Three-bit representation of three-dimensional range data

Nikolaus Karpinsky, Yajun Wang, and Song Zhang*

Department of Mechanical Engineering, Iowa State University, Ames, Iowa 50011, USA

*Corresponding author: song@iastate.edu

Received 20 November 2012; revised 5 March 2013; accepted 5 March 2013;
posted 6 March 2013 (Doc. ID 180229); published 4 April 2013

Our previous research has shown that 3D range data sizes can be substantially reduced if they are converted into regular 2D images using the Holoimage technique. Yet, this technique requires all 24 bits of a standard image to represent one 3D point, making it impossible for a regular 2D image to carry 2D texture information as well. This paper proposes an approach to represent 3D range data with 3 bits, further reducing the data size. We demonstrate that more than an 8.2:1 compression ratio can be achieved with compression root-mean-square error of only 0.34%. Moreover, we can use another bit to represent a black-and-white 2D texture, and thus both 3D data and 2D texture images can be stored into an 8 bit grayscale image. Both simulation and experiments are presented to verify the performance of the proposed technique. © 2013 Optical Society of America

OCIS codes: 120.2650, 100.5070, 100.6890.

1. Introduction

Advancements in real-time 3D scanning are being made at an unprecedented rate, driving the technology further into mainstream life, as can be seen from real-time 3D scanners, such as the Microsoft Kinect [1,2]. With these advancements, large amounts of data are being generated, bringing forth the challenge of streaming and storing this information in an efficient manner. Classical geometry compression approaches compress the 3D geometry and its attributes, such as normals, texture coordinates, etc., in a model format such as OBJ, PLY, or STL. Though these formats work well for static scans or structured meshes, the same does not hold true for 3D scans from a real-time 3D scanner due to its unstructured nature [3].

To address this challenge, newer approaches better suited to data coming from 3D scanners have been developed, including heuristic-based point-cloud encoding [4,5] and image-based encoding approaches [6–8]. Image-based encoding approaches work well, as the geometry can be projected into images, and then 2D image compression can be utilized until 3D reconstruction is desired. Since 2D image compression is a long studied field, high compression ratios with relatively low amounts of error can be achieved.

Holoimage [8] is an image-based encoding technique that allows for real-time encoding and decoding at high compression ratios. It leverages techniques from optical metrology, namely fringe projection. Due to the error tolerance in fringe projection, the fringe patterns can be highly compressed with little error to the reconstructed 3D geometry. Karpinsky and Zhang [9] proposed to utilize the Holoimage technique and Hou *et al.* [10] proposed a similar virtual structured light technique to compress 3D geometry. Based on Holoimage's real-time encoding and decoding, it is able to compress data from real-time 3D scanners [3]. With these merits, it is well suited as a format for high-speed 3D scans, which can then be streamed and stored.

Although Holoimage is a good technique for compressing 3D geometry from a real-time 3D scanner, it still uses 24 bits to represent a 3D coordinate, which in practice takes up the three standard image

1559-128X/13/112286-08$15.00/0

channels [red, green, and blue (RGB)]. With this representation, there is no room in a standard image for other information, such as a texture or a normal map. This research addresses this by representing the image with only 3 bits instead of 24 through the use of image dithering. This leaves 21 remaining bits for other information, such as texture or normal maps, allowing for more information to be stored and streamed. With this new encoding, compression ratios of 8.1:1 have been achieved when compared with a 24 bit Holoimage with a mean-squared error of 0.34%.

Section 2 explains the principle behind Holoimage, applying image dithering, and how it fits into the Holoimage pipeline. Section 3 shows experimental results of a 3D unit sphere and a David bust and discusses the findings. Finally, Section 4 summarizes the paper.

2. Principle

A. Holoimage Encoding and Decoding

Holoimage is a form a 3D geometry representation that is well suited to quickly and efficiently compress 3D geometry coming from 3D scanners [9]. It works off of the principle of fringe projection from optical metrology. Encoding works by creating a virtual fringe projection system and virtually scanning 3D geometry into a set of 2D images that can then later be used to decode back into 3D. Figure 1 shows a conceptual model of the holovideo system. The projector projects a pattern onto the geometry, which can be done using OpenGL shaders [3], and then the camera captures the resulting scene, which can be done by saving the framebuffer as an image. Once in the image format, standard 2D image-processing techniques, such as compression or dithering can be applied.

