on computing a full-parallax partial optical field U_q over H formed by a slice W_{q0} defined in Section 5. However, each slice W_{q0} is processed independently of the others without taking into consideration the occlusions among the slices. The final diffraction pattern is the sum of the optical fields generated by slices, i.e., $U=\Sigma_q U_q$. In other words, the proposed approach solves the occlusion and solid-surface problems along the x axis, but the occlusions along the y axis are omitted. However, the result is still a full-parallax hologram.

The computational approach shares a part of the algorithm presented in Section 5 that calculates the optical field for $l=0$. The slicing, point-source generation, and occlusion evaluation are the same. The difference in the algorithm takes place when a sample u_{pq} is processed and the distance r_{pqlm} to the nearest intersection $\mathbf{s}_{pqlm}$ is calculated. In the original algorithm, the point source created at the intersection $\mathbf{s}_{pqlm}$ contributes only to a single sample u_{pq} following Eq. (9); that, after all samples are evaluated, results in an HPO optical field for a slice. However, when we do not care about the occlusions along the y direction, each u_{pq}, for a fixed p and variable q, receives a field contribution from the source at the already computed intersection point. We denote the field u_{pq} for fixed p as the column τ_p and take advantage of this observation as described in the next paragraph.

We start by using the result obtained in Section 7 by precomputing and storing a discrete generic kernel. This gives us an optical field on H that would be obtained by a point source located at $(0,0,z)$. Using the coordinates of the computed nearest intersection $\mathbf{s}_{pqlm}$ and the point $\mathbf{x}_{pq}$ at τ_p, we find the appropriate zoom σ in Eq. (22) and the spatial shift to be applied to the precomputed optical field. Note that the needed zoom and shift might necessitate subpixel accuracy. We handle this by interpolating the precomputed field linearly along the q direction. Since we compute one column τ_p at a time, only a column of the precomputed field is scaled and shifted. This results in an efficient calculation. Also, due to the symmetry of the generic kernel, it is sufficient to store only its one quadrant to reduce memory requirements.

9. GPU

The GPU is a central element of a graphics card. It is a specialized processor based on highly parallel architectures devoted to speeding up conventional computer graphics tasks such as processing triangular meshes, solving occlusion problems, and rendering final 2D images to be displayed on computer monitors. The current generation of GPUs is programmable, and therefore it is possible to transfer appropriate CPU tasks to the GPU. One such application is digital hologram synthesis [13,14,37,38]. In this section we describe a version of our method that exploits the GPU to decrease computation time.

The GPU is designed to transform triangular meshes and sample them. The sampling task is performed by a hard-wired unit called the rasterizer. The output of the rasterizer is a rectangular grid of the samples to be displayed on the monitor. We use the GPU to output our optical field on H, which is also discretized in a regular rectangular fashion. To show the relationship between the rasterizer and our approach, let us assume an orthographic projection of the meshes. Then the task of the rasterizer is equivalent to casting a ray R_{pq00} from each sample point $\mathbf{x}_{pq}$ on the grid. The directional vector of each ray is constant: $\hat{\mathbf{r}}_{00}=(0,0,1)$. The rasterizer finds intersections of the rays with the meshes. In further processing of intersections, the GPU selects the nearest intersection by using the depth buffer technique [25]. The diffraction computation described in this paper requires similar computational steps.

In order for the rasterizer to be used to evaluate the intersection discussed in Section 5, the vertices of a triangular mesh $G=(\mathcal{V},\mathcal{T})$ have to be transformed by a transformation $\mathcal{P}_{lm}$. The transformation $\mathcal{P}_{lm}$ is an extension of the transformation $\mathcal{M}_l$ used in Section 5. The extension into the x axis, however, leads to a hemisphere parameterization different from Eq. (5). Let us define two angles just for the sake of defining the transformation: $\bar{\xi}_m$ and $\bar{\psi}_l$. The transformation exploits the fact that $\hat{\mathbf{r}}_{00}\cdot\hat{\mathbf{r}}_{lm}=1/\zeta_{lm}$, where ζ_{lm} is proportional to the distance r_{pqlm} used in Eqs. (9) and (13). For the left-handed coordinate system, $\mathcal{P}_{lm}$ is

$$\mathcal{P}_{lm}=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ -\tan\bar{\xi}_m & -\tan\bar{\psi}_l & \zeta_{lm}\end{bmatrix}. \tag{23}$$

A relation that maps hemisphere parameterization of Eq. (23) to hemisphere parameterization of Eq. (5) is

$$\tan\bar{\xi}_m=\tan\xi_m,\quad \tan\bar{\psi}_l=\frac{\tan\psi_l}{\cos\xi_m},\quad \zeta_{lm}=\frac{1}{\cos\xi_m\cos\psi_l}. \tag{24}$$

After transforming, the z-axis coordinate of the transformed vertex $\mathbf{v} \in \mathcal{P}_{lm}\{\mathcal{V}\}$ equals the distance r_{pqlm}. The whole GPU synthesis is summarized in Alg. 2. Nevertheless, there is still the issue of the numerical accuracy of the GPU, which is discussed in the following paragraphs.

