3DTV Conference proceedings, IEEE ISBN 978-1-4244-0722-4, 2007

FULL-PARALLAX HOLOGRAM SYNTHESIS OF TRIANGULAR MESHES USING A GRAPHICAL PROCESSING UNIT

Ivo Hanák, Martin Janda, and Václav Skala

University of West Bohemia
Department of Computer Science and Engineering
Univerzitní 8, Pilsen, 306 14, Czech Republic

ABSTRACT

Application of the GPU to the computer generated holography is a topic of research for some time. While the majority of authors aim on performance, we aim on visual aspects. We present a new approach that is capable to synthesise a hologram of a scene described by triangles using the GPU and it is capable to respect a local intensity variation on a surface caused by textures and solve occlusion at the same time.

Index Terms— Holography, Computer graphics, Rendering

1. INTRODUCTION

Computer generated holography (CGH) is a research topic for some time. Even thought the computational power is increasing, the generation of the full-parallax hologram is a problem on available computation power of an off-shelf computer today. At the same time, a graphical processing unit (GPU) offers a computational power that is greater than the comparable CPU. This encouraged attempts on general purposed processing on the GPU.

Currently, there are several approaches that deals with the CGH using the GPU [1, 2]. These solutions provides almost real-time hologram synthesis. Yet, they trade the performance for the visual quality as they assume that the scene consists of low number of points or lines. The solid surface is out of reach for them as the number of surface elements required to gain an impression of a solid surface is far too high. The approach presented in this paper deals with the CGH with the aim on the visual quality of the output. We are able to render surface and to solve the occlusion, too. In order to speedup the process we use a programmable GPU.

This work was supported by EU 3DTV NoE project No. 511568 and by MSMT CR project CPG No. LC06008. We would like to thank prof. L. Onural from Bilkent University for help and for valuable advises. Used models are based on data from the Stanford 3D Scanning Repository.

1.1. Utilized CGH model

For purposes of our approach we made following assumptions and restrictions. First, we assume that surfaces are decomposable to point sources that are ideal and thus their complex amplitude at the source is not influenced by other point sources in the scene. Second, a point on a surface has a complex amplitude with a phase equal to zero. As proposed in [3] this shall not harm the hologram. Third, a point on a surface has to be aligned to a distance $n\lambda$, where n is integer and λ is a recording wavelength. According to numerical experiments, only such points allows an arbitrary tilted plane to be reconstructed properly without high angular sampling rate [4].

The approach treats the scene as a composition of large point source set and the final diffraction pattern is computed as a superposition of individual point sources defined as:

$$\tilde{u}(r; A) = \frac{A}{r} \exp(-jkr), \tag{1}$$

where k is wavenumber, r is a distance to point source, and intensity of the point source is estimated according to numerical models for local lighting known from the computer graphics. The numerical verification of results is done by numerical simulation of wave propagation in angular spectrum [5, 6].

1.2. Graphics processing unit

The GPU is an implementation of a basic pipeline for rendering based on triangles. Certain parts of the GPU are customisable by a user supplied code. Thanks to its computational power based on parallelism and limitations, such as restrictions on random-write operation, limited maximum offset for read operations, lower accuracy, and lack of persistent memory allocation, it is possible to transfer certain tasks from the CPU to the GPU and gain a speedup that depends on a nature of the given algorithm. Currently, GPUs have two major programmable blocks: the vertex shader and the pixel shader. The vertex shader takes individual vertices, process them, and passes them to the clipping engine and rasterizer. The pixel shader computes color of surface elements generated by rasterizer. As the pixel shader is capable to write the result of

the processing to a memory array, it is usually a place for the major computational kernel.

2. THE PROPOSED APPROACH

Our solution works for horizontal parallax only (HPO) holograms as well as for full parallax holograms. Thus, let us first describe the solution for the HPO case, the extension to the full parallax version is straightforward. If a scene consists of a single point source then each sample on a line of diffraction pattern contains a complex amplitude according to the Eq. (1). Increasing number of points leads to the solution that is almost the same. The only difference is higher number of contributions that has to be summed to gain a sample of the diffraction pattern. Increasing of the number of point source increases the scene complexity until the situation when the scene consists of solid surfaces. Yet, in such case the number of points is too high for computation time to be reasonable. Thus, more effective approaches has to be applied by use of wave propagation [7, 6] or even more effectively by use of precomputed patterns [8]. Even thought, the latter approach is effective, it cannot handle occlusion and local variations on the surface and/or line that includes a texture. And that is a place where our solution fits in.

