Rapid hologram generation utilising layer-based approach and graphic rendering for realistic 3D image reconstruction by angular tiling

Jhen-Si Chen[1], Daping Chu[1], Quinn Smithwick[2]

[1] University of Cambridge, Photonics and Sensors Group, Department of Engineering, 9 JJ Thomson Avenue, Cambridge, UK, CB3 0FA

[2] Disney Research, Glendale, CA, USA, 91201-5020

Abstract: An approach of rapid hologram generation for the realistic 3D image reconstruction based on the angular tiling concept is proposed, using a new graphic rendering approach integrated with a previously developed layer-based method for hologram calculation. A 3D object is simplified as layered cross-sectional images perpendicular to a chosen viewing direction, and our graphics rendering approach allows the incorporation of clear depth cues, occlusion and shading in the generated holograms for angular tiling. The combination of these techniques together with parallel computing reduces the computation time of a single-view hologram for a 3D image of XGA resolution to 176 ms using a single consumer graphics processing unit card.

Keywords: optics, digital holography, 3D images processing

Address all correspondence to: Daping Chu, University of Cambridge, Photonics and Sensors Group, Department of Engineering, 9 JJ Thomson Avenue, Cambridge UK, CB3 0FA; Tel: +44 (0) 1223 748352; Fax: +44 (0) 1223 748342; E-mail: dpc31@cam.ac.uk

1 Introduction

We recently introduced a multi-view layered holographic rendering algorithm[1] which combined layer based computation with multi-viewpoint rendering to rapidly compute full parallax holograms with occlusion and view-dependent shading. Layer-based methods should be more efficient than typical point-based methods because of the reduced calculation complexity and less amount of data involved. Multi-viewpoint rendering removes parallax artifacts and allows for a wider field-of-view, occlusion/disocclusion and view-dependent shading in layered holograms.

To realize real time computation for holographic image displays, both the production of the image source and the calculation of holograms from the image source are critical. Our previous work proposed the use of a layer-based method to reduce computational load for the step of hologram calculation, but did not provide a solution for efficient image source production. This work takes advantage of image based rendering from multiple viewpoints with angular tiling and an integrated graphics rendering pipeline to produce image sources efficiently. Combined with further optimized codes and overall performance of the algorithm, the effectiveness of our integrated approach of image source production and hologram calculation is demonstrated.

We will show that a rapid calculation speed of 4.5x10^6 pixels per second (pps, the number of hologram pixels calculated in a second), greater than typical point-based methods, can be achieved via such an approach by using a single consumer graphics processing unit (GPU) card for the generation of a 3D hologram of a complicated object. The resulting 3D views have clear depth cues, occlusion and shading.

The algorithm is also compatible with and takes advantage of our Coarse Integral Holographic Display (CIHD)[2]. An array of layered holograms of different viewpoints and with computed attached holographic lenses are angularly tiled using a single common transform lens. Real time rendering will allow us to generate interactive holograms, or achieve real time transmission and playback of holographic television on the CIHD.

2. Background review

2.1 Multi-view layered holograms

We proposed an efficient multi-view layer-based holographic algorithm[1] to take advantage of the computation speed of layer-based hologram generation while using angular multiplexing to overcome its limitations of diffusive Lambertian surfaces with limited field of view and parallax.

Our layer based method dissects a 3D point cloud into parallel 2D image layers, then uses a Fast Fourier Transform (FFT) of each layer and computed lens (with a focal length corresponding to the layer's depth) to create a diffractive pattern for each layer, which are then added together to create a Fourier- Fresnel hologram[3]. The Ping-Pong method was then used to include occlusion effects by computationally propagating light from the hologram plane forward with each plane's image silhouette acting as an occluder, then propagating the resultant light back to the hologram plane. The reconstructed 3D image has depth and accommodation cues. Unfortunately, layer-based holograms only support diffusive Lambertian surfaces and present a very limited field of view. The gaps between layers become apparent when the viewing angle is away from the axis normal to the parallel 2D image layers, as shown in Fig. 1.

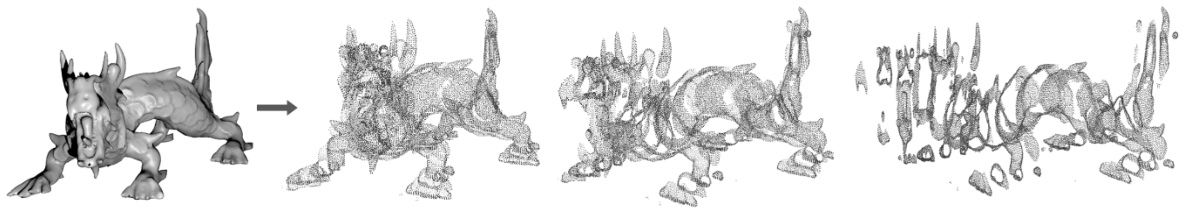


Fig. 1 Illustration of slicing and its gap issue.

To overcome these limitations, the layer-based method was extended to use render multiple viewpoints, thus handling occlusion/disoclussion, view-dependent lighting, and realigning the slicing of layers for each viewpoint. The multi-view layered holographic rendering algorithm was adapted to a CIHD system[2] which is illustrated in Fig. 2, where holograms for a 3D object viewed at different angles are replayed through a coarse integral imaging system to form realistic 3D views through angular tiling.