Algorithm 2. A skeleton of an algorithm that evaluates the diffraction pattern on the GPU as described in Section 9. For details on computation, consult the referred portion of the text.

```
for all l=L_min to L_max do                                   ▷ Sec. 3
  for all m=M_min to M_max do                                 ▷ Sec. 3
    Build transformation matrix P_lm                          ▷ Eq. (23), Eq. (24)
    :Let u'_pq=u_pq, ∀_p,q
    for all G_i⊆G do
      Compute fractional part of Φ_ilm with higher precision  ▷ Eq. (25)
      Render G_i to calculate relevant samples u'_pq
    end for
    Replace samples u_pq with u'_pq
  end for
end for
```

For performance reasons, the numerical accuracy nowadays of the GPU is limited to 32-bit floating point numbers. This accuracy is insufficient for some parts of Alg. 2. One accuracy problem occurs in Eq. (24) for angles $\xi_m \approx 0$ and $\psi_l \approx 0$. Regardless of the angles ξ_m and ψ_l, ζ_{lm} is rounded to 1 when stored as a 32-bit floating-point number. This issue is resolved by using $\zeta_{lm}-1$ instead of ζ_{lm}. A more serious accuracy problem occurs during the phase-shift calculation in Eq. (9).

In Eq. (9) the phase of a point source is shifted by $\chi_{pqlm}=\exp(\mathrm{i}2\pi\varphi_{pqlm})$, where $\varphi_{pqlm}=r_{pqlm}/\lambda$. The shift χ_{pqlm} is stored as $\chi_{pqlm}=\cos(2\pi\varphi_{pqlm})+\mathrm{i}\sin(2\pi\varphi_{pqlm})$. Since both the sine and the cosine functions are periodic, only a fractional part of φ_{pqlm} is needed. For the usual scenario, $\lambda \approx 10^{-7}$ and $r_{pqlm} \approx 10^{-1}$, and thus the phase shift is $\varphi_{pqlm} \approx 10^6 \sim 2^{20}$. This leaves only 4 bits of the fractional part in the mantissa and causes disturbing artifacts in the reconstruction.

To address the insufficient accuracy we define planes κ_i: $z=iD_z$. The distance r_κ is the length of a ray R_{pqlm} between two successive planes κ_i and κ_{i+1} as depicted in Fig. 7(a). The distance r_κ is finite because $|\psi_l| < \pi/2$ and $|\xi_m| < \pi/2$. We choose D_z so that the fractional part of r_κ/λ has at least 10 bits. The maximum error of omitted bits is $2\pi \times 2^{-10} \sim 10^{-3} \ll 1$ rad. This also matches the condition for the validity of the Fresnel approximation [15,36].

In general cases, planes κ_i are used to split the triangular mesh G into many triangular meshes G_i. As a consequence, $\varphi_{pqlm}=\varphi'_{pqlm}+\Phi_{ilm}$, where

$$\phi'_{pqlm} = \zeta_{lm}\frac{z'_{\mathbf{v}}}{\lambda}, \qquad \Phi_{ilm} = \zeta_{lm}\frac{iD_z}{\lambda}, \tag{25}$$

and $z'_{\mathbf{v}}=z_{\mathbf{v}}-iD_z$ for each vertex $\mathbf{v} \in G_i$; see Fig. 7(b). If the mesh G fits between planes κ_i and κ_{i+1}, it is not divided. Since Φ_{ilm} is constant for mesh G_i and the given direction $\hat{\mathbf{r}}_{lm}$, it is calculated with higher accuracy outside the GPU with minimal effect on performance.

10. RESULTS

Results obtained by the various computational procedures outlined in previous sections are presented in this section together with comparisons.

Numerical reconstruction results from the computed optical fields are obtained by propagating the fields to planes parallel to H. This is accomplished by taking the discrete Fourier transform (DFT) of the computed optical field, multiplying it by the transfer function corresponding to an angular spectrum, and then taking the inverse DFT [15,19]. We pad the optical field with zeros so that the resolution becomes 8192×8192 samples. This pad-

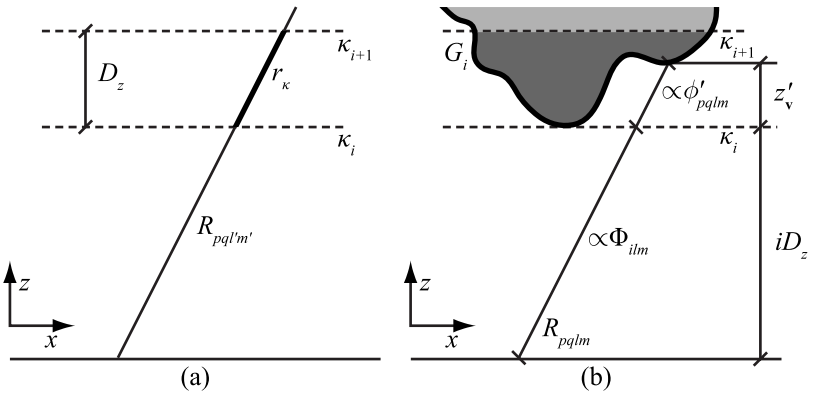


Fig. 7. (a) Evaluation of the longest distance r_κ and (b) decomposition of $z_{\mathbf{v}}$ including distances proportional to phase shifts φ'_{pqlm} and Φ_{ilm}. The longest distance is computed from D_z and $R_{pql'm'}$, where $l'=\max\{|L_{\min}|,|L_{\max}|\}$ and $m'=\max\{|M_{\min}|,|M_{\max}|\}$.

ding reduces the unwanted features associated with the periodicity implied by the DFT. Figures 8–13 show the intensity of the propagated optical fields.