Details of the holoimaging encoding and decoding algorithms have been thoroughly discussed in [3]; we only briefly explain these algorithms here. The Holoimage encoding colors the scene with the structured light pattern. To accomplish this, the model view matrix of the projector is rotated around the z axis by some angle (e.g., $\theta = 30°$) from the camera matrix. Each point is colored with the following three equations:

$$I_r(x, y) = 0.5 + 0.5\,\sin(2\pi x/P), \tag{1}$$

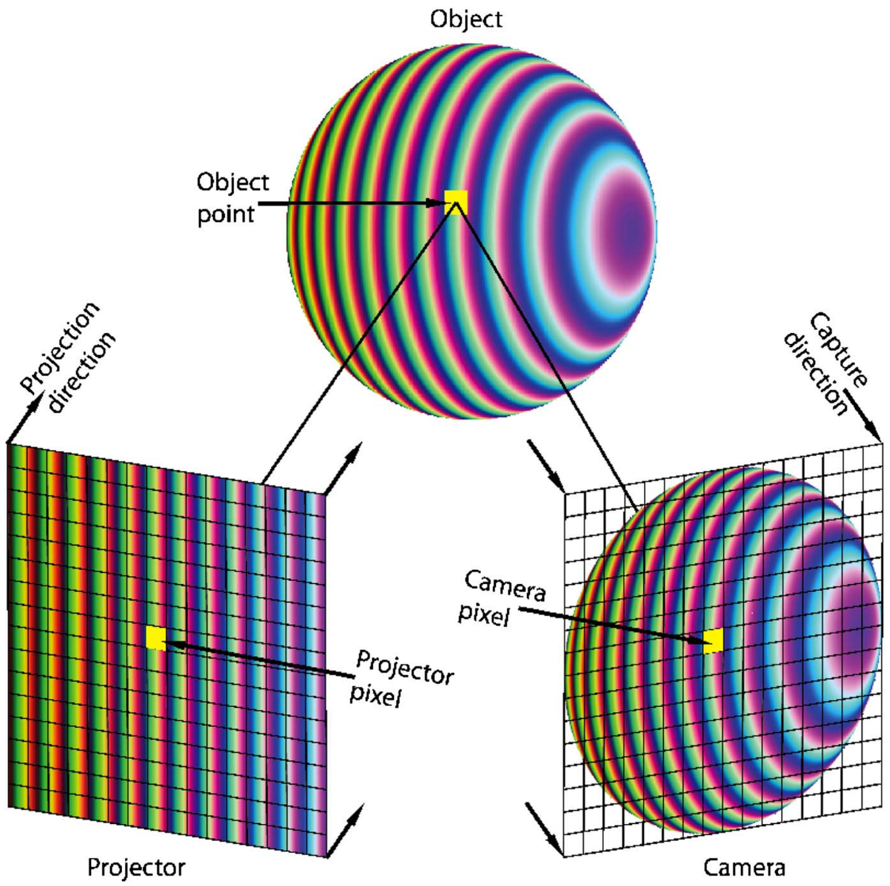

Fig. 1. (Color online) Holovideo system conceptual model. The virtual projection system projects sinusoidal fringe patterns onto the object; the result is rendered by the graphics pipeline, and then displayed on the screen. The screen view acts as a virtual camera imaging system. Because both the projector and the camera are virtually constructed, they can both be orthogonal devices. The angle between the projection system and the camera imaging system is θ.

$$I_g(x,y) = 0.5 + 0.5\cos(2\pi x/P), \quad (2)$$

$$I_b(x,y) = S \cdot Fl(x/P) + S/2 + (S-2)/2 \cdot \cos[2\pi \cdot \mathrm{Mod}(x,P)/P_1], \quad (3)$$

where P, the fringe pitch represents 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 the grayscale intensity value, $\mathrm{Mod}(a,b)$ is the modulus operator to get a over b, and $Fl(x)$ is used to obtain the integer number of x by removing the decimals.

Decoding the resulting Holoimage is more involved than encoding involving four major steps: (1) calculating the phase map from the Holoimage frame, (2) filtering the phase map, (3) calculating normals from the phase map, and (4) performing the final render. A multipass rendering was utilized to accomplish these steps, saving results from the intermediate steps to a texture, which allowed us to access neighboring pixel values in proceeding steps.

Equations (1)–(3) provide the phase uniquely for each point,

$$\Phi(x,y) = 2\pi \times Fl[(I_b - S/2)/S] + \tan^{-1}[(I_r - 0.5)/(I_g - 0.5)]. \quad (4)$$

It should be noted that this phase is already unwrapped, and thus no spatial phase unwrapping is required for this process. From the unwrapped phase $\Phi(x,y)$, the normalized coordinates (x^n, y^n, z^n) can be decoded as [9]

$$x^n = j/W, \quad (5)$$

$$y^n = i/W, \quad (6)$$

$$z^n = \frac{P\Phi(x,y) - 2\pi i \cos(\theta)}{2\pi W \sin\theta}. \quad (7)$$

This yields a value z^n in terms of P, which is the fringe pitch, i, which is the index of the pixel being decoded in the Holoimage frame, θ, which is the angle between the capture plane and the projection plane ($\theta = 30°$ for our case), and W, which is the number of pixels horizontally.

