Three-dimensional range data compression using computer graphics rendering pipeline

Song Zhang

Department of Mechanical Engineering, Iowa State University, Ames, Iowa 50011, USA
(song@iastate.edu)

Received 1 February 2012; revised 21 April 2012; accepted 7 May 2012;
posted 7 May 2012 (Doc. ID 162388); published 13 June 2012

This paper presents the idea of naturally encoding three-dimensional (3D) range data into regular two-dimensional (2D) images utilizing computer graphics rendering pipeline. The computer graphics pipeline provides a means to sample 3D geometry data into regular 2D images, and also to retrieve the depth information for each sampled pixel. The depth information for each pixel is further encoded into red, green, and blue color channels of regular 2D images. The 2D images can further be compressed with existing 2D image compression techniques. By this novel means, 3D geometry data obtained by 3D range scanners can be instantaneously compressed into 2D images, providing a novel way of storing 3D range data into its 2D counterparts. We will present experimental results to verify the performance of this proposed technique. © 2012 Optical Society of America

OCIS codes: 120.2650, 100.5070, 100.6890.

1. Introduction

With the rapid development of 3D range scanning, especially those 3D video scanning techniques, it becomes increasingly easier to obtain and access 3D contents. However, the size of 3D range data is drastically larger than that of 2D counterparts. Therefore, storing and transporting 3D range data have become important issues to be dealt with [1].

Conventional formats (e.g., STL, OBJ, PLY) to store 3D range data are effective in terms of 3D surface representation. However, they usually store (x, y, z) coordinates for each vertex, the connectivity information between vertices, and sometimes the surface normal information, and thus they utilize a lot of storage space. Over the years, various methods [2–4] have been developed to compress 3D range scanned data. There are generic to arbitrary 3D mesh data, and their compression ratios are quite high. However, these often involve very time-consuming encoding processes, and thus cannot be used for real-time 3D video applications.

Another method is to represent 3D range video data as a phase depth map [5], which has been demonstrated successful for 3D live video communication. Furthermore, the floating point phase map could be represented with a regular 24-bit image by packing the most significant 24 bits into red, green, and blue (RGB) channels of the regular image, and discarding the least significant bits. The 24-bit RGB images can then be unpacked to recover 3D geometry with a little quality loss. Though it could be successful, this technique is limited to utilize lossless 2D image format. This is because the most significant bits contains the power bits, any change will result in significant error for the unpacked floating point number.

Inspired by our research on 3D shape measurement with fringe projection techniques, we have recently developed a 3D range data compression technique using Holoimage [6] to convert 3D data into regular 2D images [7], and later extended to 3D range video compression [1]. Specifically, this technique consists in building a virtual fringe projection system called Holoimaging using advanced computer graphics tools to image virtual 3D objects as 2D RGB images, and to further compress 2D images

1559-128X/12/184058-07$15.00/0

with standard 2D compression techniques (e.g., JPG, PNG). Since 3D geometry information is encoded into cosine functions, the compression ratio was found very high and the recovered 3D geometry was of great quality. However, because one 8-bit channel spatially encodes 2π phase jumps, the Holoimage technique is limited to use a finite number of fringe stripes, resulting in relatively low-resolution 2D images to represent 3D geometries, which will be problematic if the original 3D range data is of higher resolution.

This paper presents a new technique to overcome the limitations of the Holoimage compression method by eliminating its spatial encoding requirement. Instead, this technique directly encodes depth (z information) into RGB images. This technique naturally encodes 3D range data into regular 2D images utilizing the advanced computer graphics pipeline (e.g., OpenGL). To render 3D geometry into 2D images on a computer screen, the computer graphics rendering pipeline (CGRP) provides a means to sample 3D geometry data into 2D images. Moreover, the advanced computer graphics tools also provide a way to obtain the depth (z) for each sampled pixel. The depth information for each pixel is further encoded into RGB color channels of a regular 2D image. The 2D images can then be compressed with existing 2D image compression techniques. Similar to the Holoimage technique, each channel of the RGB image is represented as a cosine function, and thus the encoded 2D image can be highly compressed without a significant loss of quality. Comparing with our previously proposed Holoimage compression technique, this technique directly encodes depth z into 2D images without spatial encoding, and thus it can be extended to sample arbitrary size 3D objects into arbitrary resolution 2D images. Moreover, because 3D objects can be rendered onto computer screens at a high speed, this novel encoding technique permits compressing 3D data into 2D images in real time, providing an effective and efficient means to store 3D range data into their 2D counterparts.

