Calculation method for computer-generated holograms with cylindrical basic object light by using a graphics processing unit

Hironobu Sakata, Kouhei Hosoyachi, Chan-Young Yang, and Yuji Sakamoto*

Graduate School of Information Science and Technology, Hokkaido University, North 14 West 9 Sapporo-shi 060-0814 Japan

*Corresponding author: yuji@ist.hokudai.ac.jp

Received 1 August 2011; revised 28 October 2011; accepted 28 October 2011; posted 31 October 2011 (Doc. ID 152099); published 6 December 2011

It takes an enormous amount of time to calculate a computer-generated hologram (CGH). A fast calculation method for a CGH using precalculated object light has been proposed in which the light waves of an arbitrary object are calculated using transform calculations of the precalculated object light. However, this method requires a huge amount of memory. This paper proposes the use of a method that uses a cylindrical basic object light to reduce the memory requirement. Furthermore, it is accelerated by using a graphics processing unit (GPU). Experimental results show that the calculation speed on a GPU is about 65 times faster than that on a CPU. © 2011 Optical Society of America

OCIS codes: 090.1760, 090.1995.

1. Introduction

A computer-generated hologram (CGH) is a hologram made by computer simulation that is reconstructed into a three-dimensional (3D) image like that of an ordinary hologram. The object lights from various objects are simulated using wave propagation theory. Interference patterns of CGHs are simulated for not only real, but also for virtual, objects, and are recorded without using an optical system. However, both the computation time and calculation amount for CGHs are enormous, so that faster calculation and less memory use is required.

There are two major types of methods for calculating CGHs: point light methods and methods based on Fourier transforms [1]. With point light methods, calculating the object light from complex objects is easy, but the calculation time increases linearly with the number of point lights. With methods based on Fourier transforms, calculation is faster than with point light methods for complex objects because methods based on Fourier transforms calculate the propagation from an object plane to a hologram plane, and each plane includes many points. A fast Fourier transform (FFT) algorithm enables Fourier transform calculations to be faster.

Various methods have been proposed for reducing the time needed to calculate CGHs. Some precalculate the object light from a predetermined shape. Since objects consist of various shapes, these methods create an all-inclusive object light by combining object light that has been transformed in accordance with various object shapes. The study in [2] performs the combination and transforms in the frequency domain. The studies in [3,4] do so in the spatial domain. In both cases, the CGH calculation is faster.

A polygon model has been proposed that describes the shape of objects using polygons, i.e., an object is composed of polygons. For faster calculation, a precalculating method is also applied to the polygon. The object light propagating from a predetermined polygon is precalculated, and this precalculated object light is transformed into the object light of various polygons. The transforms are 3D affine transformations in the spatial domain. Scaling, rotation, and

0003-6935/11/34H306-09$15.00/0

skew transformations in the direction of the x, y, and z axes within the Fresnel region have been proposed [5,6]. With FFT for a square-shaped hologram with n pixels on each side, the calculation order of propagation is $O(n^2 \log n)$. With the calculation method using the 3D affine transformation, the calculation order is $O(n^2)$. This means that a larger n produces a larger difference in calculation time between the calculation method using the 3D affine transformation and the method using FFT. However, with the precalculating method, increasing n increases memory use.

We propose a new method using 3D affine transformation that requires a smaller memory by improving the basic object light. Processing is accelerated by using a graphics processing unit (GPU) acceleration to the proposed method. High-speed calculation of CGHs by using GPUs has been reported [7–10]. The proposed method is suitable for fast calculation using GPUs, because each part of the calculation is processed in parallel. For calculation using GPUs, data of a precalculated object light must be copied to the GPU memory beforehand. The time needed to copy data increases with the data size. Since the proposed method reduces the data size, it works effectively with a GPU.

2. Theory

A. Conventional Methods of Calculating CGHs

A calculation of hologram data requires object light that propagates from objects to a hologram plane. The propagation is defined by the Fresnel–Kirchhoff diffraction integral:

$$u_h(x_h,y_h,z_0) = \frac{j}{\lambda}\int\int_{-\infty}^{\infty} t(\hat{x},\hat{y},0)\frac{\exp(-jkr)}{r}\mathrm{d}\hat{x}\mathrm{d}\hat{y}, \quad (1)$$

where

$$r = [(x_h-\hat{x})^2 + (y_h-\hat{x})^2 + z_0^2]^{1/2},$$

$u_h(x_h,y_h,z_0)$ and $t(\hat{x},\hat{y},0)$ are object lights at the hologram plane and objects, respectively, z_0 is the distance of the hologram plane from objects, $j(=\sqrt{-1})$ is the imaginary unit, and $k(=2\pi/\lambda)$ is the wavenumber of the light [Fig. 1]. In the Fresnel region, Eq. (1) is approximated for the Fresnel diffraction equation, expressed as

$$u_h(x_h,y_h,z_0) = \frac{j}{\lambda z_0}\exp(-jkz_0)\int\int_{-\infty}^{\infty} t(\hat{x},\hat{y},0) \times \exp\left\{j\frac{k}{z_0}[(x_h-\hat{x})^2 + (y_h-\hat{y})^2]\right\}\mathrm{d}\hat{x}\mathrm{d}\hat{y}. \quad (2)$$

B. Method Using Plane Basic Object Light

In the calculation method proposed by Sakata and Sakamoto [6], an object light distribution propagated from a predetermined polygon is first calculated. As shown in Fig. 2, the polygon is a small triangular polygon called a "basic patch" for which the light distribution is defined as $g(\hat{x},\hat{y},0)$. The precalculated object light distribution, called a "plane basic object light" $u_b(x_b,y_b)$, is calculated on a plane parallel to the basic patch.