From the normalized coordinates (x^n, y^n, z^n), the original 3D coordinates can recovered point by point:

$$x = x^n \times S_c + C_x, \quad (8)$$

$$y = y^n \times S_c + C_y, \quad (9)$$

$$z = z^n \times S_c + C_z. \quad (10)$$

Here S_c is the scaling factor to normalize the 3D geometry, and (C_x, C_y, and C_z) are the center coordinates of the original 3D geometry.

B. Image Dithering

Image dithering is the process of taking a higher color depth image and reducing the color depth to a lower level through a quantization technique [11]. Different types of image-dithering techniques exist, such as ordered dithering [12] and error diffusing [13]. In this research, two of the most popular algorithms were investigated: Bayer [12] and Floyd–Steinberg [14] dithering.

1. *Bayer Dithering*

Bayer dithering, sometimes known as ordered dithering, involves quantizing pixels based on a threshold matrix [12]. In the simple case of quantizing to a binary image, it involves taking each pixel in an image and applying Algorithm 1:

Algorithm 1: Bayer dithering

```
Input: Pixel—Structure representing properties of a pixel in
       an image. Has color components ranging from 0.0 to 1.0
Input: ThresholdMap—Matrix of threshold values
Ouput: Pixel.color—Pixel's dithered color component,
       either 0 or 1
for Each Pixel do
    if Pixel.color >= ThresholdMap[pixel.x mod map Width]
      [pixel.y mod mapHeight]
    then
        Pixel.color = 1;
    else
        Pixel.color = 0;
    end
end
```

$$M = \frac{4.0}{255.0} * \begin{bmatrix} 0 & 32 & 8 & 40 & 2 & 34 & 10 & 42 \\ 48 & 16 & 56 & 24 & 50 & 18 & 58 & 26 \\ 12 & 44 & 4 & 36 & 14 & 46 & 6 & 38 \\ 60 & 28 & 52 & 20 & 62 & 30 & 54 & 22 \\ 3 & 35 & 11 & 43 & 1 & 33 & 9 & 41 \\ 51 & 19 & 59 & 27 & 49 & 17 & 57 & 25 \\ 15 & 47 & 7 & 39 & 13 & 45 & 5 & 37 \\ 63 & 31 & 55 & 23 & 61 & 29 & 53 & 21 \end{bmatrix}. \quad (11)$$

Equation (11) gives an example of an 8×8 threshold matrix, which was also the matrix used in this work.

Bayer has shown that if the sizes of the matrices are 2^N (N is an integer), then optimal matrices can be derived; the matrices can be obtained as follows:

$$M_1 = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}, \quad (12)$$

which is the smallest 2×2 base dither pattern. Larger dither patterns can be obtained using

$$M_{n+1} = \begin{bmatrix} 4M_n & 4M_n + 2U_n \\ 4M_n + 3U_n & 4M_n + U_n \end{bmatrix}, \tag{13}$$

where U_n is an n-dimensional unit matrix (one for all elements). Using larger threshold matrices allows more distinct tones to be represented in the final image; thus larger threshold matrices could result in lower error. In this research, we used an 8×8 since it is a large matrix that should theoretically yield 64 different tones in the resulting image. However, we also found that if the matrix is too large, the resultant dithered pattern is not of high quality. We typically use 8×8 or 16×16 matrices of our study.

With Bayer dithering, the threshold map adds minor local error noise to the quantized pixel, but the overall intensity is preserved. Since this algorithm is a parallel algorithm, it can easily be integrated into the Holoimage pipeline in the fragment shading stage of the encoding, allowing for little to no overhead in encoding.

2. *Floyd–Steinberg Dithering*

Floyd–Steinberg dithering is a form of error diffusing dither, which diffuses quantization error of a specific pixel into neighboring pixels [14]. Through error diffusing, the cumulative quantization error is kept to a minimum, which is near zero. The original Floyd–Steinberg dithering algorithm is given with Algorithm 2.

Algorithm 2: Floyd–Steinberg dithering

```
Input: Image—Original Image to be dithered. Has color
       components ranging from 0.0 to 1.0
for $y \leftarrow 0$ to Image.Height do
    for $x \leftarrow 0$ to Image.Width do
        if Image.Pixel($x,y$).color >= 0.5 then
            newColor = 1;
        else
            newColor = 0;
        end
        quantError = Image.Pixel($x,y$)-newColor;
        Image.Pixel($x,y$) = newColor;
        //Diffuse Error;
        Image.Pixel($x+1,y$)+ = 7/16 * quantError;
        Image.Pixel($x-1,y+1$)+ = 3/16 * quantError;
        Image.Pixel($x,y+1$)+ = 5/16 * quantError;
        Image.Pixel($x+1,y+1$)+ = 1/16 * quantError;
    end
end
```

In the first part of the algorithm, the image's pixel value is quantized into either 1 or 0. Then the quantization error from this operation is calculated, and then diffused into neighboring pixels, to the right and down. It should be noted that unlike ordered dithering, this algorithm is a serial algorithm, operating on the image pixels one by one. In the standard Floyd–Steinberg dithering algorithm, the route is from left to right, top to bottom, realized as a forward array. Once a pixel has been quantized, it is no longer changed. However, we did find that the resultant image quality depends on the starting location and path of diffusing the error [15]. Another zigzag route could be taken, but the algorithm would have to be altered slightly so that the error is not diffused into pixels that have been dithered in the new zigzag route.