Section 2 explains the principle of encoding and decoding. Section 3 shows experimental results. Section 4 discusses the merits and limitations of the proposed technique, and Section 5 summarizes this paper.

2. Principle

A. Phase-Shifting Technique for 3D Shape Measurement

Phase-shifting techniques have been extensively adopted in optical metrology due to their numerous merits over other techniques, such as their capability to achieve pixel-by-pixel spatial resolution. Over the years, numerous phase-shifting algorithms have been developed, as summarized in this book chapter [8]. Although a multiple-step phase-shifting algorithm is not very sensitive to linear phase shift errors [9], a three-step phase-shifting algorithm is usually desirable for high-speed applications since it requires the minimum number of fringe images to obtain high-quality phase. For a three-step phase-shifting algorithm with equal phase shifts, three fringe images can be described as

$$I_1(x,y) = I'(x,y) + I''(x,y)\cos(\phi - 2\pi/3), \tag{1}$$

$$I_2(x,y) = I'(x,y) + I''(x,y)\cos(\phi), \tag{2}$$

$$I_3(x,y) = I'(x,y) + I''(x,y)\cos(\phi + 2\pi/3), \tag{3}$$

where $I'(x,y)$ is the average intensity, $I''(x,y)$ the intensity modulation, and $\phi(x,y)$ the phase to be found. From these three equations, we can calculate the phase,

$$\phi(x,y) = \tan^{-1}\left[\frac{\sqrt{3}(I_1 - I_3)}{2I_2 - I_1 - I_3}\right]. \tag{4}$$

This equation provides the wrapped phase ranging from 0 to 2π with 2π discontinuities. Conventionally, these 2π phase jumps can be removed by adopting a spatial phase-unwrapping algorithm, such as one of the algorithms discussed in [10]. If the system is properly calibrated [11], (X,Y,Z) coordinates can be obtained from the unwrapped phase $\Phi(x,y)$ pixel-by-pixel:

$$X = f_1(x,y;\Phi), \tag{5}$$

$$Y = f_2(x,y;\Phi), \tag{6}$$

$$Z = f_3(x,y;\Phi). \tag{7}$$

However, since a spatial phase-unwrapping algorithm is adopted, such a system can neither measure large step-height changes that cause phase changes larger than π, nor does it handle discontinuous surfaces.

B. Holoimage Encoding

It is important to notice that the X and Y in Eqs. (5) and (6) are usually not uniformly distributed for 3D range data coming from a 3D shape measurement system, and thus it is not sufficient to solely use depth Z to represent recovered 3D shapes. On the other hand, to compress 3D range data, it is desirable to ensure that X and Y are uniformly distributed.

To accomplish this task, we have previously developed a virtual fringe projection system called Holoimage [6]. In such a system, both the projector and the camera use "telecentric lenses" so that they create parallel projections instead of perspective projection, making the spatial sampling uniform; in other words, X and Y are uniformly distributed.

Since the Holoimage system is virtually built, the "ambient light" can be controlled, and the surface reflectivity can be perfectly uniform. Therefore, only two fringe images are required to recover the phase, making it possible to use the third image to assist phase unwrapping. Modifying from Eqs. (1–3), three encoded patterns can be described as

$$I_r(x,y) = 127.5 + 127.5\,\sin(2\pi x/P), \tag{8}$$

$$I_g(x,y) = 127.5 + 127.5\,\cos(2\pi x/P), \tag{9}$$

$$I_b(x,y) = S \times \mathrm{Fl}(x/P) + 0.5 + 0.5(S-2) \times \cos[2\pi \cdot \mathrm{Mod}(x,P)/P_1]. \tag{10}$$

Here, P is the fringe pitch, the number of pixels per fringe stripe, $P_1 = P/(K+0.5)$ is the local fringe pitch and K is an integer number, S is the stair height in gray scale value, $\mathrm{Mod}(a,b)$ is the modulus operator to get a over b, and $\mathrm{Fl}(x)$ is to get the integer number of x by removing the decimals. It should be noted that Eq. (10) varies sinusoidally to enable lossy compression [1]. Both the aforementioned method and the previously proposed technique [7] utilized a stair image to ensure that the stair changes perfectly align with the 2π discontinuities.