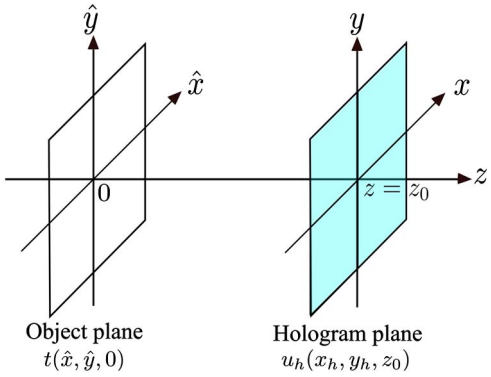


Fig. 1. (Color online) Fresnel–Kirchhoff diffraction theory.

In that method, object surfaces are covered by triangular polygons, which is called the "polygon model." The model consists of various triangular shapes, some of which are congruent with the basic patch and some of which are not. The object light that propagates from a triangle polygon different from the basic patch in terms of the shape, size, and location are calculated by transforming the basic object light.

A triangle can be transformed into various triangles by using 3D affine transformations, as shown in Fig. 3. There are five types of transforms. The transforms of a triangle by the 3D affine transformation are relevant to transform of its object light. In [6], the authors formulated transform algorithms of object light in accordance with the 3D affine transformation. The various combinations of transforms of the basic object light produce object light for various triangles. Consequently, an all-inclusive object light propagated from objects is represented as superposition of the object light for various triangles.

We call a triangular polygon that constitutes the objects a "polygon," and a polygon is derived from the basic patch by 3D affine transformation.

The method for calculating an object light includes four steps.

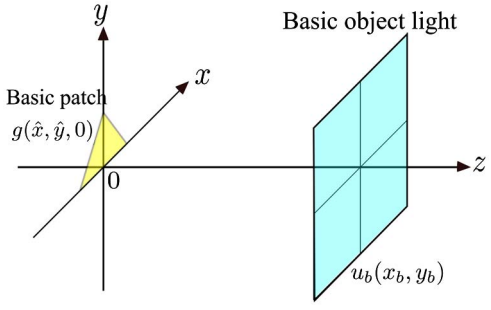


Fig. 2. (Color online) Plane basic object light.

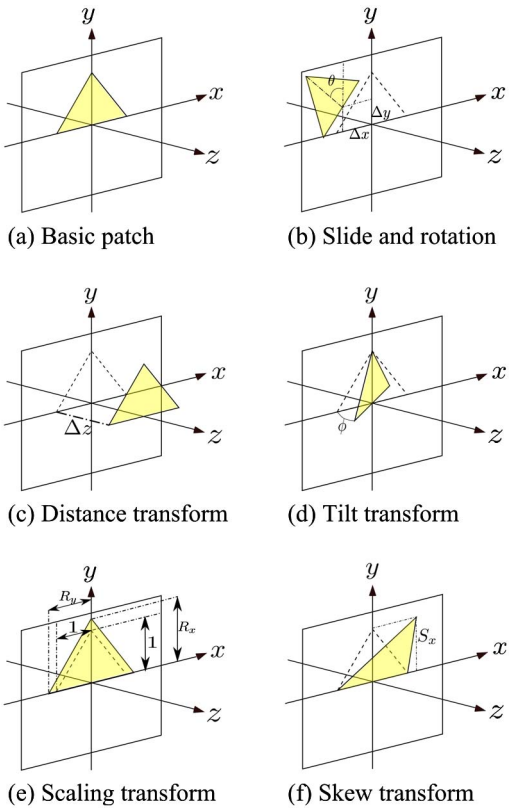


Fig. 3. (Color online) Transform of triangle by 3D affine transformation.

1. Define the basic patch and calculate the basic object light $u_b(x_b, y_b)$ propagated from the patch to a plane by using FFT (see Fig. 2).
2. Transform the basic object light u_b into the object light of each polygon i, u_{hi} on the hologram plane by using

$$u_{hi} = T_i[u_b], \tag{3}$$

where T_i is a transform of the object light.
3. Sum up object light $u_{h\,\mathrm{all}}$ from each polygon calculated in step 2. The object light wave from the whole object is

$$u_{h\,\mathrm{all}} = \sum_{i=1}^{P} u_{hi}. \tag{4}$$

4. Calculate interference pattern I using reference light R:

$$I = |u_{h\,\mathrm{all}} + R|^2. \tag{5}$$

Object light of various polygons are made by combining the five transforms.

The transform calculations are simple and low cost. When the number of polygons is P, the calculation order is $O(Pn^2)$, which is smaller than the calculation order of Fourier-transform-based methods, i.e., $O(Pn^2 \log n)$.