3. Experiments

To test the effects of image dithering on Holoimages, we performed both Bayer and Floyd–Steinberg dithering on Holoimages of a unit sphere and 3D scan of the statue of David. In all of our experiments we had a fringe frequency of 12, θ of 30 deg, and Holoimage size of 512×512.

To begin, we performed the dithering on the unit sphere and then stored the resulting images in the lossless portable network graphics (PNG) format. Figures 2 and 3 show the results. Figure 2(a) shows the Holoimage. RGB channels of the Holoimage are then dithered with the Bayer-dithering technique individually, and then stored into the three most significant bits of the 8 bit grayscale image shown in Fig. 2(b), with R being stored as the most significant bit, and B being stored as the third most significant bit. This grayscale image contains all the information required to recover the whole 3D geometry carried on by the 24 bit Holoimage shown in Fig. 2(a). Similarly, the other dithering technique can also be employed to convert the 24 bit Holoimage into the three most significant bits of an 8 bit grayscale image. Figure 2(c) shows the dithered image using the Floyd–Steinberg dithering technique.

Before the 3D geometry can be decoded from the Holoimage, 2D image processing needs to be reversed to attempt to put the Holoimage back into its original state. In terms of dithering, this can be done by applying a low-pass filter, such as a Gaussian filter, to the dithered image. In this research, we used a 7×7 Gaussian filter with a standard deviation of 7/3 pixels. It is also important to know that in the Holoimage pipeline, filtering can be applied after phase unwrapping. Previous work has shown that median filtering can remove spiking noise in the final reconstruction [16,17]. This is done by median filtering, and then instead of using the median, detecting the correct number of phase jumps from the median and applying it to the phase at the current pixel.

Figure 2(e) shows the reconstructed result from the Bayer-dithered pattern shown in Fig. 2(b). In comparison with the 3D result recovered from the 24 bit Holoimage shown in Fig. 2(d), the Bayer-dithered result has some random noise on top of the recovered 3D results. Yet, the sphere was well recovered. Figure 2(f) shows the recovered results using the Floyd–Steinberg dithering technique, which are significantly better than the results obtained from the Bayer-dithering technique.