From these three images and the setup parameters of the Holoimaging system, (X,Y,Z) coordinates can be recovered as

$$X^n = j/W, \tag{11}$$

$$Y^n = i/W, \tag{12}$$

$$Z^n = \frac{P\Phi(x,y) - 2\pi i\,\cos(\theta)}{2\pi W\,\sin\theta}, \tag{13}$$

where

$$\Phi(x,y) = 2\pi \times \mathrm{Fl}[(I_b - 0.5S)/S] + \tan^{-1}\left[\frac{I_r - 127.5}{I_g - 127.5}\right]. \tag{14}$$

θ is the angle between the projector and the camera, (i,j) are image pixel indices, W is the image width, and (X^n, Y^n, Z^n) are normalized coordinates that can be converted back to their original coordinates by applying the predefined scaling factor and translation vector. It is important to note that in this case, the Holoimaging system was set up so that both the projector and the camera use orthographic projections, and the fringe stripes are vertical along y direction (i.e., vary horizontally along x direction).

C. Direct Depth Z Encoding

However, because of quantization error, the stair height cannot be only 1 gray scale value. Further more, lossy compression techniques require much larger stair height to ensure that coded images are less vulnerable to noise. In practice, the stair height is usually larger than 10. *This means that there are only approximately 25 stairs to use.* Since the fringe patterns are spatially (along x or y direction) sampled by the virtual fringe projection system, the Holoimage technique can neither encode dense fringe images nor reach high-resolution representation.

In contrast, if the encoding is performed along depth Z direction, instead of spatially along X or Y, the spatial resolution limitation will be eliminated. In other words, we directly encode depth Z such that

$$I_r(i,j) = 127.5 + 127.5\,\sin(2\pi Z/P), \tag{15}$$

$$I_g(i,j) = 127.5 + 127.5\,\cos(2\pi Z/P), \tag{16}$$

$$I_b(i,j) = S \times \mathrm{Fl}(Z/P) + 0.5S + 0.5(S-2) \times \cos\left[2\pi \times \frac{\mathrm{Mod}(Z,P)}{P_1}\right]. \tag{17}$$

Equations (15–17) provide the depth Z uniquely for each point:

$$Z = P\left[\mathrm{Fl}\left(\frac{I_b - 0.5S}{S}\right) + \frac{1}{2\pi} \times \tan^{-1}\left(\frac{I_r - 127.5}{I_g - 127.5}\right)\right]. \tag{18}$$

If the X and Y coordinates are sampled uniformly so that they are proportional to their image indices (i,j), which are scaled by the pixel size, i.e.,

$$X = j \times c, \tag{19}$$

$$Y = i \times c. \tag{20}$$

Here, c is constant that can be specified by the user. By this means, 3D shape can also be encoded as a single image while eliminating some limitations of the Holoimage system.

D. Computer Graphics Rendering Pipeline for Uniform Sampling

The depth Z encoding technique introduced in Subsection 2.C. requires directly sampling 3D range data uniformly along X and Y directions. Unfortunately, this is usually nontrivial considering the irregular shape of the object, and the irregular (x,y) coordinates coming from a range scanner. Utilizing a conventional interpolation technique could be extremely time-consuming. This paper addresses this challenge by taking advantage of the CGRP.

Figure 1 illustrates the typical rendering pipeline that starts with geometry processing. The geometry processing step takes care of the back-face culling that is to remove those vertices that face away from

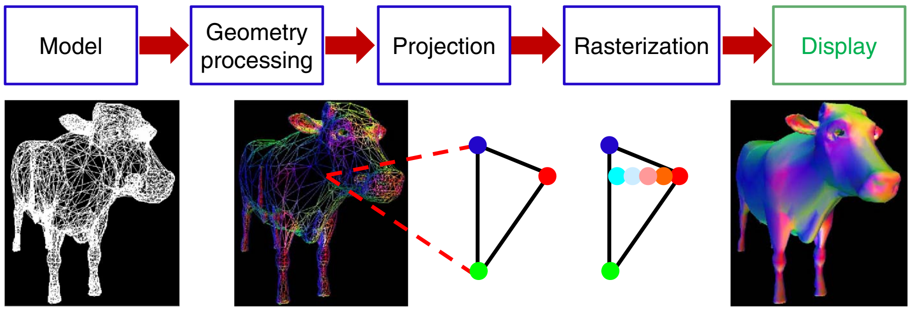

Fig. 1. (Color online) Computer graphics rendering pipeline.