However, with one plane basic object light, the transforms can be performed only in a restricted region. In particular, the region in which tilt transform can be performed is very narrow. Transforming over the restricted region requires preparing plane basic object light from tilted basic patches. The data size for all the plane basic object light increases with the number.

3. Proposed Method

Using a plane shape for sampling the basic object light causes problems as described in Subsection 2.B: the restricted transform region and enormous required data size. Since a highly angled tilt transform requires high spatial frequency data, the tilt angle is restricted by the limitation of the recorded spatial frequency. A highly angled tilt transform also requires a wide area for the plane basic object light due to the use of data which are far from its center. These two requirements produce a large amount of data. For example, a 90 deg tilt transform needs plane basic object light with high resolution and an infinity. In [6], to avoid the narrow region, many plane basic object lights precalculated at regular intervals are used. Using a set of plane basic object lights, transforms can be performed in a wide region, but it requires large memory for storing them (see Subsection 3.F).

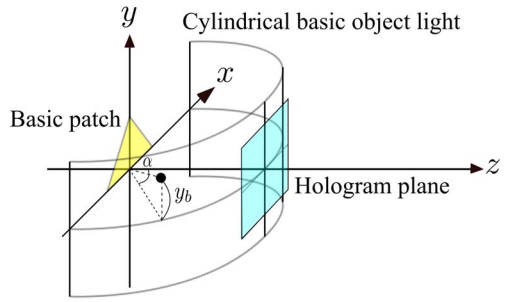
A small object creates a light that is close to a spherical light. The basic patch is enough small to apply the approximation. Consequently, light on a spherical surface centered on the basic patch does not have high spatial frequency, and there are highly angled positions on the spherical surface. The calculation of the object light that propagates from the basic patch and is recorded on a spherical surface produces less data. However, a coordinate system on a spherical surface has unequally spaced sampling and singular points, making it difficult to use.

To solve these problems, we propose a method using the cylindrical basic object light (CBOL), which reduces the data size and enlarges the transform region. As shown in Fig. 4, the CBOL is the calculated object light, that propagates from the basic patch, and is recorded on a cylindrical surface instead of a planar surface. Using CBOL enables tilt transform without restriction in every region, which keeps the data size small.

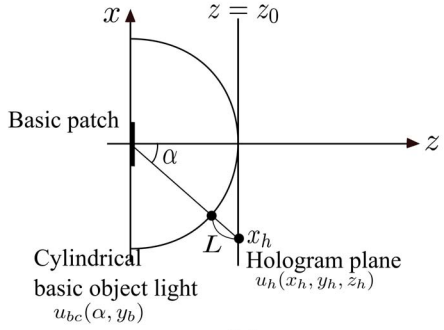
The light distribution $u_{bc}(\alpha, y_b)$ of CBOL is expressed in cylindrical coordinates and Eq. (2) is rewritten in cylindrical coordinates (α, y_b) as follows:

$$\begin{aligned} u_{bc}(\alpha, y_b) = {} & \frac{j}{\lambda z_0 \cos\alpha} \exp\left\{-jk\left(z_0 \cos\alpha + \frac{R^2 \sin\alpha^2 + y_b^2}{2 z_0 \cos\alpha}\right)\right\} \\ & \times \int\int_{-\infty}^{\infty} g(\hat{x}, \hat{y}, 0) \\ & \times \exp\left(jk \frac{u z_0 \sin\alpha + v y_b}{z_0 \cos\alpha}\right) \mathrm{d}\hat{x}\mathrm{d}\hat{y}, \end{aligned} \tag{6}$$

where



(a)



(b)

Fig. 4. (Color online) Cylindrical basic object light.

$$\alpha = \tan^{-1}\left(\frac{x_h}{z_0}\right),$$

and $g(\hat{x},\hat{y},0)$ is the object's distribution of the basic patch in Fig. 2.

Equation (2) is rewritten using CBOL u_{bc} as follows:

$$\begin{aligned} u_h(x_h,y_h,z_0) &= \frac{j}{\lambda z_0}\exp(-jkz_0)\int\int_{-\infty}^{\infty} g(\hat{x},\hat{y},0) \\ &\quad\times \exp\left\{j\frac{k}{z_0}[(x_h-\hat{x})^2+(y_h-\hat{y})^2]\right\}\mathrm{d}\hat{x}\mathrm{d}\hat{y} \\ &= C\exp(-jkL)u_{bc}(\alpha, y_h\cos\alpha), \end{aligned} \tag{7}$$

where C and L are transform parameters. This equation means that light wave distribution u_h is calculated by multiplying CBOL u_{bc} by a phase with the difference of L and a constant C. 3D affine transformations are performed by selecting appropriate transform parameters C and L and sampling positions $(\alpha, y_h\cos\alpha)$ on the CBOL. Equation (7) corresponds to Eqs. (13) and (15)–(18) reported in [6]; however, parameters C and L and sampling positions $(\alpha, y_h\cos\alpha)$ are different, because the CBOL is calculated on a cylindrical surface. The transform parameters are described in Subsections 3.A–3.E, and are derived from Eqs. (6) and (7).