Five different scenes with different levels of complexity and properties are used. The magnitudes of A_{pqlm} over the object surfaces are obtained according to Phong's illumination model [39]. The "Photo" is the simplest scene, which is a textured rectangular 2D pattern having a resolution equal to the resolution of the computed optical field. The texture contains small details and smooth intensity variations. The second scene, "Primitives", consists of four different simple 3D objects placed along the z axis at different depths. This scene has the longest distance between the nearest and the farthest object points of all the scenes (0.2 m). The third scene, "Lancaster", contains a model of the Avro Lancaster airplane. This model consists of over 80,000 triangles, and the magnitude of the texture is slowly varying. The fourth scene, "Chess," has several subsurfaces that occlude each other. The last scene, "Bunny", also has textured sub-surfaces. This scene demonstrates well the ability of our method to handle photorealistic content. The complete parameters of all scenes are provided in Table 1.

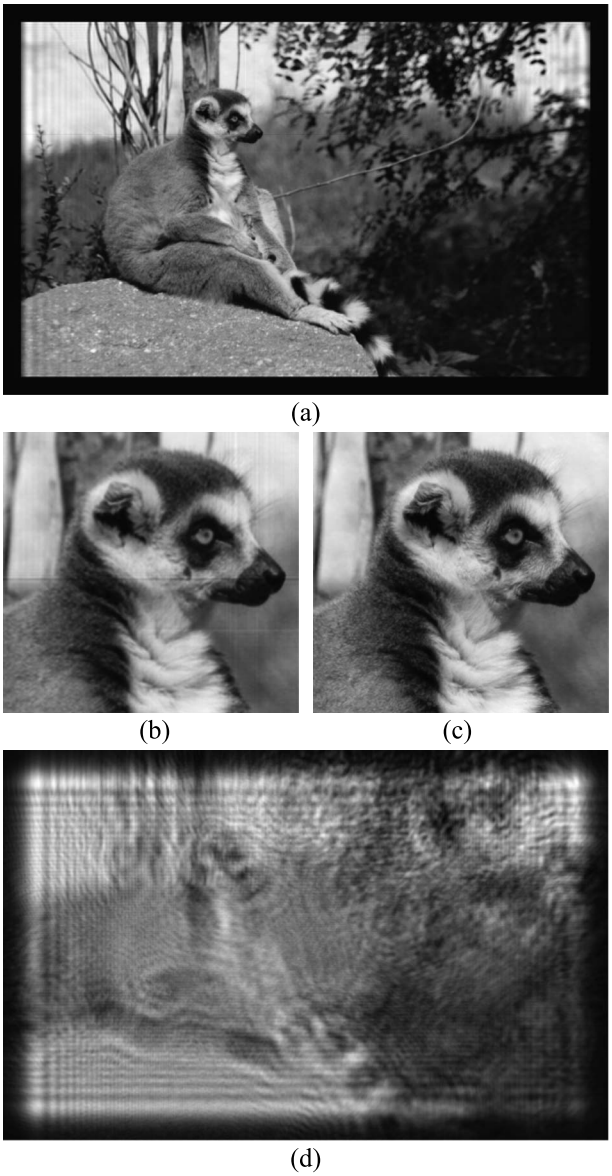


Fig. 8. (a) Numerical reconstruction (intensity) from a GPU-computed optical field. The scene consists of a single textured plane (2D object) parallel to H. The distance between the object plane and the hologram plane is 0.42 m. The resulting image is clipped to 2048×1320 pixels. (b) An enlarged detail of the reconstruction is compared with (c) the original texture. (d) The magnitude of the entire computed optical field pattern which is then used to get the reconstruction. (The photo is courtesy of Libor Váša).