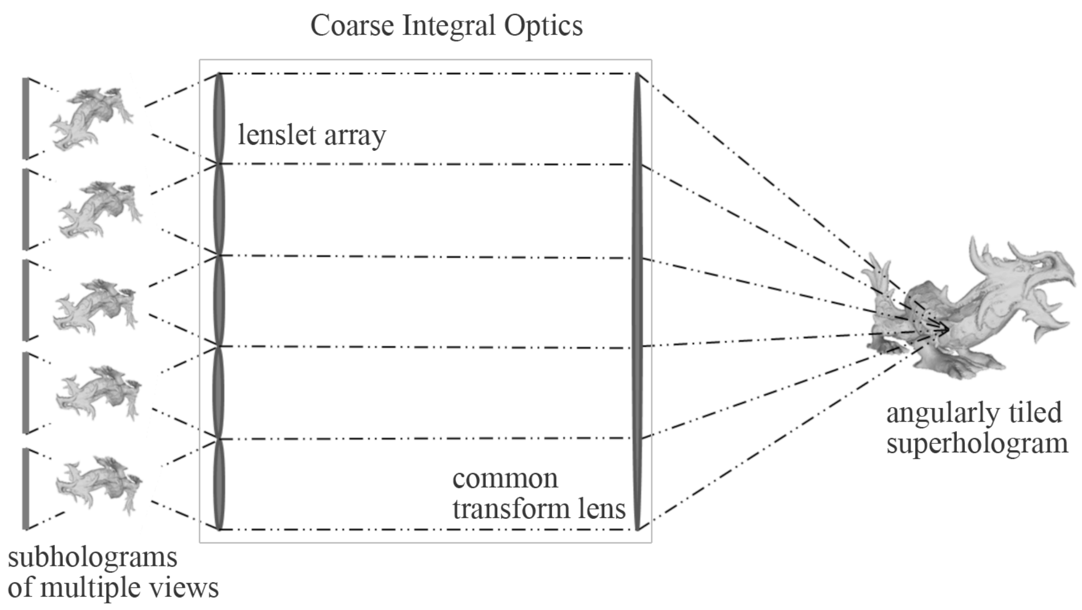
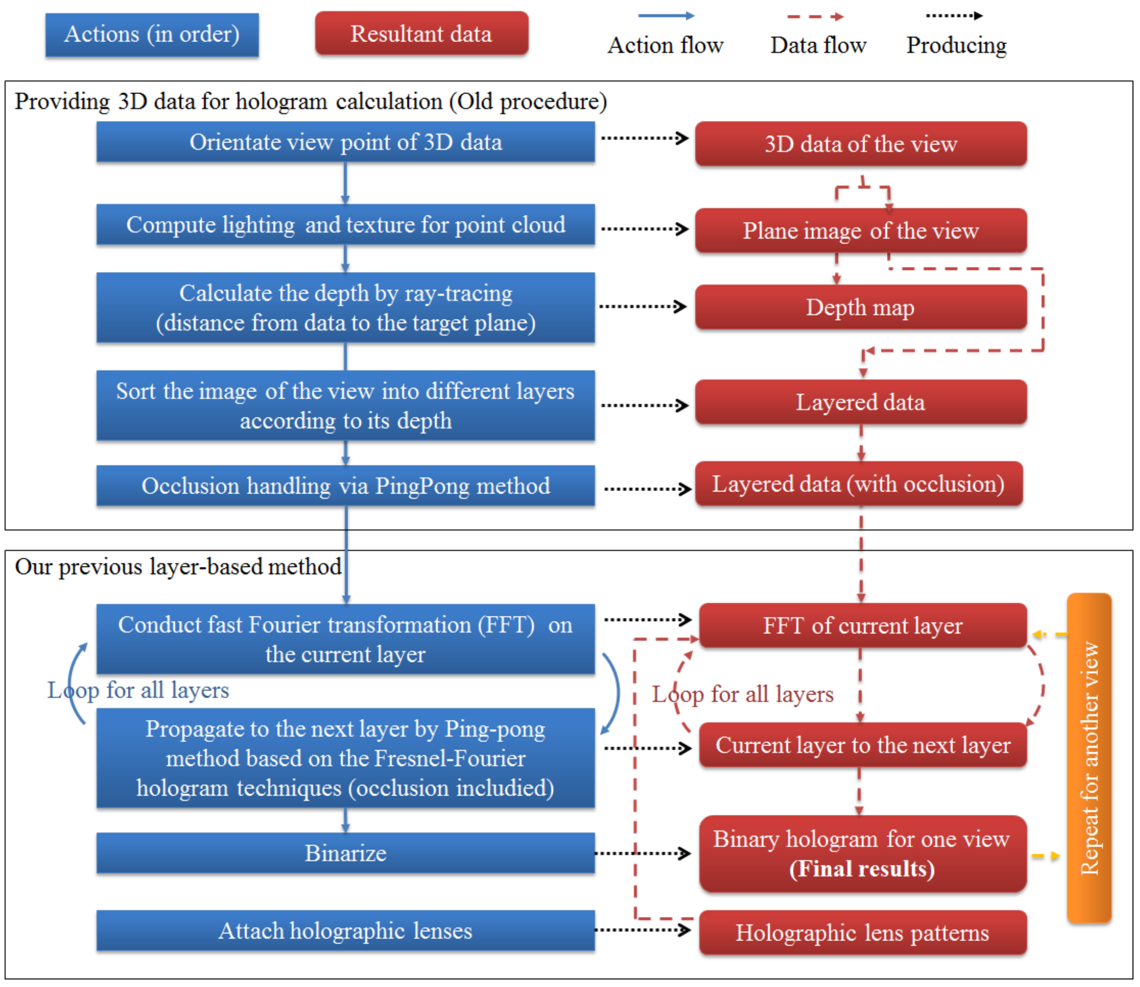


Fig 2. Illustration of a CIHD system, in which multiple subholograms corresponding to different viewing angles are integrated optically to provide a realistic 3D view of the object through angular tiling.