A. Slide and Rotation Transform

In a Cartesian coordinate system with x, y, and z axes, the slide transform is performed along the x and y axes and the rotation transform is performed around the z axis. When the amount of slide is Δx and Δy, and the amount of rotation is θ, the object light of the transformed polygon is expressed as

$$\begin{aligned} u_h(x_h,y_h,z_0) &= \frac{j}{\lambda z_0}\exp(-jkz_0)\int\int_{-\infty}^{\infty} g(\hat{x},\hat{y},0) \\ &\quad\times \exp\left\{j\frac{k}{z_0}[(x'_h-\hat{x})^2+(y'_h-\hat{y})^2]\right\}\mathrm{d}\hat{x}\mathrm{d}\hat{y} \\ &= C_1\exp(-jkL_1)u_{bc}(\alpha, y'_h\cos\alpha), \end{aligned} \tag{8}$$

where

$$\begin{aligned} &C_1 = 1, \qquad \alpha = \tan^{-1}\left(\frac{x'_h}{z_0}\right), \\ &x'_h = x_h\cos\theta + y_h\sin\theta - \Delta x, \\ &y'_h = y_h\cos\theta + x_h\sin\theta - \Delta y, \\ &L_1 = (x'^2_h + y'^2_h + z_0^2)^{1/2} - [(z_0\sin\alpha)^2 + (y'_h\cos\alpha)^2 \\ &\qquad + (z_0\cos\alpha)^2]^{1/2}. \end{aligned}$$

This equation shows that slide and rotation transforms are performed by resampling the data on the CBOL in accordance with $(\alpha, z_0\cos\alpha)$ and multiplying by the phase with respect to L_1 and C_1.

B. Distance Transform

When a transformed polygon is Δz away on the z axis from the basic patch, the object light of the transformed polygon is expressed as

$$\begin{aligned} u_h(x_h,y_h,z_0+\Delta z) &= \frac{j}{\lambda(z_0+\Delta z)}\exp(-jk(z_0+\Delta z)) \\ &\quad\times\int\int_{-\infty}^{\infty} g(\hat{x},\hat{y},0) \\ &\quad\times\exp\left\{j\frac{k}{z_0+\Delta z}[(x_h-\hat{x})^2 \right. \\ &\quad\left. +(y_h-\hat{y})^2]\right\}\mathrm{d}\hat{x}\mathrm{d}\hat{y} \\ &= C_2\exp(-jkL_2)u_{bc} \\ &\quad\times\left(\alpha, y_h\frac{z_0\cos\alpha}{z_0+\Delta z}\right), \end{aligned} \tag{9}$$

where

$$\begin{aligned} &C_2 = \frac{z_0}{z_0+\Delta z}, \\ &\alpha = \tan^{-1}\left(\frac{x_h}{z_0+\Delta z}\right), \\ &L_2 = [x_h^2 + y_h^2 + (z_0+\Delta z)^2]^{1/2} \\ &\qquad - \left[(z_0\sin\alpha)^2 + \left(y_h\frac{z_0\cos\alpha}{z_0+\Delta z}\right)^2 + (z_0\cos\alpha)^2\right]^{1/2}. \end{aligned}$$

The object light propagated from a distance-transformed polygon is calculated in the same manner as described in Subsection 3.A, except for resampling of the data on the CBOL in accordance with $(\alpha, \frac{z_0 y_h \cos\alpha}{z_0} + \Delta z)$. In the remaining transforms, as described below, the CBOL is transformed by resampling and by multiplying itself by the phase and the constant.

C. Tilt Transform

Tilt transform is achieved by rotating the CBOL around the y axis. When the basic patch is rotated by angle ϕ around the y axis, the object light of the transformed polygon is expressed as

$$u_h(x_h, y_h, z_0) = C_3 \exp(-jkL_3) u_{bc}(\alpha - \phi, y_h \cos(\alpha - \phi)), \tag{10}$$

where

$$C_3 = 1,$$
$$\alpha = \tan^{-1}\left(\frac{x_h}{z_0}\right),$$
$$L_3 = (x_h^2 + y_h^2 + z_0^2)^{1/2} - [(z_0 \sin\alpha)^2 + (y_h \cos\alpha)^2 + (z_0 \cos\alpha)^2]^{1/2}.$$

D. Scaling Transform

When the basic patch is scaled by R_x on the x axis and R_y on the y axis, the scaled polygon is represented as

$$g_{\text{scale}}(\hat{x}, \hat{y}, 0) = g\left(\frac{\hat{x}}{R_x}, \frac{\hat{y}}{R_y}, 0\right).$$