the viewer. The frontal vertices are then projected 3D coordinates onto a 2D image plane by applying the projection matrix. The rasterization stage is to fill in the polygons (if required) through 2D interpolation, and to remove those points that are hidden by another through depth buffering. Finally, the 2D images are displayed on a 2D computer screen pixel-by-pixel.

Since most computer screens contain squared pixels, this CGRP usually results in squared 2D points on the computer screen. If the projection is orthographical, then the screen coordinates (i,j) are naturally proportional to the original (X,Y) coordinates in the object space. This means that the CGRP provides a means to sample 3D shape uniformly along x and y directions. Since the advanced computer graphics tools can do high-resolution, real-time 3D rendering, it thus also provides a very efficient way for this procedure. Therefore, if we can obtain depth Z pixel-by-pixel on the computer screen, we can adopt the direct depth encoding technique introduced in Subsection 2.C. for 3D compression. Fortunately, the advanced computer graphics rending techniques provide a way called *render to texture*. By rendering the scene to texture instead of computer screen, the depth Z can be recovered pixel-by-pixel through unprojection. This paper uses this technique for 3D range data encoding, and thus for 3D shape compression.

3. Experiments

We experimented with an ideal unit sphere generated by a computer to verify the performance of the proposed technique. In this experiment, the 3D sphere was rendered to a 512×512 resolution texture. For each pixel, the depth information was recorded and encoded into color information following Eqs. (15–17). The 2D encoded image is shown in Fig. 2(a), and Figs. 2(b)–2(d) show its three color channels. From red and green channels, the wrapped phase can be obtained, as shown in Fig. 2(e). Figure 2(f) shows the stair image that can be quantified into $k(x,y)$ for phase unwrapping. The unwrapped phase map is shown in Fig. 2(g). It is important to note that due to sampling and/or noise for lossy image formats, the perfectly aligned phase jumps and the stair changes may shift 1 or 2 pixels. To solve this problem, the unwrapped phase was processed by a 13×13 median filter to locate those misaligned pixels, and then properly adjust the $k(x,y)$ for those pixels. After this step, the depth map can be properly obtained.

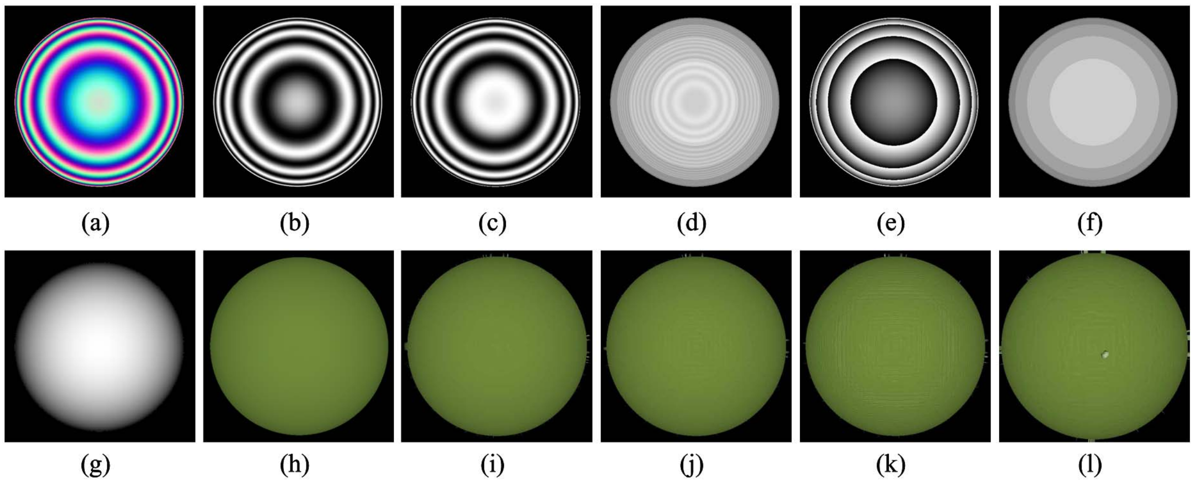

Fig. 2. (Color online) Experimental results of an ideal sphere. (a) Encoded 2D color image, (b)–(d) three color channels, (e) wrapped phase from red and green channels, (f) stair image, (g) unwrapped phase, (h) 3D recovered result, (i)–(l) 3D results from JPG images with quality levels of 12, 10, 8, and 6, respectively.