Most holographic rendering algorithms can be considered consisting of two main steps: transforming 3D data to an information format which is suitable for hologram calculation, and the computation of the holographic diffraction pattern.

For the multi-view layer-based holographic rendering, each view goes through the procedure shown in following Fig. 3 and it is repeated for every view.



Actions (in order)
Resultant data
Action flow
Data flow
Producing
Providing 3D data for hologram calculation (Old procedure)
Orientate view point of 3D data
3D data of the view
Compute lighting and texture for point cloud
Plane image of the view
Calculate the depth by ray-tracing (distance from data to the target plane)
Depth map
Sort the image of the view into different layers according to its depth
Layered data
Occlusion handling via PingPong method
Layered data (with occlusion)
Our previous layer-based method
Conduct fast Fourier transformation (FFT) on the current layer
FFT of current layer
Loop for all layers
Loop for all layers
Propagate to the next layer by Ping-pong method based on the Fresnel-Fourier hologram techniques (occlusion included)
Current layer to the next layer
Binarize
Binary hologram for one view (Final results)
Attach holographic lenses
Holographic lens patterns
Repeat for another view

Fig. 3 Flow chart of the procedure and the resultant data in our previous layer-based method.

Using this procedure, we were able to calculate a hologram of 900x900 pixels, for a 3D dragon of 36k points (with normals) of 30 layers, in 3.41 seconds. The speed was about 2.3×10^5 pixels per second (pps), in comparison with 1.7×10^3 pps (481.9 seconds) when using a point-based algorithm. pps is the number of pixels of a hologram calculated in a second, and is a better measure of the computational speed than the number of rendered 3D points per second, since it can be applied regardless of the rendering technique or source data format. However, the method to transform 3D data to an information format for hologram calculation was not optimized. For each view, the points were transformed and sorted to define the layers. We used point-by-point ray-tracing to calculate the shading and occlusion for the layer-based method, which was not efficient and even took more time than that of hologram calculation itself. The overall time needed is 3.4 seconds, among which 1.2 seconds (i.e. 6.75×10^5 pps) on hologram computation and 2.2 seconds (i.e. 3.68×10^5 pps) on rendering, data transformation and other processes.

2.2 Multi-view image based holograms

Instead of decomposing a 3D model itself into physical elements (points, polygons, or layers) for the 3D image hologram generation, multi-view image-based methods decompose 3D views of the model into 2D images from different viewing angles. This method is widely used in holographic stereograms[4-8], in which the hologram is composed of an array of "hogels"[9], with each hogel being a collection of diffraction gratings modulated by corresponding 2D image pixels thus steering them to their respective viewing zones. Holographic stereograms are capable of occlusion, view-dependent lighting, and support a large number of view zones providing smooth parallax and large 3D depth reconstruction, but with the disadvantage of lack of accommodation cues.

The Diffraction Specific Coherent Panoramagram (DSCP)[10,11] method reintroduced the accommodation cue into the holographic stereograms. The method used wavefront elements (wafels) that could control the direction and curvature of the wavefront at each element to controllably steer and focus the multiple 2D view images pixel by pixel into their view zones. The method requires an image of the view and its corresponding depth map for each viewpoint. The efficiency of these methods comes from the discretization of the hologram in space (hogel/wafel resolution) and frequency (viewpoints), and their ability to be parallelized and computed on GPUs [9,10.]

Multi-view image-based methods require a suitable holographic replay system (ideally with full parallax) to angularly tile their multiple 3D views with small fields of view together properly. Optical angular tiling is used to join several adjacent views of narrow viewing angle images to form a continuous and wide view image. It has been applied on holographic stereogram displays, coarse integral imaging displays[12, 13], angular viewpoints system[14, 15], and CIHD.

3. Approach

In this paper, we propose the use of a graphics rendering approach, similar to the DCSP method, integrated with the layer-based method, to significantly reduce the time spent on providing suitable information for hologram calculation. We use computer-generated imagery (CGI) color and depth renderings of the 3D model from different viewpoints to provide view-dependent lighting, occlusion/disocclusion handling, and layer slicing.

3.1 Rendering

Graphics rendering using 2D perspective projection is capable of creating images of 3D data with specific lighting and occlusion cues for a given viewing direction. OpenGL is a common standard application programming interface (API)[16] for graphics rendering. It can render a 3D polygon model into 2D perspective projection images of different viewing angles with shading and occlusion information (Fig. 4 (top)) and the corresponding depth maps (Fig. 4 (middle-top)). For each view, the 3D model is rendered and divided into N layers on the depth grid using the depth map of that direction. Here for simplicity we use equally spaced grid. For more complicated applications, uneven spacing can be used.