Substituting the scaled polygon into Eq. (7) changes the expression for the object light of the transformed polygon to

$$\begin{aligned} u_h(x_h, y_h, z_0) &= \frac{j}{\lambda z_0} \exp(-jkz_0) \int\int_{-\infty}^{\infty} g_{\text{scale}} \\ &\quad \times (\hat{x}, \hat{y}, 0) \\ &\quad \times \exp\left\{j \frac{k}{z_0}[(x_h - \hat{x})^2 + (y_h - \hat{y})^2]\right\} \mathrm{d}\hat{x}\mathrm{d}\hat{y} \\ &= C_4 \exp(-jkL_4) u_{bc}(\alpha, y'_h \cos\alpha), \end{aligned} \tag{11}$$

where

$$C_4 = R_x R_y, \qquad \alpha = \tan^{-1}\left(\frac{x'_h}{z_0}\right),$$
$$x'_h = R_x x_h, \qquad y_h = R_y y_h,$$
$$L_4 = (x_h^2 + y_h^2 + z_0^2)^{1/2} - [(z_0 \sin\alpha)^2 + (y'_h \cos\alpha)^2 + (z_0 \cos\alpha)^2]^{1/2}.$$

E. Skew Transform

When the skew factor is S_x along the x and y axes, the skewed polygon is represented as

$$g_{\text{skew}}(\hat{x}, \hat{y}, 0) = g(\hat{y} S_x + \hat{x}, \hat{y}, 0).$$

Substituting the skewed polygon into Eq. (7) changes the expression for the object light of the transformed polygon to

$$\begin{aligned} u_h(x_h, y_h, z_0) &= \frac{j}{\lambda z_0} \exp(-jkz_0) \int\int_{-\infty}^{\infty} g_{\text{skew}}(\hat{x}, \hat{y}, 0) \\ &\quad \times \exp\left\{j \frac{k}{z_0}[(x_h - \hat{x})^2 + (y_h - \hat{y})^2]\right\} \mathrm{d}\hat{x}\mathrm{d}\hat{y} \\ &= C_5 \exp(-jkL_5) u_{bc}(\alpha, y'_h \cos\alpha), \end{aligned} \tag{12}$$

$$C_5 = 1, \qquad \alpha = \tan^{-1}\left(\frac{x'_h}{z_0}\right),$$
$$x'_h = x_h, \qquad y'_h = y_h + S_x x_h,$$
$$L_5 = (x_h^2 + y_h^2 + z_0^2)^{1/2} - [(z_0 \sin\alpha)^2 + (y_h \cos\alpha)^2 + (z_0 \cos\alpha)^2]^{1/2}.$$

The object light of an arbitrary polygon is calculated by combining these transforms, and superposition of the object light from each polygon produces an all-inclusive object light on a hologram.

In this method, the tilt transform around the y axis is unlimited because of the use of the cylindrical surface centered on the y axis, as shown in Fig. 4. However, the tilt transform around the x axis is limited as it is in the method using plane basic object light. This problem is overcome by combining the rotation transform in Subsection 3.A with the tilt transform in Subsection 3.C.

F. Data Size

As mentioned, the method using plane basic object light has a limitation of angle about the tilt transform. The maximum transform angle using one plane basic object light is represented as

$$\phi_{\max} = \tan^{-1}\frac{n\Delta d}{z_0} - \tan^{-1}\frac{n\Delta d}{2z_0}, \tag{13}$$

where Δd is the sampling pitch on the hologram and the plane basic object light, and n is a pixel on the side of the hologram. As a result, to tilt at a ± 90 deg, the required number of plane basic object lights is

$$\frac{\pi}{2\phi_{\max}}. \tag{14}$$

The data size of one plane basic object light is $4n^2$ because the height and width are double those of the hologram, for transforms. Therefore, the total data size of the plane basic object light is given by

$$\frac{2n^2\pi}{\phi_{\max}}. \tag{15}$$

In the CBOL method proposed, the height of the CBOL is $2n$, which is the same as that of the plane basic object light. The arc of the cylinder is $z_0\pi/\Delta d$, where z_0 is the distance between the basic patch and the surface, i.e., the radius of the cylinder. Consequently, the data size of the CBOL is expressed by

$$\frac{2\pi z_0}{\Delta d}n. \tag{16}$$

Equations (15) and (16) show that the use of the CBOL reduces the data size in comparison with that of the plane basic object lights. In the case that z_0 is sufficiently larger than the basic patch, the data size of the CBOL is half that of the plane basic object light.

4. Implementation on GPU

A. GPU

GPUs are now being used for not only image output but also a general-purpose scientific calculation due to the GPU's parallel processing capability. This is referred to as general-purpose computing on graphics processing units (GPGPU). A GPU has a highly parallel structure, containing multiple cores, which enables the GPU to process a lot of data with parallel dispatch (Fig. 5). The calculation method using the CBOL is faster when parallel processing is used. It is performed in parallel, and GPUs perform parallel calculations efficiently.

We use the compute unified device architecture [11,12], a parallel computing architecture for GPGPU by NVIDIA, to accelerate the calculations for making holograms using GPU. It has four main steps.

1. Allocate memory space to be used by the CPU and the GPU.
2. Copy CBOL data and transform parameters into GPU memory.
3. Compute the data in the GPU memory using a transform expressed as Eq. (7).
4. Copy data from the GPU memory to main memory, for use in generating hologram data.