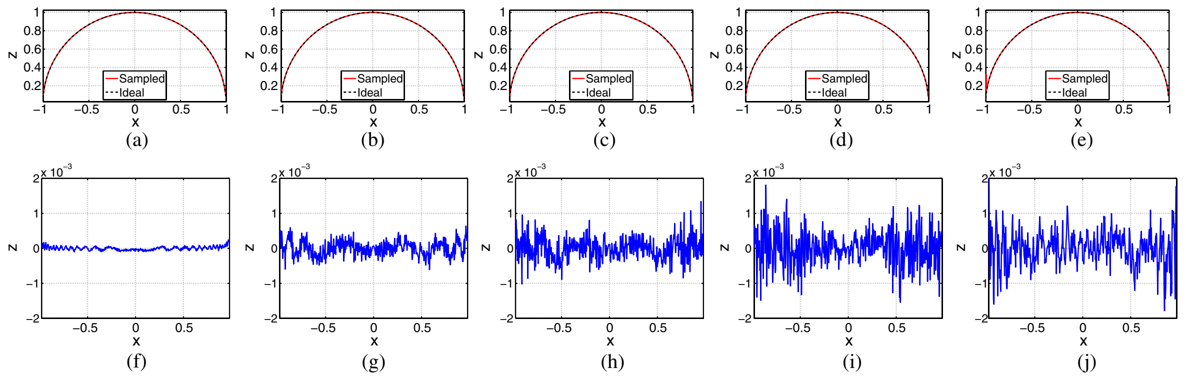
Fig. 3. (Color online) Comparison between the ideal sphere and the recovered 3D results from different quality 2D images. (a)–(e) Cross sections of the ideal shape and the recovered 3D results shown in Figs. 2(h)–2(l), (f)–(j) errors for the cross sections shown above. The rms errors for (f), (g), (h), (i), and (j) are 0.006%, 0.022%, 0.033%, 0.052%, and 0.051%, respectively.

Since the pixel size is precisely defined for this pipeline, the 3D shape can be recovered, as shown in Fig. 2(h). Furthermore, we stored the 2D image in lossy image formats, such as the JPG format with different qualities, and recovered the 3D shapes from those lossy image formats. Figures 2(i)–2(l) show the results when the images were stored at quality levels of 12, 10, 8, and 6, respectively. The quality level was defined by Adobe Photoshop CS3 with 12 being the best quality JPG format. This experiment shows that if the encoded 2D image is stored as a lower quality JPG file (with more compression), the recovered 3D sphere from the lower quality JPG file is lower. Also, one may notice that the sphere is no longer smooth for those 3D results recovered from lower quality 2D JPG files, and some artifacts appear on the sphere. Nevertheless, all 3D shapes are properly recovered even if the image quality is as low as level 6 (the size is approximately 57 KB for a 512×512 image).

To verify the accuracy of the recovered 3D sphere comparing with the ideal one, cross sections of the recovered 3D data from the results shown in Fig. 2 were presented in Fig. 3. Figure 3(a) shows the cross section of overlaying the recovered 3D shape shown in Fig. 2(h) with the ideal one. It can be seen that they are almost identical. Figure 3(f) shows the difference between these two cross sections. The 3D shape is recovered from virtually encoding the depth into color images, but this figure shows some noise. The random noise was introduced during the quantization process when the floating point data was converted to 8-bit fringe images. The error was found to be approximately root-mean-square (rms) value of