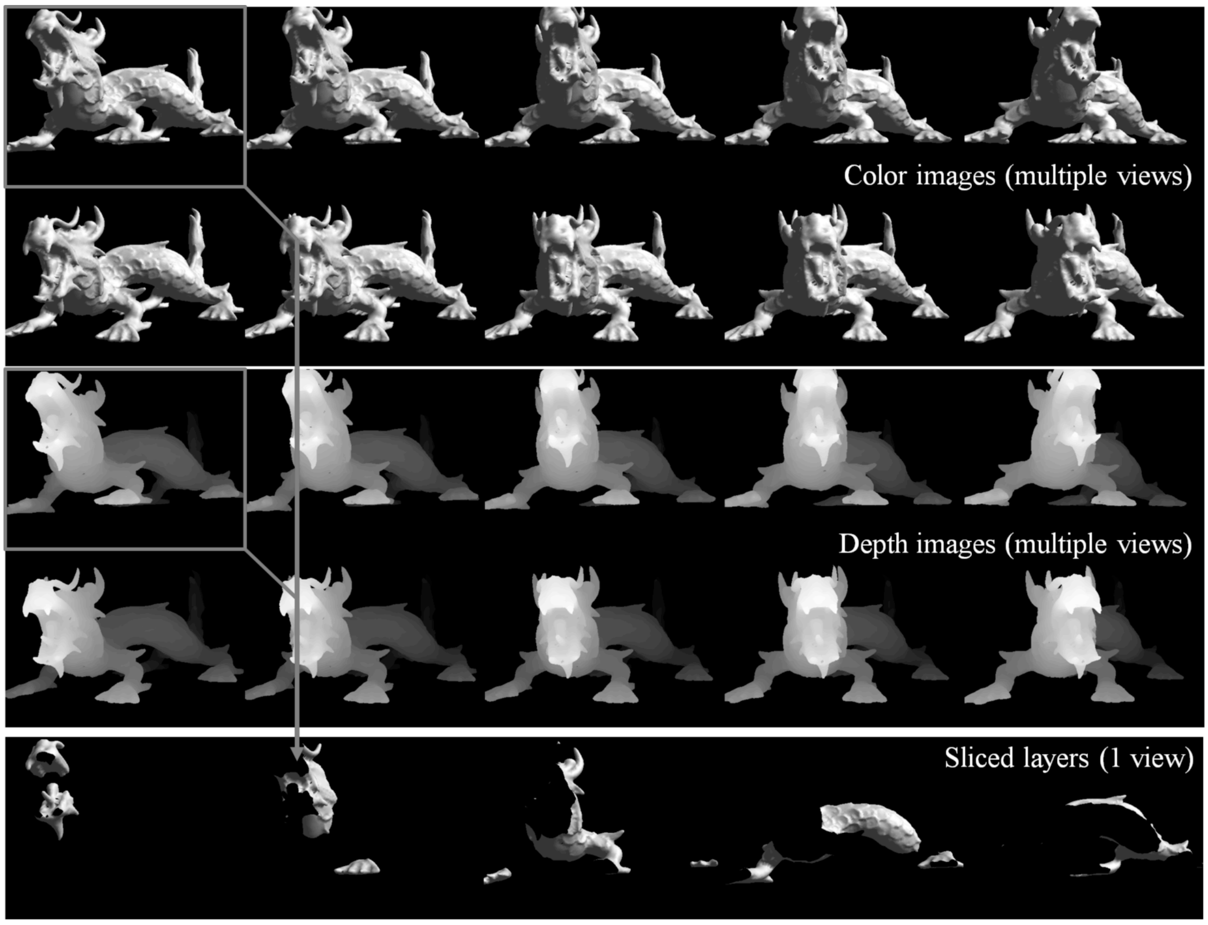


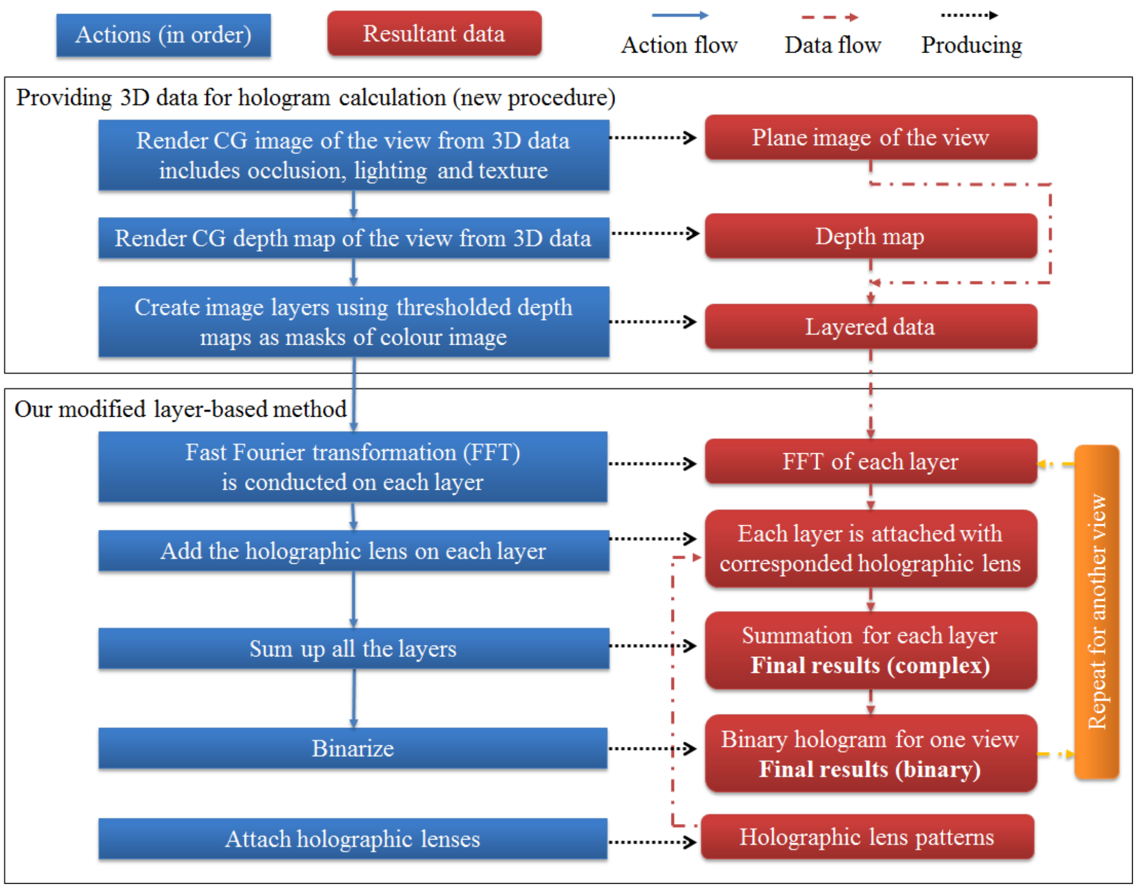
Fig. 4. 2D rendering of a 3D dragon in different viewing angles as image projections (top), their depth map (middle) and the layers for one view (bottom) used in this work for angular tiling. Each layer has a corresponding sub-hologram with an attached lens. These sub-hologram are added to form a combined hologram for one view. 2D projection includes the information of occlusion, shading and texture, and the label in each sub-hologram represents its designated viewing angle. For each view, the colour image is sliced by using thresholded depth maps as masks.