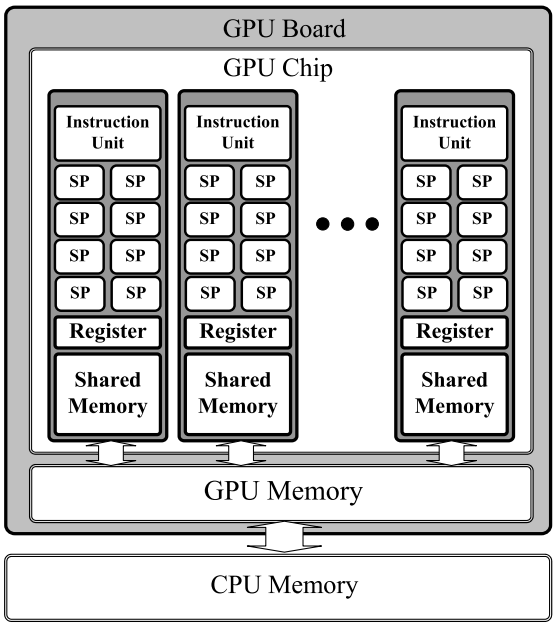


Fig. 5. GPU structure.

B. Calculations to Parallelize

In step 3 above, parallelization is performed at the pixel level (x_h, y_h, z_0). The multiple cores in the GPU simultaneously calculate a pixel of the hologram, and this parallel processing accelerates the whole calculation.

The transform calculation process is distributed to each core, as shown in Fig. 6. The data for the CBOL and the transform parameters are copied from main memory to GPU memory before transform calculation. The calculation process includes eight steps.

1. Initialize registers to zero, to store the light distribution on hologram $u_h(x_h, y_h, z_0)$.
2. Assign the transform parameters of a polygon to each core.
3. Have each core calculate coordinates $(\alpha, y_h \cos\alpha)$ of the CBOL and constant number C using hologram coordinates (x_h, y_h, z_0) and transform parameters.
4. Have each core calculate the phase difference $\exp(-jkL)$.
5. Have each core refer the CBOL $u_{bc}(\alpha, y_h \cos\alpha)$ and multiply $u_{bc}(\alpha, y_h \cos\alpha)$ by the phase difference.
6. Make each core add the calculation result of step 5 to register of the appropriate pixel.
7. Repeat steps 2–6 for all polygons.
8. Save the results in the registers to GPU memory.

C. CBOL Division

The calculation method using the CBOL with GPUs requires that the CBOL data must be copied into

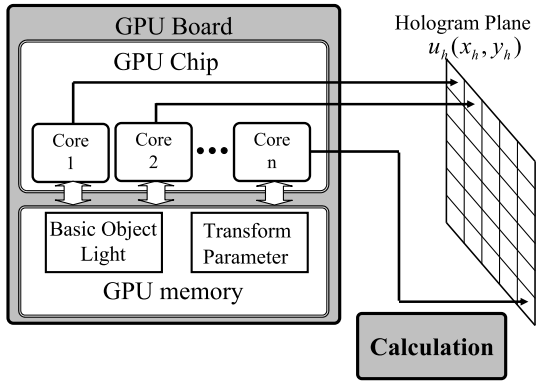


Fig. 6. Data flow on GPU board.

GPU memory in advance. However, the data of CBOL is too much to be stored in the GPU memory at one time. Thus, we need to divide the CBOL into smaller parts called "segments."

Only the data for one segment can be placed on GPU memory at one time. When several cores need more than one segment for calculating the object light of a polygon, they must wait until another segment is loaded into GPU memory. This reduces computational efficiency. This problem is solved by minimizing the number of segments that the cores need for a calculation with respect to one polygon. The CBOL is thus divided along vertical lines with the central angle of the cylinder (Fig. 7). Dividing along horizontal lines would leave several cores often needing more than one segment for transform calculation. Vertical division thus results in more efficient computation.

D. Overall Process

The method using the CBOL with a GPU is processed with copying the segments sequentially. The whole process of proposed method is as follows.

1. Extract transform parameters from each polygon and copy them to GPU memory. Calculate transform parameters of each polygon by using the patch position, shape, and tilt.
2. Copy one segment into GPU memory.
3. Calculate transform as explained in Subsection 4.B.
4. Repeat steps 2–4 for each segment.
5. Copy hologram data $u_h(x_h, y_h, z_0)$ to main memory from GPU memory.

The time required for copying the data to GPU memory exceeds the computation time. Consequently, it becomes a bottleneck. Therefore, the number of segments to be copied must be minimized to shorten the total computation time. Steps 1–5 above describe a process that minimizes the number.

5. Experiments

We carried out optical experiments to verify the validity of the proposed transform method using the CBOL, and we implemented it on a GPU to determine its effect on calculation time and data size. The parameter values used in the experiments are listed in Table 1.

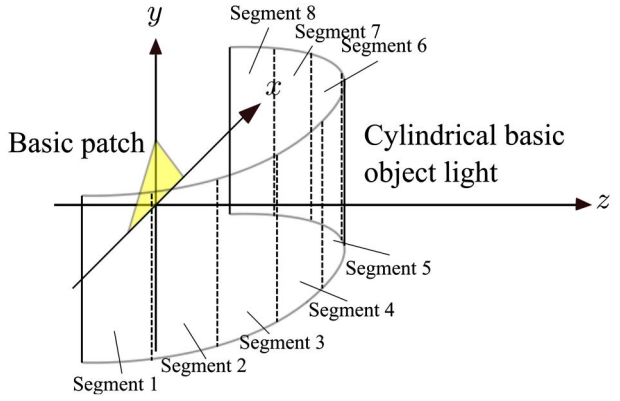


Fig. 7. (Color online) Division of cylindrical basic object light.