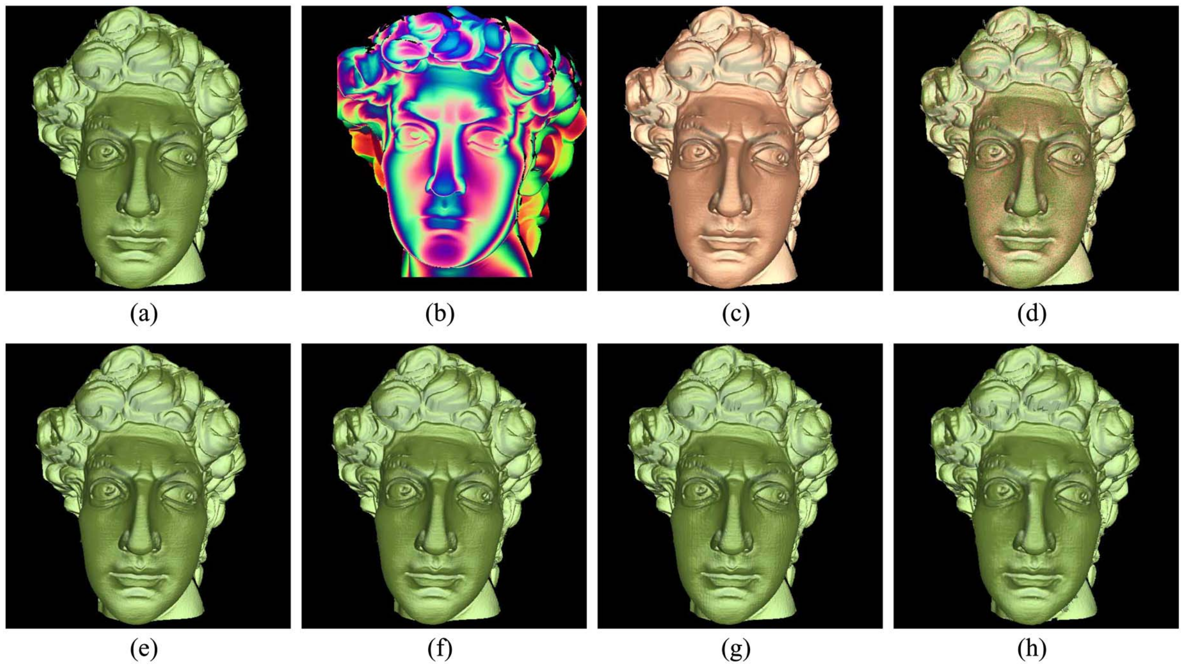
Fig. 4. (Color online) Experimental results of a more complex 3D statue. (a) Original 3D data, (b) encoded 2D image for the 3D data, (c) recovered 3D shape from the lossless PNG image, (d) overlapping between original and recovered 3D data, (e)–(h) 3D results from lossy JPG images with quality levels 12, 10, 8, and 6, respectively.

Table 1. Compression Ratios for Different Encoded Image Formats Versus 3D Mesh File Formats

	BMP	PNG	JPG12	JPG10	JPG8	JPEG6
OBJ	29.1 : 1	87.9 : 1	103.5 : 1	192.2 : 1	273.4 : 1	364.3 : 1
PLY	10.7 : 1	32.3 : 1	38.0 : 1	70.7 : 1	100.5 : 1	134.0 : 1
STL	14.8 : 1	44.7 : 1	52.6 : 1	97.8 : 1	139.0 : 1	185.3 : 1

0.006%. This indeed shows that the encoding technique can represent the original geometry with high accuracy if they are stored in a lossless image format, such as PNG. Moreover, the larger error on both ends was caused by the sampling limitation: it is impossible to sample the depth data when the sampling direction is parallel to the surface tangent plane. For this example, the largest angle between the sphere surface normal direction and the sampling direction is approximately 84°. As a comparison, Figs. 3(b)–3(e) and 3(g)–3(j) show the results if 3D shapes are recovered from lossy JPG images. It can be seen that the rms error is larger if the 2D JPG image is compressed more. It is important to note that the rms error is still pretty small (less than 0.051%) even if the 2D images are stored as a very low-quality JPG file.