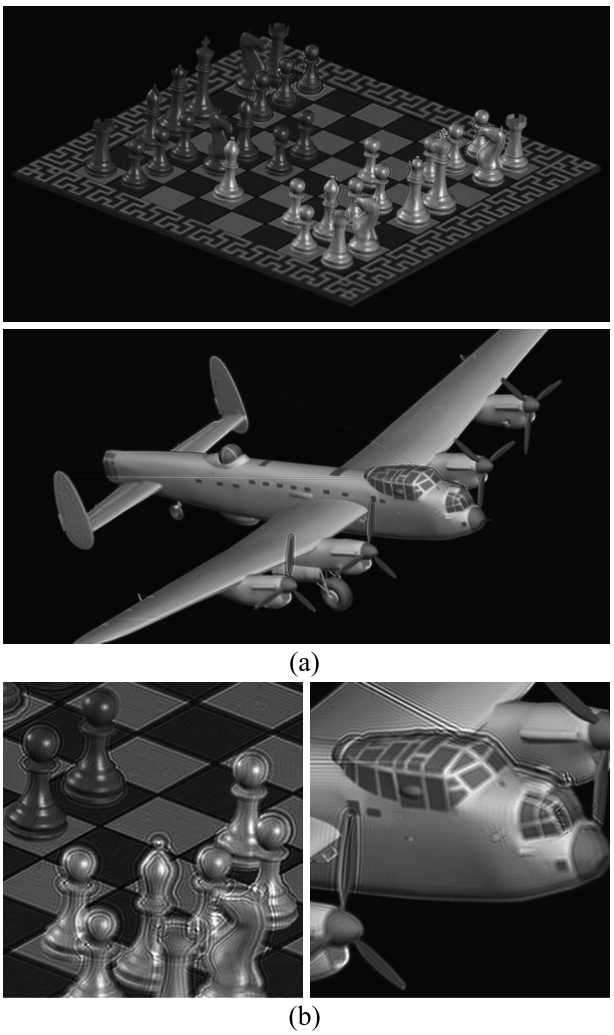


Fig. 9. (a) Numerical reconstructions (intensity) from optical fields calculated by a GPU and (b) details of the reconstructions. Presented scenes are "Chess" and "Lancaster." Both reconstructions are computed at a distance of 0.5 m.

Figure 8 presents a numerical reconstruction of 2D "Photo" from its optical field computed as described in Section 9. A closeup [Fig. 8(b)] is provided together with the original photo [Fig. 8(c)] for visual comparison. A loss of detail is visible in the closeup of the reconstruction. The fur of the creature is blurred, but the details of the hairs around its head are still visible. The magnitude of the entire computed optical field, which is then used to get the reconstruction, is presented in Fig. 8(d).

The numerical reconstructions from the optical fields computed by our full-parallax method using "Chess" and "Lancaster" are depicted in Fig. 9. Both scenes are suc-

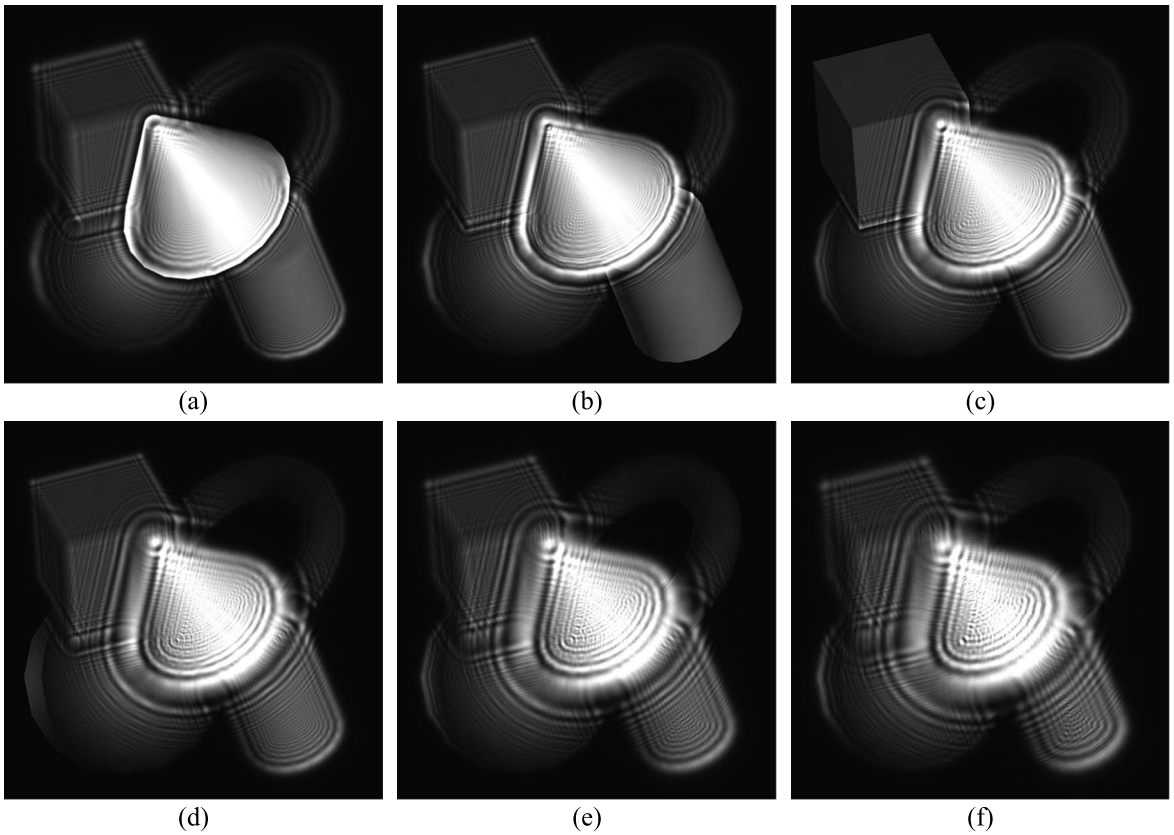


Fig. 10. Numerical reconstructions from a GPU-computed optical field focused at the (a) cone, (b) cylinder, (c) box, (d) sphere, (e) torus. (f) An out-of-focus reconstruction. The optical field is calculated by a GPU. All images are clipped to a resolution of 1100×1100 pixels.

cessfully reconstructed. Note that the wood texture present in the scene "Chess" is recognizable in Fig. 9(b).