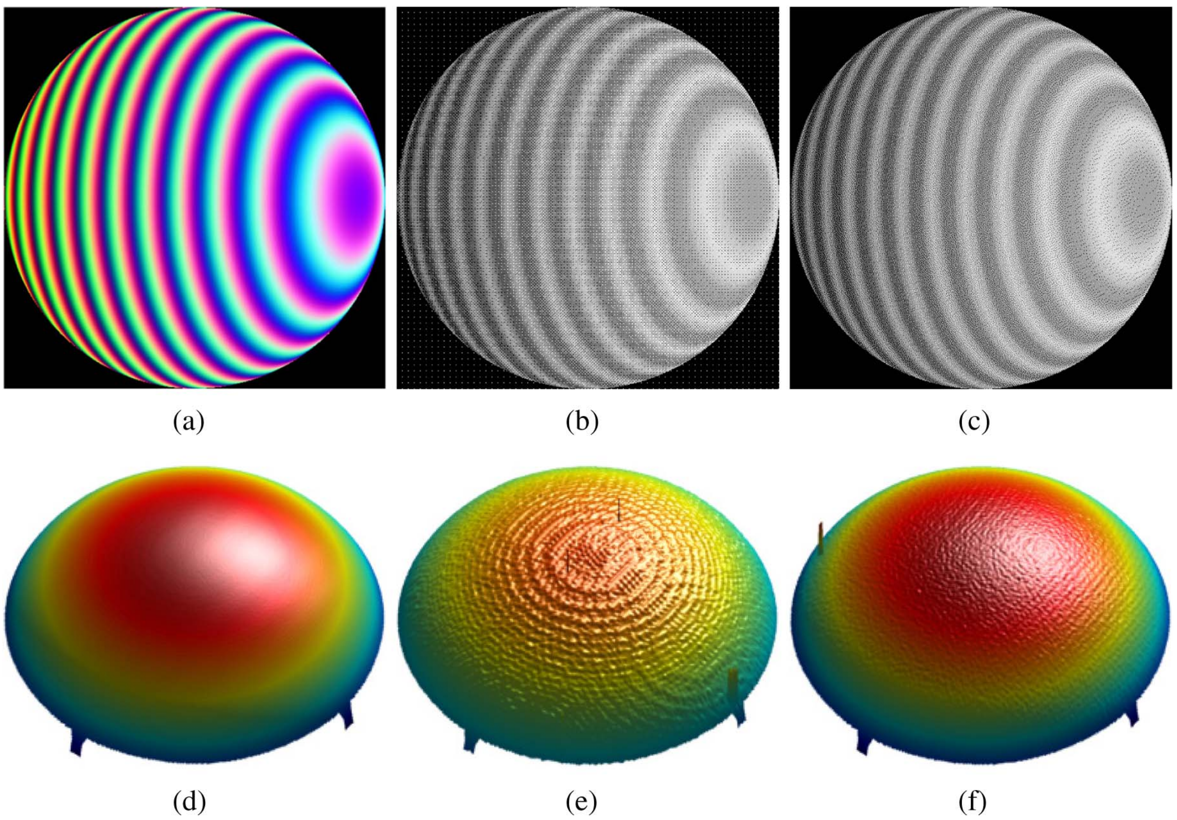

Fig. 2. (Color online) Results of dithering on unit sphere in a lossless image format. (a) Original Holoimage, (b) Holoimage with Bayer dithering, (c) Holoimage with Floyd–Steinberg dithering, (d) 3D reconstructed results for image shown in (a), (e) 3D reconstructed results for image shown in (b), and (f) 3D reconstructed results for image shown in (c).

To better compare these dithering techniques, Fig. 3 shows the cross sections of the recovered 3D results using different methods compared with the ideal unit sphere. Figures 3(a) and 3(d), respectively, show the cross section of the recovered 3D sphere overlapping with the ideal unit sphere, and the cross section of the difference between these two, when the 24 bit Holoimage is used. The results are smooth, and the error is small, which has been demonstrated previously [3]. The Bayer-dithered results [Figs. 3(b) and 3(e)] show that the overall geometry was recovered quite well, but the error is larger: an approximate root-mean-square (rms) error of 0.33%. It can be seen that this error is still quite small. Yet, the Floyd–Steinberg dithering technique can further improve the results, as shown in Figs. 3(c) and 3(f). This is due to the quantization error being diffused into neighboring pixels, reducing the overall quantization error. The error can be further reduced to have an approximate rms error of 0.2%. Is should be noted that only 3 bits were used to represent the 24 bit Holoimage, and the reconstructed geometry is still of high quality.

Compression results depend on how the resulting dithered information is stored. In this work JPEG and other lossy image compression was not used due to the fact that it makes use of a low-pass filter before compression. This takes the 3 bit binary dithered information and transforms it back into 24 bit information, which is undesirable. Instead, PNG, a lossless image compression, was utilized, and the three most significant bits of a grayscale image were utilized, shown by Fig. 4(a). This resulted in a file size of 79 kB with the unit sphere. Further compression can be achieved by saving the image in a planar format, three times as wide with image channels one after another, and then saving the PNG as a logical 1 bit image. This resulted in a file size of 62 kB, yielding a compression ratio of 3.9:1 when compared against a 24 bit Holoimage in the PNG format.

To further test dithering on Holoimages, the technique was performed on a scan of the statue of David shown in Fig. 5. Figure 5(a) shows the 24 bit Holoimage, and Fig. 5(d) shows the recovered 3D geometry. The 24 bit Holoimage is then dithered into 3 bits and stored into the three most significant bits of an 8 bit grayscale image. Figures 5(b) and 5(c), respectively, show the Bayer-dithered result and the Floyd–Steinberg dithered result, and their recovered 3D shapes are shown in Figs. 5(e) and 5(f). Again, it can be seen that Bayer dithering results in larger amounts of error seen as ripples and bumps on the surface; Floyd–Steinberg dithering has some of these errors as well, but they are not as prominent as in the case with Bayer dithering. Floyd–Steinberg dithering results a lower rms error of 0.34% when compared to Bayer dithering at 0.37%. The resulting file size is 39 kB, achieving a compression ratio of