Moreover, a more complex 3D geometric shape was compressed with the proposed technique, as shown in Fig. 4. Figure 4(a) shows the original 3D statue, and Fig. 4(b) shows the encoded 2D image, from which the 3D shape can be recovered, as shown in Fig. 4(c), when the image is stored as the lossless PNG format. Overlaying the original 3D data with the recovered one [Fig. 4(d)] shows no obvious differences between them. This once again verifies that the proposed encoding technique can represent the original 3D data with a high quality. We then stored the 2D image with the lossy JPG format under different quality levels. The recovered 3D results are shown in Figs. 4(e)–4(h). These again showed that even with low-quality JPG images, the overall 3D shape can still be properly recovered, even though some details were lost.

As aforementioned, converting 3D data to 2D images can significantly reduce storage size. We use the statue example to illustrate the compression ratios of the encoded image formats in comparison with three popular 3D mesh formats: OBJ, PLY, and STL. OBJ and PLY formats are widely used in computer graphics, while the STL format is extensively used in the manufacturing industry. Table 1 summarizes the data. This table shows that even converting the 3D data to the lossless BMP format, the lowest compression ratio is still above 10 : 1. If the image is stored as the highest quality JPG format, a 53 : 1 compression ratio is achieved in comparison with the STL format. If the lower quality 3D geometry is sufficient, the compression ratio can go over 360 : 1 comparing with the OBJ format. This experiment indeed shows that the huge storage space can be saved by storing the 3D geometry into the 2D image with the proposed compression technique.

Finally, the multiresolution representation was tested for this proposed technique. Unlike the previously proposed techniques, this technique can represent any resolution 3D shape properly. This functionality was realized by changing the field view of computer graphics pipeline, precisely moving the view patch by patch, and stitching the resultant images into a complete image. Figures 5(a)–5(d)

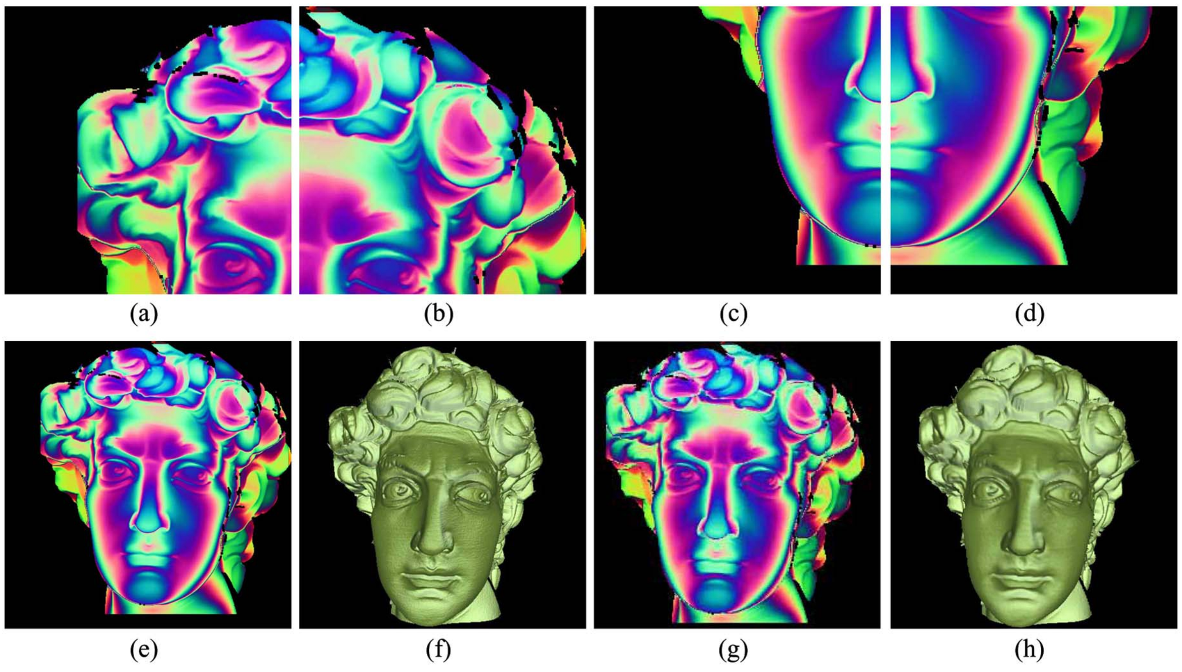

Fig. 5. (Color online) Multiresolution experiments. (a)–(d) Four images with each representing 1/4 of the 1 k × 1 k full-resolution image, (e) combined full-resolution image, (f) recovered 3D result from the high-resolution image shown in (e), (g) low-resolution image (256 × 256), (h) recovered 3D shape from the low-resolution image shown in (g).