The numerical reconstructions of the scene "Primitives" at different depths are given in Fig. 10. The objects at the focused depths are sharp, as expected. Even the farthest object (torus) is successfully reconstructed; see Fig. 10(e). Out-of-focus objects yield fringes around them, as expected.

We present the reconstruction of the scene "Bunny" in Fig. 11. This reconstruction demonstrates the visual quality that we can achieve with our full-parallax method. Unlike the previous results, the resolution of this optical field is 4096×4096 samples. Actually, we computed three different optical fields for three different wavelengths: 650 nm (red), 510 nm (green), and 475 nm (blue). From those three optical fields we reconstructed three separate images and digitally combined them into one color (online) image.

In addition to the numerical reconstructions, optical reconstructions were also carried out. Computed optical fields are converted to off-axis holograms by adding a reference beam and then computing the intensity pattern. An SLM that operates in the amplitude-only mode is used for the optical reconstruction. The SLM has a resolution of 1280×720 pixels, and the size of each pixel is 12 μm. The reconstructed optical fields were recorded by a CCD camera stripped of all additional optics. The acquired image for the "Lancaster" is shown in Fig. 12. The optical reconstruction reveals some degradation compared with the numerical reconstruction, but this is expected because of the much smaller SLM resolution and the noise inherent in the physical setup including the camera.

For a comparison, the numerical reconstructions from optical fields obtained by the full-parallax CPU implementation and the reduced-occlusion implementation are shown in Figs. 13(a) and 13(b), respectively.

We compared the results obtained from the single-CPU, multiple-CPU and GPU implementations of the full-parallax and reduced-occlusion methods. The reconstructions obtained by these methods are almost indistinguishable visually. The numerical comparison, however, reveals differences that are presented in Table 2. We compared intensity images reconstructed numerically from optical fields computed from the scene "Photo".

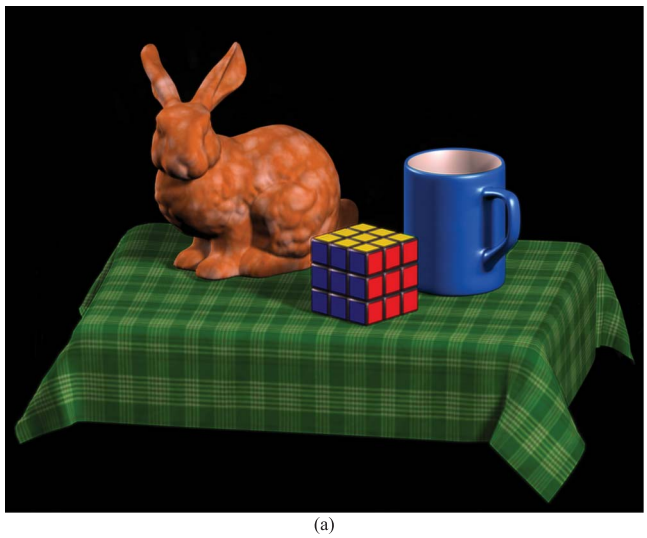
For that purpose we used the maximum difference

$$\Delta_{\max} = \frac{\max_{pq}\{||u_{pq}|^2 - |u'_{pq}|^2|\}}{\max_{pq}\{|u_{pq}|^2\}} \tag{26}$$

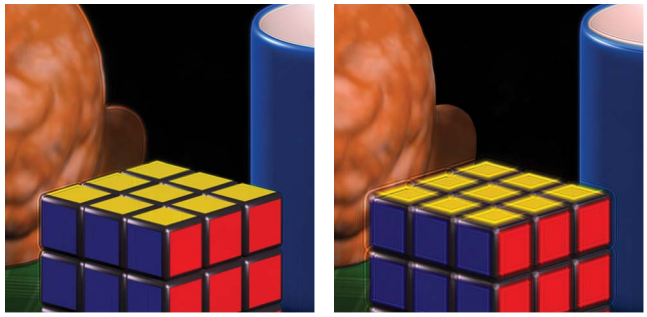
and the mean square error (MSE)

$$\mathrm{MSE} = \frac{1}{PQ}\frac{\Sigma_p \Sigma_q (|u_{pq}|^2 - |u'_{pq}|^2)^2}{\max_{pq}\{|u_{pq}|^2\}}, \tag{27}$$

where u_{pq} is a value of the optical field reconstructed from a result calculated by the CPU full-parallax version (reference optical field); u'_{pq} is the value of the compared optical field; and P and Q are the number of samples along the x and y directions, respectively. From the numbers in Table 2 it is apparent that the differences for the distrib-



(a)



(b) (c)

Fig. 11. (Color online) (a) Full-color (online) numerical reconstruction. The color is composed of three components at wavelengths 650 mn (red), 510 nm (green), and 475 nm (blue). The three optical fields are simultaneously computed by a GPU. (b) A detail of the reconstruction. (c) The same detail reconstructed at a different depth.

uted CPU full-parallax method are negligible. The more pronounced difference measured for the GPU implementation is caused by the reduced numerical accuracy of the GPU and rounding operations of the rasterizer unit. As expected, the reduced-occlusion case gives the highest difference. This is the result of the different evaluation of the vertical parallax.