First of all, our solution uses orthogonal projection because perspective is only a side-effect of a lens. Basically there are two geometrical transformations that causes this effect: a rotation and a slanting. If the scene consist of a single line parallel to the X-axis then by application of both transformation we obtain different results, see Fig. 1a and Fig. 1b. While the rotation causes non-uniform size of individual rays that were of the same size before, the slanting does not exhibit such behavior. Thus, we used slanting in a form of a transformation matrix in left-handed coordinate system:

$$x_t = x - z\tan\xi,\ \xi \in (-\pi/2; \pi/2), \tag{2}$$

where x_t is transformed x-axis coordinate of point $\mathbf{v}$ for rays of given angle ξ.

It can be seen in the Fig. 1 that the length of transformed ray is related to the original one. The difference is caused by a scaling along the Z-axis. Thus, the complete transformation for the HPO case is:

$$\mathbf{v_t} = \mathbf{v}\begin{bmatrix} 1 & 0 \\ -\tan\xi & 1/\cos\xi \end{bmatrix}, \mathbf{v} = [x; z], \tag{3}$$

where $\mathbf{v}$ is an original point and $\mathbf{v_t}$ is a transformed one that is projected by orthogonally later. Note, that the transformation does not cause non-linear deformations and thus it is possible to benefit from capabilities of the GPU to perform linear interpolation along line/surface. After the vertex $\mathbf{v}$ is transformed, the X-coordinate corresponds to an address of the sample on a hologram line and the Z-coordinate is substituted as distance r to the Eq. (1) in order to obtain a contribution. Intensity A is computed according to a local lighting models [9]. Contributions of individual angles ξ are summed together to form a sample of a final diffraction pattern.

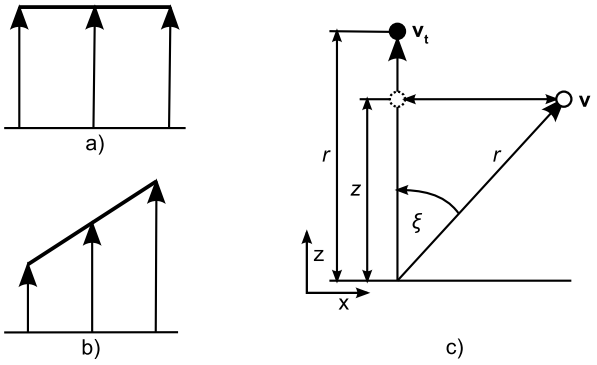

Fig. 1. Effect of a rotation transformation (a) and a slanting transformation (b) on a line parallel to X-axis and a slanting transformation in detail (c).

2.1. Extension to full-parallax

Full-parallax extension of the HPO approach applies the same equation for the Y-coordinate as well as for the X-coordinate. The only complication is a proper scaling along the Z-axis. The scale coefficient can be computed as size of a vector $\mathbf{v} = \mathbf{v_x} + \mathbf{v_y} + \mathbf{v_z}$, where $|\mathbf{v_x}| = z^2\tan^2\xi_x$ and $|\mathbf{v_z}| = z^2$. Adding a second angle ξ_y to an already existing angle ξ_x yields a transformation:

$$\mathbf{v}_t = \mathbf{v}\begin{bmatrix} 1 & 0 & -\tan\xi_x \\ 0 & 1 & -\tan\xi_y \\ 0 & 0 & (1+\tan^2\xi_x+\tan^2\xi_y)^{\frac{1}{2}} \end{bmatrix}^{\mathrm{T}}, \tag{4}$$

where $\mathbf{v} = [x; y; z]$ and $\mathbf{v}_t$ is point in the real-world coordinates that is projected orthogonally to the hologram. Note, that the transformation in the Eq. (4) is valid for a left-handed coordinate system with Y-axis up and Z-axis pointing towards the scene. The algorithm that computes a full-parallax hologram is almost the same as the HPO version. The only difference is that it contains two loops, each iterating different angle ξ_x and ξ_y independently.

2.2. Angle sampling considerations

The range of angles ξ_x and ξ_y can be estimated from an axis-aligned bounding box as it is depicted by the Fig. 2a. If the angle steps over this range, no sample of the diffraction pattern will be affected. The scene sampling is a discretized version of integration over a hemispherical domain for each diffraction pattern sample. Respective areas on the hemisphere have to be kept as close to uniformity as possible. Otherwise, a contribution has to be weighted. This is fulfilled even for a constant angular step if maximum angle is kept small, see

Fig. 2. As this is fulfilled for a proper minimum object distance due to currently available spatial light modulators we apply this scheme as well. If more suitable sampling is required it is possible to express it in term of angles ξ_x and ξ_y.