A. Data Size of Basic Object Light

With the calculation method using plane basic object light, the basic object light needs to be twice the hologram size in both width and height. In our experiments, we used 2048 × 2048 pixel holograms, so the basic object light was 4096 × 4096 pixels. When a pixel is expressed in 8 bytes, the data size of a basic object light is 128 Mbytes.

As explained in Subsection 2.B, tilt transform is performed in a narrow region. When using one plane basic object light with the hologram parameters used in our optical experiments, the tilt transform is restricted to ±2.4 deg. Therefore, to cover the range of ±90 deg, 38 planes of basic object light are needed. Since the data size for one plane basic object light is 128 MBytes, the total data size for all plane basic object lights is 4.75 GBytes.

In the proposed method, the CBOL data size obtained using Eq. (16) is 2.26 GBytes for a range of ±90 deg degrees, for which the width and height are 74,210 × 4096. The actual data size on a computer is equivalent to the theoretical one. This is about half the amount of the data when using plane basic object light. The ratio of data sizes is in agreement with the theoretical prediction in Subsection 3.F.

B. Optical Experiments

For our optical experiments, holograms were made as black-and-white binary images and then printed on transparent sheets. A red light-emitting diode was used to reconstruct the images from the holograms, and all the viewpoints of the holograms were positioned across the hologram plane from the reconstructed objects.

Table 1. Parameter Settings for Experiments

Common Parameters	
Sampling pitch Δd	6.35 [μm]
Wavelength λ	632 [nm]
Basic patch	
Shape	Equilateral triangle
Size	3.2 [mm] per side
Plane basic object light	
Number of pixels	4096 × 4096 [pixel]
Size	26 × 26 [mm]
Distance z_0	0.15 [m]
Number of planes	38
CBOL	
Number of pixels	74210 × 4096 [pixel]
Radius z_0	0.15 [m]
Height	26 [mm]
Number of segments	9
Number of segment pixels	8256 × 4096
Hologram	
Number of pixels	2048 × 2048 [pixel]
Size	13 × 13 [mm]

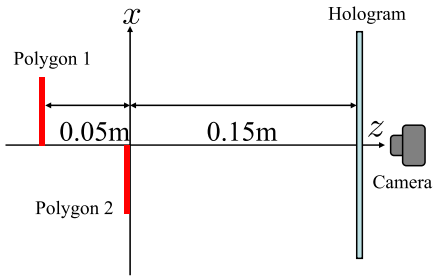


Fig. 8. (Color online) Setup.

Figure 9 shows the basic patch and four typically transformed patches, i.e., distance, tilt, scaling, and skew. The basic patch from the CBOL is shown in Fig. 9(a); it was not transformed. Figures 9(b) and 9(c) show the results of the distance transform, in which a slid basic patch was placed as shown in Fig. 8. In Fig. 9(b), the camera focused on the left polygon. The left polygon has a sharp outline and the others do not, and Fig. 9(c) shows the exact opposite. These results show that the distance transform was performed correctly. In Figs. 9(d)–9(f), the objects were transformed by tilt, scaling, and skew transforms with $\phi = 60$, $R_x = 1.2$, and $S_x = 0.8$. These results shows that the transforms were performed correctly.

Figure 10 shows an example reconstruction using multiple transforms. A star-shaped object with ten polygons was used. The object image rendered by the computer graphics technique is shown in Fig. 10(a). The reconstructed image shown in Fig. 10(b) is similar to the computer-generated image. This indicates that using the five transforms and their combination is effective.

C. Measurement of Computation Time

We implemented the proposed method to determine the effect of using a GPU on the CGH calculation time. We used a Xeon X5492 3.40 GHz CPU and a GeForce 9800 GT GPU, which has 512 Mbytes of memory. The CBOL was divided into nine segments to enable a segment to be stored in GPU memory at one time. The data size of a segment is 258 MBytes. The computation times for one polygon with the proposed and conventional methods are shown in Table 2. The shortest time was obtained for the proposed method: 3D affine transformation using a GPU. Note that the times in the table are only the transform calculation times; they do not include the time for copying segments from main memory to GPU memory. The copying time for a segment was 400 ms. However, a segment can be used to calculate more than one polygon, so a segment is copied only once. As a result, the calculation time increased with only the number of polygons.

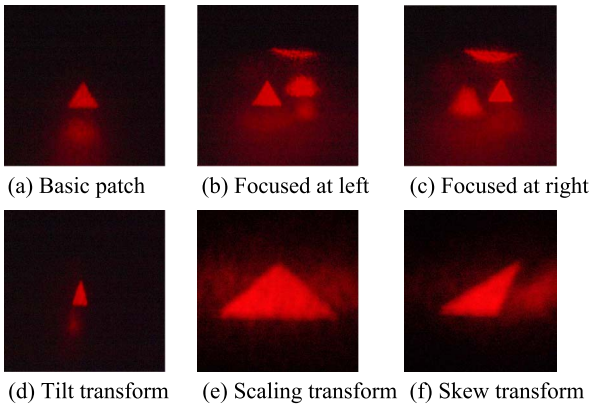


Fig. 9. (Color online) Reconstructed images.