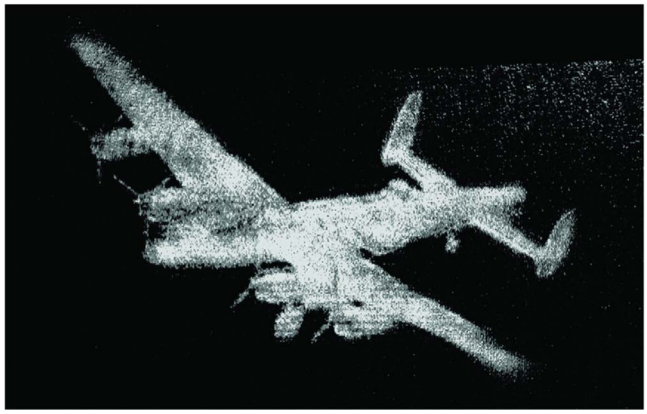
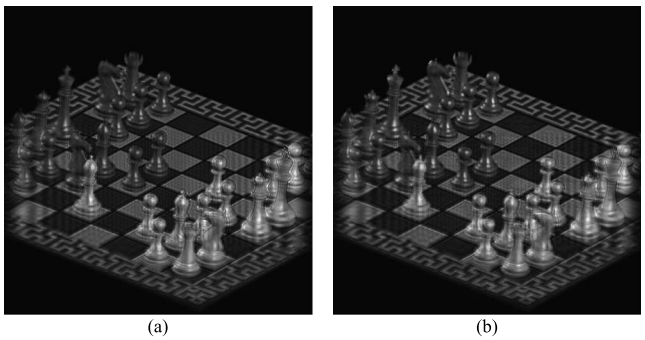


Fig. 12. Optical reconstruction from an off-axis hologram. The hologram is obtained from the optical field calculated by a GPU. The incidence angle of the reference beam is 0.758° diagonally to fit the SLM parameters. The reconstruction uses an amplitude-only SLM, and the reconstructed image is captured by a CCD camera without any lens.



(a) (b)

Fig. 13. Numerical reconstruction (a) from the CPU computed full-parallax optical field and (b) from the CPU-computed reduced-occlusion optical field. Focus of the numerical reconstruction is approximately at the black pawn in the middle of the picture.

We also measured the processing times associated with the described computational procedures. We tested implementations on a single CPU (Intel Xeon 3.2 GHz, 1 GB RAM), on a cluster of CPU's (10 × Intel Xeon 3.2 GHz, 1 GB RAM, 1 Gbps Ethernet), and on a computer equipped with NVidia GeForce 8800 GTX GPU. Taking into account the time requirements of the CPU implementations, we reduced the resolution to 1024 × 1024 samples. The only exception is the scene "Bunny," which was computed just on a GPU and had a resolution of 4096 × 4096 samples. This resolution demonstrates the ability of the GPU implementation to calculate large optical fields in acceptable times. The computation times are summarized in Table 3. Considering the rapid development of the CPU and especially the GPU performance, the measured values should be interpreted only for comparison purposes. Table 4 lists the speedup factors achieved by different implementations with respect to the single-CPU implementation. It is apparent that the speedup achieved by distributing the computation is approximately linear. What is also apparent is the expected superior computational performance of the GPU imple-

Table 1. Properties of Scenes Used for the Tests

Scene	Triangles	Projected Size[a] (%)	Scene Distance (m)	Scene Depth (mm)
Photo	2	100	0.42	0
Primitives	972	30	0.40	202
Lancaster	83,848	23	0.49	12
Chess	42,566	32	0.49	13
Bunny	84,580	29	0.45	24

[a]Percentage of the hologram frame area covered by an orthogonal projection of the scene.

Table 2. Difference Comparisons of the Scene "Photo"

Version	Δ_{max}	MSE
Distributed on 5 CPUs	0.003	0.895×10^{-7}
CPU reduced occlusion	0.259	0.180×10^{-2}
GPU	0.104	0.970×10^{-4}

Table 3. Computation Times (hr)

	CPU		Distrib. on N CPUs: Full Parallax		
Scene	Full Parallax	Reduced Occl.	$N=5$	$N=10$	GPU: Full Parallax
Photo	218.9	15.3	44.2	22.3	0.3
Primitives	65.4	5.2	13.1	6.4	0.2
Lancaster	53.3	3.8	10.5	5.2	0.4
Chess	78.6	5.5	16.4	8.2	0.3
Bunny[a]					96.0

[a]Bunny was used for computing a large (4096×4096) optical field. The computation time applies for a GPU implementation that simultaneously computes three color channels.