OpenGL quickly and efficiently renders each view with view transformation, depth sorting and occlusion, shading and texture, and rasterization together in one pass, giving us a view's color and depth images. We can use OpenGL's pipeline performed in parallel on a GPU, instead of the multiple custom subroutines in our previous algorithm with multiple data types performed sequentially on the CPU. For instance, previous separate routines for view transformation, per point lighting and texture computations, and layered point splatting all for a dense 3D point cloud with attached normals are all replaced with the standard OpenGL color image render of a standard polygon model. Layer slicing via ray-tracing (point-based) and occlusion handling via the Ping-Pong method (image-based wavefront propagation) can be replaced by masking the OpenGL's rendered color image with thresholded depth maps. Transparent objects can be handled as well, but it requires multiple passes since depth maps only encode the front-most surface. For simplicity, we handle opaque-objects in this work. Compared to what was used in our previous work, this rendering approach differs in the following ways: (1) using computer-generated imagery (CGI) rendering to replace lighting per point with interpolated lighting per vertex, (2) using CGI depth-sorting and occlusion to replace Ping-Pong method, (3) using CGI depth rendering to replace ray-tracing calculation, which is based on CPU computing, for producing the layer slicing, and (4) using CGI rendering support rasterization of polygon 3D models to remove the need for point splatting of dense 3D point clouds to fill-in spaces between points.

3.2 Integration of different attributes for a 3D image

To combine the techniques mentioned above and integrate different attributes for a 3D image to display, data rendering and hologram calculation are carried out in the following order: (1) define spatial coordinates and normal vector directions of a 3D model, fix a lighting direction, and select a viewing direction; (2) render the 3D object and its depth map to compose the 3D data for the chosen viewing direction; (3) perform a corresponding sorting/ slicing on the 3D data using the depth map (using thresholded depth maps to mask the colour image and produce each layer); (4) calculate the sub-hologram for each layer by combining the FFT of the sliced 2D image and a pre-computed holographic lens pattern of the corresponding focus depth. Note that the phase pattern of a holographic lens is identical to a blazed Fresnel zone plate; (5) stack up all the sub-hologram for different layers together to produce the final hologram for that viewing direction. Therefore it contains the information of all N layers, each of which has its own holographic lens; (6) repeat steps (1)-(5) for another viewing direction. This procedure is summarised in Fig. 5, and it is applied to every viewing directions included in the reconstruction. It should be mentioned that for hologram calculations, "combine" means to perform element to element matrix multiplication and "stack up" means element to element matrix addition.

Our layer-based method is implemented using MATLAB with GPUmat[17] for GPU parallel calculation, and Psychtoolbox[18] which allows MATLAB communicating with OpenGL for graphics rendering. Psychtoolbox in MATLAB is used to call OpenGL to conduct the graphic rendering, then pass the rendered image and depth map back to MATLAB, and then call GPUmat to parallel compute the final hologram using the layer-based method. Hardware and API information in use are provided in Table 1. In this paper, a hologram for a chosen 3D model of a dragon[19] (36k vertices and 68k polygonal faces) is calculated using the layer-based approach (in 30 layers).



Actions (in order)
Resultant data
Action flow
Data flow
Producing
Providing 3D data for hologram calculation (new procedure)
Render CG image of the view from 3D data includes occlusion, lighting and texture
Plane image of the view
Render CG depth map of the view from 3D data
Depth map
Create image layers using thresholded depth maps as masks of colour image
Layered data
Our modified layer-based method
Fast Fourier transformation (FFT) is conducted on each layer
FFT of each layer
Add the holographic lens on each layer
Each layer is attached with corresponded holographic lens
Sum up all the layers
Summation for each layer Final results (complex)
Binarize
Binary hologram for one view Final results (binary)
Attach holographic lenses
Holographic lens patterns
Repeat for another view

Fig. 5 Flow chart of the procedure and the produced data in our improved layer-based method

Table 1 List of the hardware and software in use.