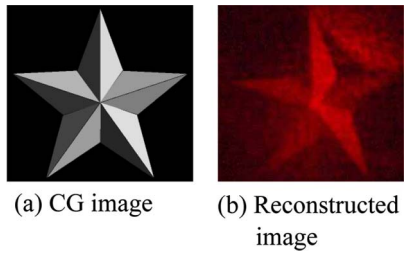


Fig. 10. (Color online) Images of combined transform.

Table 2. Computation Times for Evaluated Methods

Calculation Method for CGH	Calculation Time [ms]
Point light method on CPU	908,000
Fourier-transform-based method on CPU	2400
Proposed method on CPU	650
Point light method on GPU	265,000
Fourier transform-based method on GPU	65
Proposed method on GPU	10

Figure 11 shows the increase in computation time with the number of polygons. Because the calculations for all polygons used the same segment, a segment was copied to GPU memory only once, as mentioned above. This means the calculation time became shorter from the second polygon. As shown in Fig. 11, the GPU calculation was 65 times as fast as the CPU calculation when 100 polygons were calculated. This indicates that calculation using a GPU is more efficient when an object has more polygons. Since the calculation time depends on the parameter values, the results in Fig. 11 would change with the complexity of the target object.

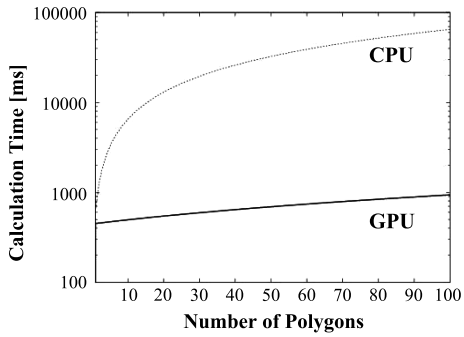


Fig. 11. Time increase against number of polygons.

6. Conclusion

Our proposed method for calculating aCGH uses CBOL and 3D affine transformation without FFT. The CBOL has a data size that is 50% of that of the plane basic object lights for our experimental parameters. We also proposed using a GPU for parallel calculation to increase computation speed. Calculation using the CBOL has high concurrency, so it is well suited for parallel computation using a GPU. Experimental results showed that using a GPU made the processing noticeably faster than using a CPU. A bottleneck occurs when the data is copied from main memory to GPU memory, particularly when the CBOL is very large. It indicates that reducing the data size by using CBOL reduces the copying time. As a result, computation speed is increased by using parallel calculation and the CBOL.

References

1. K. Matsushima, H. Schimmel, and F. Wyrowski, "Fast calculation method for optical diffraction on tilted planes by use of the angular spectrum of plane waves," J. Opt. Soc. Am. A **20**, 1755–1762 (2003).
2. L. Ahrenberg, P. Benzie, M. Magnor, and J. Waston, "Computer generated holograms from three dimensional meshes using an analytic light transport model," Appl. Opt. **47**, 1567–1574 (2008).
3. H. Yoshikawa, T. Yamaguchi, and R. Kitayama, "Real-time generation of full color image hologram with compact distance look-up table," in *Digital Holography and Three-Dimensional Imaging*, OSA Technical Digest (CD) (Optical Society of America, 2009), paper DWC4.
4. S. Kim and E. Kim, "Fast computation of hologram patterns of a 3D object using run-length encoding and novel look-up table methods," Appl. Opt. **48**, 1030–1041 (2009).
5. Y. Sakamoto and T. Nagao, "A fast computational method for computer-generated Fourier hologram using patch model," Electron. Commun. Jpn. Part 2 **85**, 16–24 (2002).
6. H. Sakata and Y. Sakamoto, "Fast computation method for Fresnel hologram using three-dimensional affine transformations in real space," Appl. Opt. **48**, H212–H221 (2009).
7. N. Masuda, T. Ito, T. Tanaka, A. Shiraki, and T. Sugie, "Computer generated holography using a graphics processing unit," Opt. Express **14**, 603–608 (2006).
8. R. H.-Y. Chen and T. D. Wilkinson, "Computer generated hologram with geometric occlusion using GPU-accelerated depth buffer rasterization for three-dimensional display," Appl. Opt. **48**, 4246–4255 (2009).
9. Y. Pan, X. Xu, S. Solanki, X. Liang, R. B. A. Tanjung, C. Tan, and T.-C. Chong, "Fast CGH computation using S-LUT on GPU," Opt. Express **17**, 18543–18555 (2009).
10. T. Shimobaba, T. Ito, N. Masuda, Y. Ichihashi, and N. Takada, "Fast calculation of computer-generated-hologram on AMD HD5000 series GPU and OpenCL," Opt. Express **18**, 9955–9960 (2010).
11. http://www.nvidia.com/.
12. http://developer.nvidia.com/category/zone/cuda-zone.