Table 4. Relative Speedup

	CPU		Distrib. on N CPUs: Full Parallax		
Scene	Full Parallax	Reduced Occl.	$N=5$	$N=10$	GPU: Full Parallax
Photo	1.00	14.30	4.96	9.81	718.86
Primitives	1.00	12.58	5.00	10.53	327.10
Lancaster	1.00	13.95	5.11	10.16	140.16
Chess	1.00	14.40	4.80	9.58	245.69

mentation. This is due to the massive parallelism in the GPU architecture. Thus, we expect that by increasing the number of computation nodes of the cluster sufficiently, we can achieve similar results in the distributed environment, as well. Finally, the acceleration by reducing the parallax provides significantly shorter computation time by sacrificing the quality. This approach is therefore quite suitable for a fast preview mode.

11. SUMMARY AND CONCLUSIONS

We have presented detailed computational algorithms for computing optical fields of objects; our primary goal was to achieve photorealistic reconstructions. Therefore, our 3D objects had a large number of triangles in their mesh representations. Furthermore, we adopted realistic surface illumination as commonly employed in computer graphics applications. A diffraction model suitable for this goal was adopted and discretization effects were discussed. The model is based on the local angular reflectivity distribution of a textured surface. We developed two algorithms for the full-parallax and reduced-occlusion cases; the former algorithm is implemented on a single CPU, multiple CPU, and a GPU, and the latter algorithm is implemented only on a single CPU.

We conclude that the optical fields obtained from the adopted diffraction model and various implementations of its discrete version provide successful photorealistic reconstructions. The presented algorithms perform well both for simple (planar) objects and quite complicated 3D scenes with large depth. Our conclusions are based both on numerical reconstructions and on optical reconstructions.

In addition to full-parallax implementations, we investigated a reduced-occlusion alternative for faster computation. It is observed that a significant speedup with rather small degradation is possible; this approach is therefore quite suitable for generating fast previews. Optical reconstruction quality is lower than numerical reconstruction quality. This is due to the SLM resolution used as well as the noise inherent in the physical reconstruction environment.

On the basis of comparisons of different hardware (single-CPU, multiple-CPU, and GPU) implementations, we conclude that all provide almost the same visual quality but that the needed computation times vary significantly. As expected, GPU implementation is considerably faster.

The proposed solution emphasizes the ability to exploit parallel and distributed computing. We are convinced that the computational complexity of the diffraction-pattern computation cannot be significantly reduced without sacrificing the quality of the reconstructed image. As a consequence of the particular sampling scheme over the object surface, our method allows a straightforward exploitation of acceleration techniques such as computational cluster or GPU. The presented algorithms achieve a speedup factor that is almost a linear function of the number of processors. We showed that holograms with one mega-pixel in size can be computed in tens of minutes using commonly available computational resources.

We conclude that the presented algorithms and their indicated implementations are able to generate holograms of arbitrary scenes that have a format common in modern 3D authoring tools used in the multimedia industry. This feature is crucial for easy integration into well-established computation pipelines. Furthermore, there

are no drawbacks that would prevent processing of more complicated scenes and computation of larger holograms. Therefore, the presented procedures can be easily used in applications that require larger holograms of larger and more complicated scenes.

ACKNOWLEDGMENTS

This work has been supported by the Ministry of Education, Youth, and Sports of the Czech Republic under the research program LC-06008 (Center for Computer Graphics) and by the Europen Union (EU) project within FP6 under grant 511568 with the acronym 3DTV. The authors thank Metodi Kovachev, Rossitza Ilieva, and Fahri Yaraş for the optical reconstructions conducted at Bilkent University holographic 3DTV Laboratories. The authors are grateful to Václav Skala for his invaluable comments while this work was conducted.