Hardware	CPU	Intel Core i3 560	3.33 GHz
	GPU	NVIDIA GeForce	GTX 460 SE
		CUDA Cores:	288
		Graphic Clock	650 MHz
		Memory Bandwidth	108.8 GB/sec
API		MATLAB	R2011b
	Library	GPUmat	V.027
		Psychtoolbox	3.010

4 Results

4.1 Calculation speed

It took 176 ms to generate a sub-hologram of Extended Graphics Array (XGA) resolution (1,024x768 pixels) for the dragon on the computer specified in Table 1, which was equivalent to a calculation speed of 4.47 x 10^6 pps. The calculation was performed on a single commodity consumer gaming graphics card (NVIDIA GeForce GTX 460SE). The computation time includes that for rendering 3D data, sorting/slicing using depth maps, generating a hologram and binarization. Within the 176 ms, 86 ms was spent on layer-based calculation (Fourier-Fresnel hologram), 56 ms was spent on exchanging data between CPU memory and GPU memory, 16 ms was spent on conducting sorting according the depth map, 10 ms was spent on rendering, and 8 ms was spent on binarization, as shown in Table 2. Compared to the algorithms in Section 2.1, the speed of hologram calculation here (i.e. 9.14×10^6 pps) is 13.54 times faster and the speed of rendering, data transferring, etc., (i.e. 8.74×10^6 pps) is 23.75 times faster. The improvement comes from the implementation of the new procedure and optimised codes.

Table 2 Time spent on each step.

	Rendering	CPU-GPU data transferring	Sorting / Slicing	Main hologram calculation	Binarization	Total
Cost time	10 ms	56 ms	16 ms	86 ms	8 ms	176 ms
Percentage	~ 6 %	~32%	~9%	~49%	~4%	100%

*Note: The calculation time is averaged from the multiple calculations of multiple views. Different complexity of content needs different rendering time. In our case, the 3D dragon with 36k vertices (68k polygons) costs us about 10 ms. this number can be less for a simpler content, or much more for an extremely complicated scene.

4.2 Holographic results

In order to show the reconstructed image from a hologram generated using the above approach, a binary hologram of the dragon for a single view was calculated and loaded on a digital micro mirror device (DMD) which has a XGA resolution and 13.68 μm pixel pitch. The main focus of this work is the algorithm and its speed, so we only present a single view projection here to show the accommodation cue of the holographic image and its practical image quality. Its detailed optical system will be reported in a future work. The results are shown in Fig. 6, where Fig. 6(a) is the graphic rendering and Figs. 6(b)-(d) the results when the holographic image is projected on a diffusive screen. The reason of doing so is to compensate the small viewing angle of a single view. Please note that, Fig. 6(b) is the reconstructed image with a single layer while Figs. 6(c) and 6(d) are the results of the same computed hologram with 30 layers and a depth range of about 50 mm with the diffusive screen being placed at different locations.

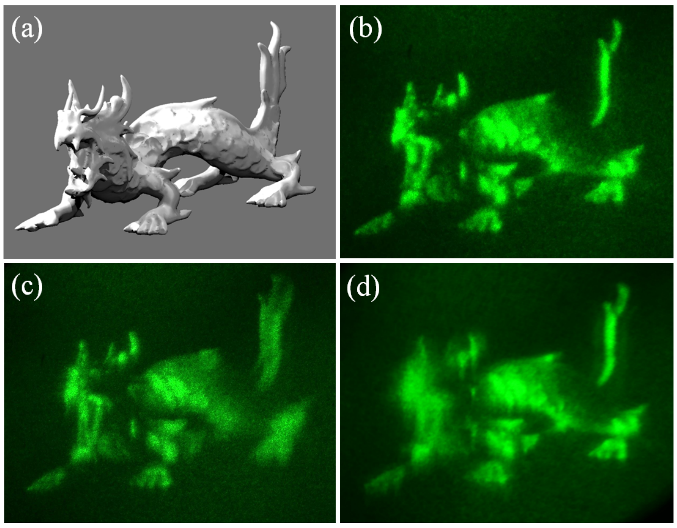


Fig. 6 Target and reconstructed holographic images for the single view on a diffusive screen, (a) target image as rendered by CG, (b) holographic reconstruction of the target image with a single plane (all data points are assigned on the same depth plane where the diffusive screen is), and (c) and (d) holographic reconstructions with depth information with the diffusive screen at different depths, around the head and the tail, respectively.

5. Discussions

5.1 Advantages

We accomplished an integrated rendering structure which previously took multiple steps: 3D transformation, view-dependent lighting, occlusion/disocclusion, and layer dissection. The CG pipeline is efficient and optimized with dedicated hardware acceleration on GPUs using standard models and interfaces, as compared to hand coded algorithms using non-standard models (3D point cloud with included normals and colors).

There are a number of significant advantages of using such a rendering approach. (1) Only the visible 2D data with occlusion, shading and texture information are kept for each view and they can be calculated easily. This is largely contributed by the back-face culling and depth sorting used in OpenGL which makes the occlusion cue rendering efficient and reduces the necessary object data to be taken into account for the specific view. (2) Depth information is associated with each 2D image only; (3) only a finite number of views are used which are sufficient to provide a smooth 3D viewing effect and directional shading updates; (4) standardized and optimized graphics pipeline (GPU acceleration) is used, which is compatible with standard polygon models, shaders and other advanced CGI techniques (ambient occlusion, normal mapping[20], morph target animation, etc.) (5) Since only colour and corresponding depth maps from different viewpoints are required, the layered hologram may be computed from live imagery such as captured using a color+depth camera (e.g. Kinect camera).

5.2 Performance Comparison

In our example, the speed of computing the hologram of the 3D dragon was equivalent to a calculation speed of 0.45 x 10^7 pps and about 19.4 times faster in overall time than that of our previous work. This is mainly due to the use of the new graphics rendering structure which significantly reduces the amount of computation needed to produce 3D data with the desired cues. As a means of comparison, we look at the calculation speed of various holographic rendering approaches implemented on GPUs.