show four subimages with each being 512×512. These four subimages were combined into a $1\ \mathrm{k} \times 1\ \mathrm{k}$ resolution single image, as shown in Fig. 5(e), from which the 3D shape can be recovered. Figure 5(f) shows the recovered 3D shape from the high-resolution image. It clearly shows a lot more details than that shown in lower resolution images, such as those shown in Fig. 4. Furthermore, the low-resolution image can be easily generated. For example, Fig. 5(g) shows the 256×256 resolution encoded image, and Fig. 5(h) shows the recovered 3D result. It can be seen that the 3D shape lost a lot of details, but the overall 3D shape was still properly recovered.

4. Discussions

The proposed compression technique has the following merits over the previously proposed Holoimage technique:

- *Multiresolution capability.* This proposed technique allows for representing 3D shapes with 2D images of arbitrary size. Since it directly encodes depth into RGB color channels of the image, the limitation of Holoimage technique does not present in this new technique.
- *Easy encoding and decoding.* This technique directly utilizes the CGRP without additional configurations, and thus it could be potentially the most efficient means to instantaneously perform encoding and decoding.
- *Flexible depth range encoding.* This technique normalizes depth z to range of [0, 1] before encoding process, and thus the depth z range could be large or small.

However, the proposed technique is still limited to encode one side of the surface, meaning that the back surface information will be lost. Therefore, setting up the viewing angle becomes vital to encoding the most important data coming from a range scanner. Nevertheless, this technique is especially valuable if it is directly linked with a 3D range scanning device, since the view can be set up to be the same as the real camera's view. By this means, minimum information will be lost, but the storage space can be drastically saved.

5. Summary

We have presented the idea of naturally encoding 3D range data into regular 2D images utilizing the advanced CGRP. We have demonstrated the viability of the proposed technique. Experimental data showed that this technique does not have the spatial resolution limitation of the previously proposed Holoimage encoding technique. Moreover, this proposed technique has the potential to instantaneously compress and transport 3D live videos captured from 3D range scanning devices.

References

1. N. Karpinsky and S. Zhang, "Holovideo: Real-time 3D video encoding and decoding on GPU," Opt. Lasers Eng. **50**, 280–286 (2012).
2. S. Gumhold, Z. Kami, M. Isenburg, and H.-P. Seidel, "Predictive point-cloud compression," *SIGGRAPH 2005 Sketches 137* (ACM, 2005).
3. B. Merry, P. Marais, and J. Gain, "Compression of dense and regular point clouds," Comput. Graph. Forum **25**, 709–716 (2006).
4. R. Schnabel and R. Klein, "Octree-based point-cloud compression," *Proceedings of IEEE/Eurographics Symposium on Point-Based Graphics* (IEEE, 2006), pp. 111–120.
5. A. Jones, M. Lang, G. Fyffe, X. Yu, J. Busch, I. McDowall, M. Bolas, and P. Debevec, "Achieving eye contact in a one-to-many 3D video teleconferencing system," *SIGGRAPH '09* (ACM, 2009).
6. X. Gu, S. Zhang, L. Zhang, P. Huang, R. Martin, and S.-T. Yau, "Holoimages," in *'06 ACM Symposium on Solid and Physical Modeling* (ACM, 2006), pp. 129–138.
7. N. Karpinsky and S. Zhang, "Composite phase-shifting algorithm for three-dimensional shape compression," Opt. Eng. **49**, 063604 (2010).
8. H. Schreiber and J. H. Bruning, *Optical Shop Testing*, 3rd ed. (Wiley, 2007), pp. 547–666.
9. J. Novak, P. Novak, and A. Miks, "Multi-step phase shifting algorithms insensitive to linear phase shift errors," Opt. Commun. **281**, 5302–5309 (2008).
10. D. C. Ghiglia and M. D. Pritt, *Two-Dimensional Phase Unwrapping: Theory, Algorithms, and Software* (Wiley, 1998).
11. S. Zhang and P. S. Huang, "Novel method for structured light system calibration," Opt. Eng. **45**, 083601 (2006).