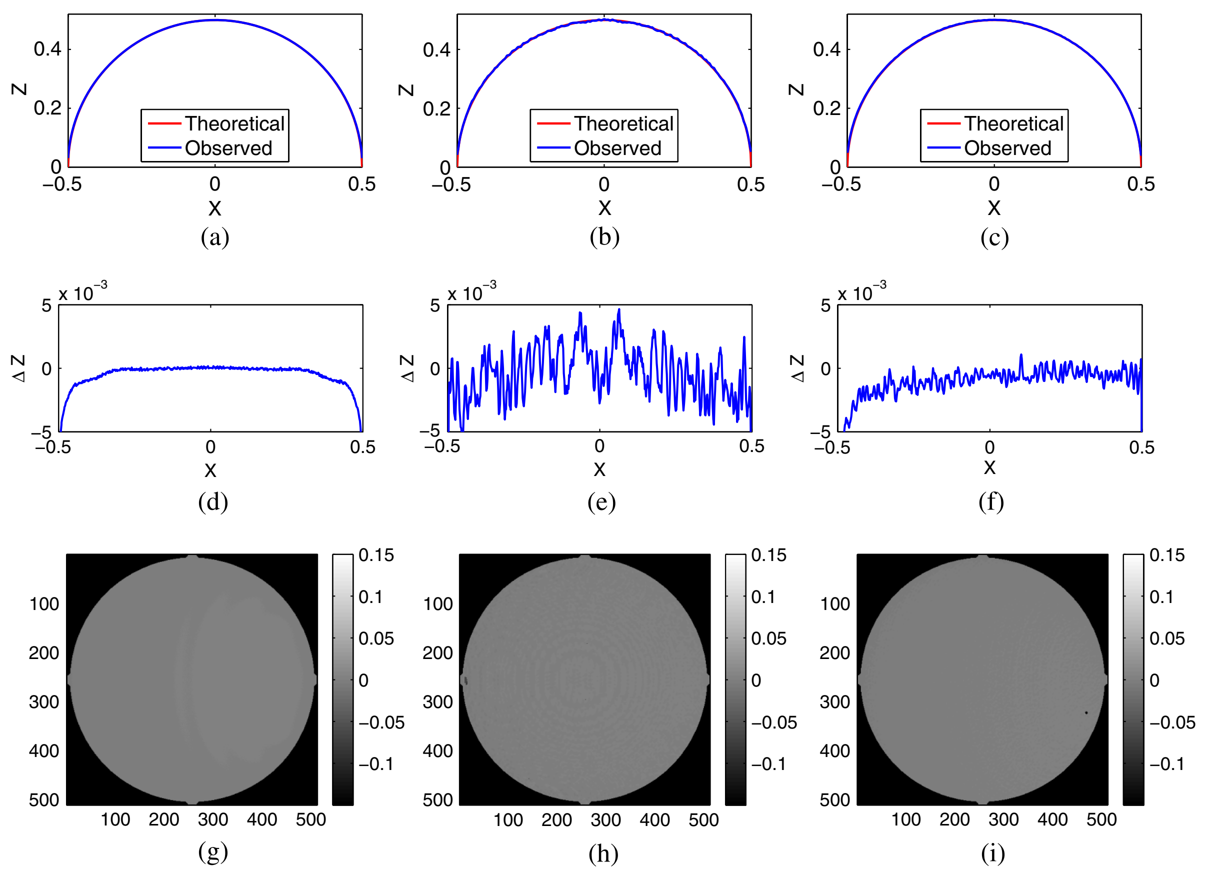

Fig. 3. (Color online) Reconstruction errors of dithering on unit sphere in a lossless image format. (a) Cross section of reconstructed result shown in Fig. 2(d), (b) cross section of reconstructed results shown in Fig. 2(e), (c) cross section of reconstructed result shown in Fig. 2(f), (d) reconstruction error between the reconstructed and ideal unit sphere for the result in (a), (e) reconstruction error between the reconstructed and ideal unit sphere for the result in (b) (approximate rms error 0.33%), (d) reconstruction error between the reconstructed and ideal unit sphere for the result in (c) (approximate rms error 0.2%), and (g)–(i) difference map of technique to ideal unit sphere.

8.2:1 when compared to the 24 bit Holoimage. Although it might be expected that a simple unit sphere would have a higher compression, this is not the case, as PNG compression depends on precompression and DEFLATE steps, which can result in different file sizes for similar images.

Since the proposed technique only requires 3 bits to represent the whole 3D geometry, there are 21 bits remaining to encode more information such as the grayscale texture that comes from the 3D scanner, which can be encoded into the same image. There are essentially two approaches to carry on texture with 3D geometry. The first method is to pack the 8 bit grayscale image directly into the 24 bit image. Figure 6(a) shows the resultant image, and its recovered 3D geometry with texture mapping is shown in Fig. 6(b). The file size is approximately 189 kB, which is a substantial reduction compared with the original 24 bit Holoimage stored in PNG format 320 kB.