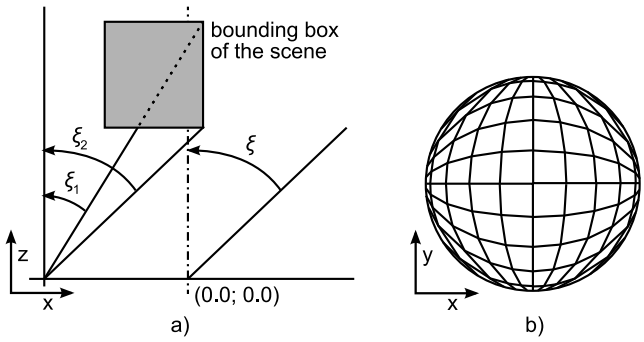

Fig. 2. Evaluation of a maximum sampling angle $\xi = \max(\xi_1, \xi_2)$ for right side (a) and hemisphere from a top-view sampled by a constant step in both angles (b).

3. IMPLEMENTATION

The implementation on the GPU is rather straightforward. Yet, it has few difficulties that have to be avoided in order to let the algorithm to behave properly and give reasonable results. The actual implementation is done by use of the Direct3D 9.0c interface and the GPU of shader model (SM) 2.0.

3.1. Numerical accuracy

The accuracy is crucial for computation of phase of the hologram [10] and thus a high error in phase of a single contribution causes a significant degradation of the diffraction pattern. The contribution is a complex number based on the Eq. (1). The argument of the contribution is a distance r divided by the wavelength λ but only its fraction part is required. As $1/\lambda \approx 10^6$ and $r < 10^0$ the integer part of the result is $10^6 \approx 2^{20}$. Thus an unnecessary integer part requires 20 bits but on SM 2.0 a mantissa is guaranteed to have only 16 bits. Even SM 3.0 has only 23 bits reserved for mantissa.

This numerical accuracy is bypassed by replacing the Z-coordinate by z/λ with the result slit up into an integer and a fractional part. The integer part has to be split up once more as it does not fit into 16-bit mantissa, the fraction part has acceptable accuracy because it is used only for rounding of the Z-coordinate to nearest multiply of λ that is greatly simplified by the decomposition.

Also, for the HPO version we found out that it is difficult to evaluate $1/\cos(\xi)$ from the Eq. (3) precisely enough for a very small angles as the cosine changes very slowly. The same problem was detected for full-parallax Z-coordinate scale factor from the Eq. (4). As the angle ξ is small, we applied Taylor expansion around zero together with decomposition scheme similar to the Z-coordinate.

In another words, both Z-coordinate and the Z-coordinate scale factor are computed and stored in form of vectors. In the case of the scale factor the fractional part is split further up because for the HPO hologram the integer part starts to grow for larger angles. The ideal boundary of splits is 8 bits because for 16-bit mantissa the multiply of two 8-bit numbers will not cause loss of bits. This leads to use of a four-component vector in order to get acceptable number of bits. Yet, according to our experiments, even a split boundary of 10 bits did not cause significant degradation in the result.

The sum of individual contributions did not shown any signs of accuracy problems as we did not see any disturbing artifacts after numerical reconstruction. Thus, a vector that consist of two components is used to store the resulting complex number.

3.2. Implementation structure

We decided to use GPU of the SM 2.0 as it is a standard now even for low-cost integrated GPUs. Due to that, the implementation requires four passes and two render target switches for a combination of angles ξ_x and ξ_y. Even thought the number of passes can be reduced to three by merging the second and the third pass by use of higher shader model, the number of render target switches stays the same. For GPU of SM 2.0 each iteration includes step:

1. Render depth buffer of the scene including a stencil buffer in order to benefit from GPU capability to skip surface elements that are hidden according to the depth buffer.

2. Compute phase and intensity of a complex amplitude for a contribution with aid of a depth buffer from previous step. Phase is stored in tree-component vector, intensity occupies only a single scalar value.

3. Compute contribution from phase and intensity, sum it with the diffraction pattern from the previous iteration, and store it into new diffraction pattern.

4. Copy diffraction pattern samples that were not modified by this step to the new diffraction pattern.

4. RESULTS

We implemented our solution in C# 2.0, Direct3D 9.0, and the HLSL, a language for the GPU. We tested our algorithm on a AMD Athlon 3200+ with a single NVIDIA GeForce FX 6800GT. The resolution of the diffraction pattern was set to 1024 samples as the GPU-friendly resolution. The object was centered around $200\mathrm{mm}$ with maximum depth averaging around $10\mathrm{mm}$. The constant step for angles ξ_x and ξ_y was

experimentally estimated to $1/300^{\circ}$ as the longest step that does not lead to appearance of false object reconstructions.

For a bunny model that consist of 16000 triangles with simple lighting and one texture applied the computation time was 8.1 hours. For more complex model with 96000 triangles the time was 11.8 hours and for a simple model of 64 triangles the was 6.0 hours. We assumed that last mentioned time is approximately the overhead of the GPU caused by multiple passes and render target switching for single combination of angles ξ_x and ξ_y. The resulting diffraction pattern was reconstructed numerically and for the bunny model, the results are shown in the Fig. 3.