REFERENCES

1. D. Gabor, "A new microscopic principle," Nature **161**, 777–778 (1948).
2. D. Gabor, "Microscopy by reconstructed wavefronts," Proc. R. Soc. London, Ser. A **197**, 454–487 (1949).
3. L. Onural, A. Gotchev, H. M. Ozaktas, and E. Stoykova, "A survey of signal processing problems and tools in holographic 3DTV," IEEE Trans. Circuits Syst. Video Technol. **17**, 1631–1646 (2007).
4. L. Yaroslavskii and N. Merzlyakov, *Methods of Digital Holography*, (Consultants Bureau, 1980).
5. G. Tricoles, "Computer generated holograms: an historical review," Appl. Opt. **26**, 4351–4360 (1987).
6. J. Rosen, "Computer-generated holograms of images reconstructed on curved surfaces," Appl. Opt. **38**, 6136–6140 (1999).
7. D. Mendlovic, G. Shabtay, U. Levi, Z. Zalevsky, and E. Marom, "Encoding technique for design of zero-order (on-axis) fraunhofer computer-generated holograms," Appl. Opt. **36**, 8427–8434 (1997).
8. A. G. Kirk and T. J. Hall, "Design of computer generated holograms by simulated annealing: observation of metastable states," J. Mod. Opt. **39**, 2531–2539 (1992).
9. V. Arrizón, G. Méndez, and D. Sánchez-de La-Llave, "Accurate encoding of arbitrary complex fields with amplitude-only liquid crystal spatial light modulators," Opt. Express **13**, 7913–7927 (2005).
10. Y. Sando, M. Itoh, and T. Yatagai, "Full-color computer-generated holograms using 3-D Fourier spectra," Opt. Express **12**, 6246–6251 (2004).
11. K. Matsushima and M. Takai, "Recurrence formulas for fast creation of synthetic three-dimensional holograms," Appl. Opt. **39**, 6587–6594 (2000).
12. M. Cywiak, M. Servin, and F. M. Santoyo, "Wave-front propagation by gaussian superposition," Opt. Commun. **195**, 351–359 (2001).
13. L. Ahrenberg, P. Benzie, M. Magnor, and J. Watson, "Computer generated holography using parallel commodity graphics hardware," Opt. Express **14**, 7636–7641 (2006).
14. A. Ritter, J. Böttger, O. Deussen, M. König, and T. Strothotte, "Hardware-based rendering of full-parallax synthetic holograms," Appl. Opt. **38**, 1364–1369 (1999).
15. J. Goodman, *Introduction to Fourier Optics*, 3rd ed. (Roberts, 2005).
16. G. D. Sherman, "Application of the convolution theorem to Rayleigh's integral formulas," J. Opt. Soc. Am. **57**, 546–547 (1967).
17. N. Delen and B. Hooker, "Free-space beam propagation between arbitrarily oriented planes based on full diffraction theory: a fast Fourier transform approach," J. Opt. Soc. Am. A **15**, 857–867 (1998).
18. G. Esmer and L. Onural, "Computation of holographic patterns between tilted planes," Proc. SPIE **6252**, 62521K (2006).
19. L. Onural and H. M. Ozaktas, "Signal processing issues in diffraction and holographic 3DTV," Signal Process. Image Commun. **22**, ss169–177 (2007).
20. L. Onural, "Sampling of the diffraction field," Appl. Opt. **39**, 5929–5935 (2000).
21. A. Stern and B. Javidi, "Improved-resolution digital holography using the generalized sampling theorem for locally band-limited fields," J. Opt. Soc. Am. A **23**, 1227–1235 (2006).
22. L. Onural, "Exact analysis of the effects of sampling of the scalar diffraction field," J. Opt. Soc. Am. A **24**, 359–367 (2007).
23. J. T. Kajiya, "The rendering equation," ACM SIGGRAPH Comput. Graph. **20**, 143–150 (1986).
24. A. S. Glassner, *Principles of Digital Image Synthesis*, (Morgan Kaufmann, 1995), 1st ed..
25. A. Watt, *3D Computer Graphics*, 3rd ed. (Addison-Wesley, 2000).
26. M. Lucente, "Optimization of hologram computation for real-time display," Proc. SPIE **1667**, 32–43 (1992).
27. M. Lucente and T. A. Galyean, "Rendering interactive holographic images," in *Proceedings of SIGGRAPH '95* (ACM, 1995), pp. 387–394.
28. J. L. Juárez-Pérez, A. Olivares-Pérez, and L. R. Berriel-Valdos, "Nonredundant calculation for creating digital Fresnel holograms," Appl. Opt. **36**, 7437–7443 (1997).
29. G. B. Esmer, L. Onural, H. M. Ozaktas, V. Uzunov, and A. Gotchev, "Performance assessment of a diffraction field computation method based on source model," in Proceedings of the 3DTV-Conference 2008 (IEEE Xplore, 2008), pp. 257–260; http://ieeexplore.ieee.org.
30. M. Kovachev, R. Ilieva, P. Benzie, G. B. Esmer, L. Onural, J. Watson, and T. Reyhan, *Three-Dimensional Television: Capture, Transmission, and Display*, (Springer, 2007), chap. 15.
31. S. D. Roth, "Ray Casting for Modeling Solids," J. Comput. Graph. Image Process. **18**, 109–144 (1982).
32. D. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching* 2nd ed. (Addison-Wesley, 1998).
33. M. Janda, I. Hanák, and V. Skala, "Digital HPO hologram rendering pipeline," in *Proceedings of EG2006 Short Papers*, (Eurographics Association, 2006), pp. 81–84.
34. M. Lucente, "Diffraction-specific fringe computation for electro-holography," Ph.D. thesis (MIT, 1994).
35. G. Amdahl, "Validity of the single-processor approach to achieving large scale computing capabilities," in *AFIPS Conference Proceedings*, Vol. 30 (American Federation of Information Processing Societies Press, 1967), pp. 483–485.
36. M. Born and E. Wolf, *Principles of Optics*, 7th ed. (Cambridge U. Press, 2005).
37. N. Masuda, T. Ito, T. Tanaka, A. Shiraki, and T. Sugie, "Computer generated holography using parallel commodity graphics hardware," Opt. Express **14**, 603–608 (2006).
38. C. Petz and M. Magnor, "Fast hologram synthesis for 3d geometry models using graphics hardware," Proc. SPIE **5005**, 266–275 (2003).
39. B. Phong, "Illumination for computer generated pictures," Commun. ACM **18**, 311–317 (1975).