The 8 bit texture image can be dithered as well to further compress the data. Figure 6(c) shows the packed image that stores the 3D geometry along with

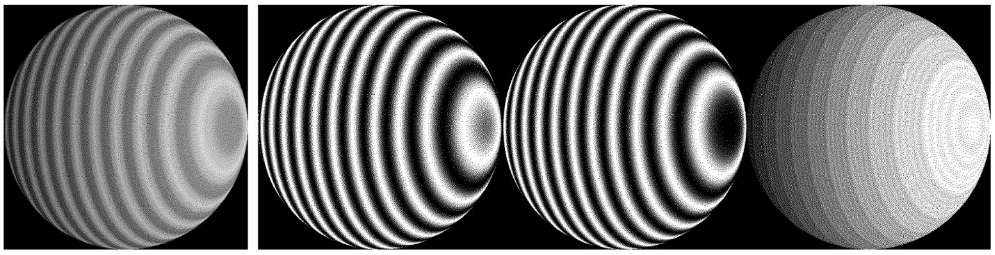

Fig. 4. Different ways to hold a packed dithered Holoimage. (a) Dithered channels packed in the three most significant bits and saved as a grayscale PNG with resulting file size of 79 kB. (b) Dithered channels packed into a planar format and then saved as a logical PNG with resulting file size of 62 kB.

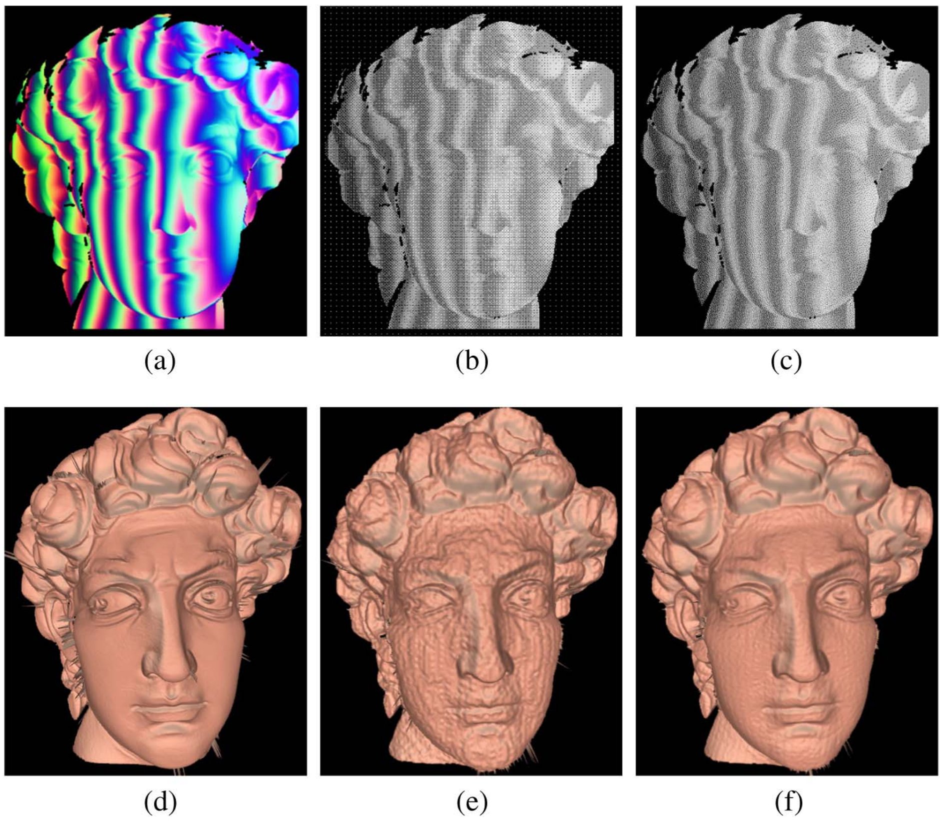

Fig. 5. (Color online) Results of dithering on scan of David statue in a lossless image format. (a) Original Holoimage, (b) Holoimage with Bayer dithering, (c) Holoimage with Floyd–Steinberg dithering, (d) recovered 3D geometry from (a), (e) recovered 3D geometry from (b), and (f) recovered 3D geometry from (c).

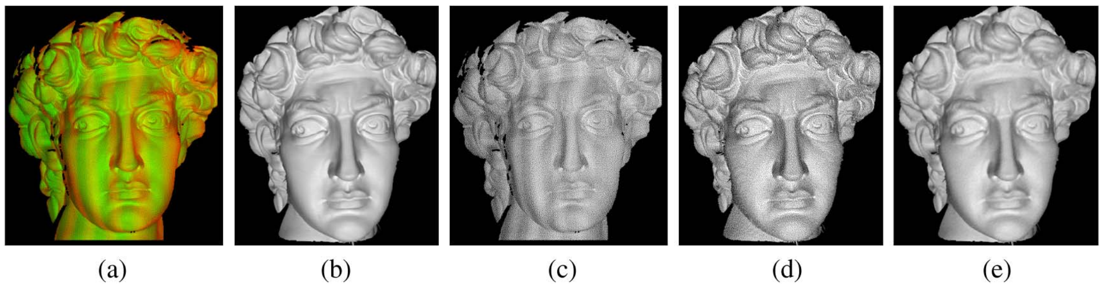

Fig. 6. (Color online) Packing dithered Holoimage with texture. (a) 3 bit packed Holoimage with 8 bit grayscale texture, (b) 3D geometry with original 8 bit texture mapping, (c) 3 bit packed Holoimage with 1 bit dithered texture, (d) 3D geometry with 1 bit dithered texture mapping, and (e) 3D geometry with 1 bit dithered texture after texture is Gaussian filtered.