Recently, Song, *et.al* achieved a 60 ms calculation time for a high definition (HD) resolution hologram with 4 graphics processing units (GPUs) (NVidia GTX 590)[21]. This performance is equal to a speed of 4 x 10^7 pixels per second (pps). Other research by Nakada, *et.al* calculated a hologram of 6,400 x 3,072 pixels also in 60 ms using 12 GPUs (NVidia GTX 480)[22]. This is equal to a speed of 3 x 10^8 pps. Both studies use approximately 3,000 points to represent their 3D models, which may not be sufficient for a large and complicated object. Obviously the number of elements used in the decomposition has a significant effect on the overall computation time. For example, if the same point density (the number of points divided by the number of the hologram pixels) as that in Ref. [21] was used in Ref. [22], the equivalent computation speed of the latter would decrease by one order to 3 x 10^7 pps.

Apart from these two examples, a speed of 2 x 10^7 pps was achieved also using a point-based method with 3 GPUs (NVidia GTX 285) and the help of the split look-up table (LUT) method[23]. In comparison, a polygon-based method by Matsushima had achieved a speed of 3 x 10^5 pps without using GPU[24]. Another polygon-based work[25] by Hosoyachi calculated a 3D model of 1,000 polygons in 3 seconds using a GPU (NVidia GTX 480), which is equal to a speed of 1 x 10^6 pps. In addition, Tsang, *et.al* achieved 1.7 x 10^7 pps in hologram calculation using the wave-front record plane (WRP) method with a GPU (NVidia GTX 580) and their specialized algorithm[26]. However, WRP is not a general method, and can only be used for objects with shallow depth.

Overall, the calculation speed for holograms generation at present is in the order of 10^5~10^7 pps, and can be increased to the order of 10^8 pps by using multiple GPUs for the limited number of basic elements. Introduction of shading and occlusion will further complicate the situation. Therefore, the practical speed to calculate a complicated 3D objects with shading and occlusion may well be much less than 10^7 pps even if multiple GPUs are used.

We have computed the hologram of a complicated 3D dragon with occlusion and shading effects at a calculation speed equivalent to 0.45 x 10^7 pps. Using a single GPU of consumer grade, we achieved the speeds comparable to those of multiple GPUs solutions. We believe that with a high-performance GPGPU/GPU or a combination of several mid-range consumer GPUs, the calculation speed reported here can easily be further improved by at least another order.

6. Conclusions

The approach proposed above combines a new graphic rendering method for 3D image data production which is integrated into a layer-based method of hologram calculation for angular tiling. It was implemented on a normal computer with a consumer graphic card to show its fast computation speed in generating a hologram with necessary depth cues.

The demonstrated calculation speed can be further improved by using a high-end GPU or even multiple devices. Furthermore, the GPUmat used in this work doesn't support parameter optimization. Therefore, programing in C/C++ with CUDA can further speed up the algorithm. In addition, the use of the sparse FFT[27, 28] may also help. This algorithm was implemented on GPU recently[29] and its performance was shown to be more efficient than of the cuFFT library[30] in CUDA.

Acknowledgments

This research was performed through a joint collaboration project between the University of Cambridge and Disney Research through the CAPE consortium. JSC and DPC would like to thank the UK Engineering and Physical Sciences Research Council (EPSRC) for the support through the Platform Grant in Liquid Crystal Photonics. JSC also want to thank Taiwan Education Ministry for their funding to his study in Cambridge.

References

1. J.-S. Chen, Q. Smithwick, and D. Chu, "Implementation of shading effect for reconstruction of smooth layer-based 3D holographic images", Proc. SPIE 86480R–86480R-9 (2013) [doi: 10.1117/12.2004704].
2. Q. Smithwick, J.-S. Chen, and D. Chu, "A Coarse Integral Holographic Display," SID Symposium Digest of Technical Papers 44, 310–313 (2013) [doi:10.1002/j. 2168-0159.2013.tb06208.x]
3. D. Abookasis and J. Rosen, 'Three types of computer-generated hologram synthesized from multiple angular viewpoints of a three-dimensional scene', Appl. Opt., vol. 45, no. 25, pp. 6533–6538, Sep. 2006. [doi:10.1364/AO.45.006533]
4. S. A. Benton and V. M. Bove Jr, "Holographic Imaging", John Wiley & Sons, Hoboken, New Jersey, (2008).
5. M. Lucente, "Diffraction-Specific Fringe Computation for Electro-Holography," Doctoral Thesis Dissertation, Massachusetts Institute of Technology Department of Electrical Engineering and Computer Science (1994).
6. W. Plesniak, M. Halle, J. Bove, J. Barabas, and R. Pappu, "Reconfigurable image projection holograms", Opt. Eng 45, 115801–115801 (2006) [doi:10.1117/1.2390678]
7. H. Kang, T. Yamaguchi, and H. Yoshikawa, "Accurate phase-added stereogram to improve the coherent stereogram", Appl. Opt. 47, D44–D54 (2008) [doi:10.1364/AO.47.000D44]
8. Q. Y. J. Smithwick, J. Barabas, D. E. Smalley, and V. M. Bove Jr, "Real-time shader rendering of holographic stereograms", Proc. SPIE 7233, p. 723302, (2009) [doi:10.1117/12.808999]
9. M. Lucente, 'Diffraction-Specific Fringe Computation for Electro-Holography, Chapter 4", Cambridge MA, USA: Massachusetts Institute of Technology Department of Electrical Engineering and Computer Science, 1994., pp. 45-75.
10. Q. Y. J. Smithwick, J. Barabas, D. E. Smalley, and V. M. Bove, Jr., "Interactive holographic stereograms with accommodation cues", Proc. SPIE 7233, 723302-1–723302-12 (2009) [doi: 10.1117/12.840526]