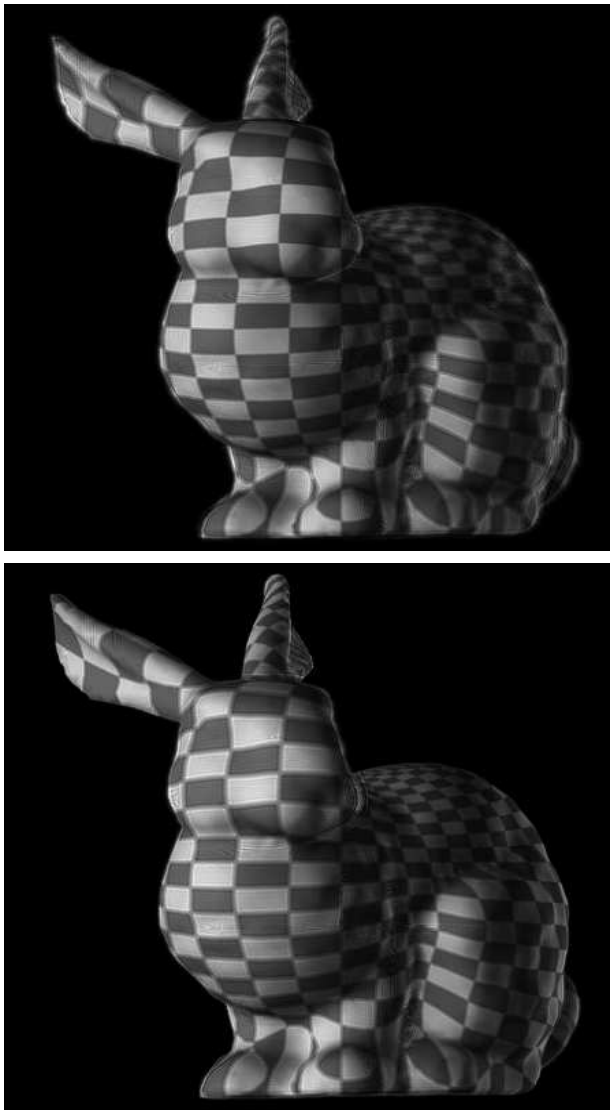

Fig. 3. Numerical reconstruction of diffraction pattern of a textured model focused on muzzle $z = 197\mathrm{mm}$ (top) and on body $z = 200\mathrm{mm}$ (bottom).

5. CONCLUSION

We presented a new approach that allows rendering of full parallax diffraction patterns for a scene defined by a triangular mesh. We can solve occlusion and apply local lighting and/or texture and we are compatible with compute graphics methods for triangle-based visualisation. Even though we ignore all effects caused by sharp edges we did not detected any visible degradation in the numerical reconstruction. Our approach is easily scalable and the performance can be improved by distributed computation. We expect that the relation between speedup and a number of computers/GPUs to be almost linear.

6. REFERENCES

[1] N. Masuda, T. Ito, T. Tanaka, A. Shiraki, and T. Sugie, "Computer generated holography using parallel commodity graphics hardware," *Optics Express*, vol. 14, no. 2, pp. 603–608, 2006.

[2] C. Petz and M. Magnor, "Fast hologram synthesis for 3d geometry models using graphics hardware," in *Practical Holography XVII and Holographic Materials IX*. SPIE, 2003, vol. 5005, pp. 266–275.

[3] M. Lucente, "Interactive computation of holograms using a look-up table," *J. El. Imag.*, vol. 2, pp. 28–34, 1993.

[4] M. Janda, I. Hanák, and V. Skala, "Digital HPO hologram rendering pipeline," in *EG2006 short papers conf. proc.*, 2006, pp. 81–84.

[5] G.B. Esmer and L. Onural, "Computation of holographic patterns between tilted planes," in *Holography 2005*. SPIE, 2006, vol. 6252, p. 62521K.

[6] J.W Goodman, *Introduction to Fourier Optics*, Roberts & Company Publishers, 3rd edition, 2005.

[7] H.P. Moravec, "3d graphics and the wave theory," in *SIGGRAPH '81*, 1981, pp. 289–296.

[8] M. Koenig, O. Deussen, and T. Strothotte, "Texture-based hologram generation using triangles," in *Practical Holography XV and Holographic Materials VII*. SPIE, 2001, vol. 4296, pp. 1–8.

[9] A. Watt, *3D Computer Graphics*, Addison-Wesley, 3rd edition, 2000.

[10] O. Matoba, T. J. Naughton, Y. Frauel, N. Bertaux, and B. Javidi, "Three-dimensional object reconstruction using phase-only information from a digital hologram," in *Three-Dimensional TV, Video, and Display.* SPIE, 2002, vol. 4864, pp. 122–128.