the 1 bit dithered texture image into the four most significant bits of an 8 bit grayscale image. From this image, the texture can be recovered by applying a very small Gaussian filter (7×7) as shown in Fig. 6(d). It can be seen that the texture image is of good quality. With only 4 bits, the file size is approximately 64 kB. This example clearly demonstrates that the proposed technique can embed both the 3D geometry and the texture into a regular 2D image, making it a novel technique to store 3D range data in a substantially reduced size, with minor loss of quality. Furthermore, because it only utilizes 4 bits, this proposed 3D range data compression technique can be leveraged for applications in which other critical information, such as connectivity or bump maps, needs to be carried along.

4. Conclusion

An approach to represent 3D geometry has been presented, specifically applying image dithering to the Holoimage technique to reduce the bit depth from 24 to 3 bits. The technique was presented with two forms of image dithering, and sample data of a unit sphere and a 3D scan of David have been shown. A mean-squared error of 0.2% was achieved on the unit sphere with a compression of 3.9:1 when compared with the 24 bit Holoimage technique, and an rms error of 0.34% was achieved on the scan of David with a compression of 8.2 : 1 when compared with the 24 bit Holoimage. With the remaining 21 bits, grayscale texture information was also encoded, effectively embedding 3D geometry and texture into a single 8 bit grayscale image. Future work for this research includes extending it to video with animation appropriate dithering.

References

1. G. Geng, "Structured-light 3D surface imaging: a tutorial," Adv. Opt. Photon. **3**, 128–160 (2011).
2. S. Zhang, "Recent progresses on real-time 3-D shape measurement using digital fringe projection techniques," Opt. Lasers Eng. **48**, 149–158 (2010).
3. N. Karpinsky and S. Zhang, "Holovideo: real-time 3D video encoding and decoding on GPU," Opt. Lasers Eng. **50**, 280–286 (2012).
4. B. Merry, P. Marais, and J. Gain, "Compression of dense and regular point clouds," Comput. Graph. Forum **25**, 709–716 (2006).
5. S. Gumhold, Z. Kami, M. Isenburg, and H.-P. Seidel, "Predictive point-cloud compression," in *ACM SIGGRAPH 2005 Sketches* (ACM, 2005), pp. 137–141.
6. X. Gu, S. J. Gortler, and H. Hoppe, "Geometry images," ACM Trans. Graph. **21**, 355–361 (2002).
7. R. Krishnamurthy, B. Chai, and H. Tao, "Compression and transmission of depth maps for image-based rendering," in *Proceedings of 2001 International Conference on Image Processing* (IEEE, 2001), pp. 828–831.
8. X. Gu, S. Zhang, P. Huang, L. Zhang, S.-T. Yau, and R. Martin, "Holoimages," in *Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling* (ACM, 2006), pp. 129–138.
9. N. Karpinsky and S. Zhang, "Composite phase-shifting algorithm for three-dimensional shape compression," Opt. Eng. **49**, 063604 (2010).
10. Z. Hou, X. Su, and Q. Zhang, "Virtual structured-light coding for three-dimensional shape data compression," Opt. Lasers Eng. **50**, 844–849 (2012).
11. T. L. Schuchman, "Dither signals and their effect on quantization noise," IEEE Trans. Commun. Technol. **12**, 162–165 (1964).
12. B. Bayer, "An optimum method for two-level rendition of continuous-tone pictures," in *IEEE International Conference on Communications* (IEEE, 1973), pp. 11–15.
13. T. D. Kite, B. L. Evans, and A. C. Bovik, "Modeling and quality assessment of halftoning by error diffusion," in *IEEE International Conference on Image Processing* (IEEE, 2000), pp. 909–922.
14. R. W. Floyd and L. Steinberg, "An adaptive algorithm for spatial gray scale," J. Soc. Inf. Disp. **17**, 75–77 (1976).
15. W. Lohry and S. Zhang, "Genetic method to optimize binary dithering technique for high-quality fringe generation," Opt. Lett. **38**, 540–542 (2013).
16. N. Karpinsky and S. Zhang, "3D video compression with the H.264 codec," Proc. SPIE **8290**, 829012 (2012).
17. M. McGuire, "A fast, small-radius GPU median filter," *ShaderX6: Advanced Rendering Techniques* (Charles River Media, 2008).