11. J. Barabas, S. Jolly, D. E. Smalley, and V. M. Bove Jr, "Diffraction Specific Coherent Panoramagrams of Real Scenes", in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series (2011), Vol. 7957, p. 1. [doi:10.1117/12.873865]

12. H. Kakeya, "Formulation of coarse integral imaging and its applications", Proc. of SPIE 6803, 680317–1-680317–10 (2008) [doi:10.1117/12.766338]

13. H. Kakeya, "Improving image quality of coarse integral volumetric display," Proc. SPIE 7237, 723726-1–723726-9 (2009) [doi:10.1117/12.805469]

14. D. Abookasis and J. Rosen, "Computer-generated holograms of three-dimensional objects synthesized from their multiple angular viewpoints", J. Opt. Soc. Am. A 20, 1537–1545 (2003) [doi:10.1364/JOSAA.20.001537]

15. D. Abookasis and J. Rosen, "Three types of computer-generated hologram synthesized from multiple angular viewpoints of a three-dimensional scene", Appl. Opt. 45, 6533–6538 (2006) [doi:10.1364/AO.45.006533]

16. "OpenGL - The Industry Standard for High Performance Graphics," http://www.opengl.org/.

17. "GPUmat: GPU toolbox for MATLAB," http://gp-you.org/.

18. "Psychtoolbox Wiki: Psychtoolbox-3," http://psychtoolbox.org/HomePage.

19. "The Stanford 3D Scanning Repository," http://www-graphics.stanford.edu/data/3Dscanrep/.

20. Dave Shreiner, Graham Sellers, John Kessenich, and Bill Licea-Kane, 'Procedural Texutring', in OpenGL-Programming Guid 8th edition- The official guide to learning OpenGL, version 4.3, 8th ed., Addison Wesley, 2013, pp. 411–484.

21. J. Song, J. Park, and J.-I. Park, "Fast calculation of computer-generated holography using multi-graphic processing units", 2012 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), pp. 1 –5, (2012) [doi:10.1109/BMSB.2012.6264286].

22. N. Takada, T. Shimobaba, H. Nakayama, A. Shiraki, N. Okada, M. Oikawa, N. Masuda, and T. Ito, "Fast high-resolution computer-generated hologram computation using multiple graphics processing unit cluster system", Appl. Opt. 51, 7303–7307 (2012) [doi:10.1364/AO.51.007303]

23. 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). [doi: 10.1364/OE.17.018543]

24. K. Matsushima and S. Nakahara, "Extremely high-definition full-parallax computer-generated hologram created by the polygon-based method", Applied optics 48, 54–63 (2009) [doi: 10.1364/AO.48.000H54]

25. K. Hosoyachi and Y. Sakamoto, "Acceleration of calculation method for CGH with spherical basic object light by using graphic processing units," Proc. SPIE 82810U–82810U (2012) [doi: 10.1117/12.906629]

26. P. Tsang, W.-K. Cheung, T.-C. Poon, and C. Zhou, "Holographic video at 40 frames per second for 4-million object points", Opt. Express 19, 15205–15211 (2011) [doi:10.1364/OE.19.015205]

27. H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Simple and practical algorithm for sparse Fourier transform", in Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA ’12 (SIAM, 2012), pp. 1183–1194.

28. H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Nearly Optimal Sparse Fourier Transform", arXiv:1201.2501 (2012) [doi:10.1145/2213977.2214029]

29. J. Hu, Z. Wang, Q. Qiu, W. Xiao, and D. J. Lilja, "Sparse Fast Fourier Transform on GPUs and Multi-core CPUs", in 2012 IEEE 24th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD) (2012), pp. 83 –91.[doi:10.1109/SBAC-PAD.2012.34]

30. "CUFFT :: CUDA Toolkit Documentation," http://docs.nvidia.com/cuda/cufft/index.htm

Authors' Biographies

Jhen-Si Chen is a Ph.D. candidate at Engineering Department, University of Cambridge. He received his Bachelor of Science degree in Physics from National Taiwan University in 2009, and joined Photonics & Sensors Group at the University of Cambridge in 2010. His research focuses on algorithm and optics system of holographic 3D displays.

Daping Chu is the Head of the Photonics & Sensors Group and Chairman of the Centre for Advanced Photonics and Electronics at the University of Cambridge. His current interests include future display technologies including 2D/3D holography and full color high brightness trans-reflective displays, GHz/THz tunable dielectrics, energy saving and radiation control for the built environment, metal oxide materials and transparent electronics, and printable and flexible electronics and inkjet fabrication.

Quinn Smithwick is a Research Scientist at Disney Research in Glendale, California. His main research area is novel display technologies with particular emphasis on autostereoscopic displays and spatial augmented reality environments. He also represents and directs Disney Research's fundamental display research with the Centre for Advanced Photonics and Electronics at the University of Cambridge (UK).

Caption List

Fig. 1 Illustration of slicing and its gap issue.

Fig 2 Illustration of a CIHD system.

Fig. 3 Flow chart of the procedure and the produced data in our previous layer-based method.

Fig. 4 2D rendering of a 3D dragon in different viewing angles.

Fig. 5 Flow chart of the procedure and the produced data in our improved layer-based method.

Fig. 6 Reconstructed holographic image (single view on a scatter).

Table 1 List of the hardware and software in use.

Table 2 Time spent on each step.