applied optics

Three-dimensional range geometry compression via phase encoding

Tyler Bell,[1,2] Bogdan Vlahov,[1] Jan P. Allebach,[2] and Song Zhang[1,*]

[1]School of Mechanical Engineering, Purdue University, West Lafayette, Indiana 47907, USA
[2]School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana 47907, USA
*Corresponding author: szhang15@purdue.edu

Received 21 August 2017; revised 12 October 2017; accepted 17 October 2017; posted 18 October 2017 (Doc. ID 305219); published 16 November 2017

One of the state-of-the-art methods for three-dimensional (3D) range geometry compression is to encode 3D data within a regular 24-bit 2D color image. However, most existing methods use all three color channels to solely encode 3D data, leaving no room to store other information (e.g., texture) within the same image. This paper presents a novel method which utilizes geometric constraints, inherent to the structured light 3D scanning device, to reduce the amount of data that need be stored within the output image. The proposed method thus only requires two color channels to represent 3D data, leaving one channel free to store additional information (such as a texture image). Experimental results verify the overall robustness of the proposed method. For example, a compression ratio of 3038:1 can be achieved, versus the STL format, with a root-mean-square error of 0.47% if the output image is compressed with JPEG 80%.

OCIS codes: (120.2650) Fringe analysis; (100.5070) Phase retrieval; (100.6890) Three-dimensional image processing.

https://doi.org/10.1364/AO.56.009285

1. INTRODUCTION

Three-dimensional (3D) scanning technologies have the ability to capture high-quality data at real-time speeds [1]. Such abilities have led to an increased adoption of these technologies within many various industries, such as medicine, communication, entertainment, and manufacturing. Given the large amounts of data generated by 3D scanning technologies, the real-time storage and transmission of such data becomes important.

One way to represent 3D data is with a mesh format. A mesh is described by a set of *vertices* (3D coordinates) and a set of *edges*, which specify the structure of the mesh (i.e., how the coordinates should be *connected* to one another). Some additional attributes of the mesh may also be stored, such as a normal map, vertex colors, or a texture image along with texture image coordinates. Standard mesh file formats (e.g., OBJ, STL, PLY) are based on simple listings of the information required to reconstruct a mesh with its attributes. In the past several decades, much work has been done to try and represent this information as efficiently as possible.

Researchers have sought new ways to efficiently encode a mesh's connectivity information in order to reduce the amount of information needed overall to represent the mesh. Connectivity information can be efficiently encoded using intelligent methods of traversing the vertices or structures within the mesh. A well-designed encoding method reduces redundancy within the connectivity information, thus reducing overall file sizes, and many methods have been proposed (e.g., triangle strip [2–4], spanning tree [5], valence encoding [6,7], triangle traversal [8,9]). Once connectivity information has been encoded, the actual positions of the vertices are then encoded. This is typically done by following a three-step procedure of quantization, prediction, and entropy encoding [10,11].

The above methods are connectivity-driven, meaning that the encoding of geometry information follows the order of the connectivity encoder. Given that the data size for a 3D mesh is generally more impacted by geometry (i.e., coordinate) data [10,11], there have also been geometry-driven methods developed for the compression of 3D meshes. Such methods let the encoding be driven by what best encodes the coordinate positions, even if it does not result in an optimal encoding of the connectivity information. For example, Kronrod and Gotsman [12] proposed a method which optimizes the predictions between the positions of adjacent vertices. The connectivity information would then be encoded by following the optimized predictions. It was found that this optimization could provide much more compact meshes overall while paying only a small penalty for the non-optimal connectivity encoding [12]. Mesh compression problems become more simple if precise restoration of a mesh's connectivity is not required, or if the data has an underlying structure that automatically carries the

connectivity information. For example, a regular grid or pixel structure is often assumed with range data captured by a camera. In such cases, the encoding methods can primarily focus on how to precisely and efficiently encode 3D data itself.

In the field of phase-shifting interferometry digital holography, 3D hologram information is encoded within a 2D complex wavefront. Multiple phase-shifted interference, or *fringe*, patterns are captured by a camera and these patterns can recover both amplitude and phase information. Deformations of the phase from a reference plane are related to deviations of the object surface from the reference plane being captured. Typically, the information within the fringe patterns are used to derive a complex wavefront, or Fresnel field, which is used to reconstruct the captured object. As these wavefronts consist of complex, floating point values, methods for compressing the data are desired.

Since digital holograms have an inherent grid structure (the wavefront is computed from data digitally recorded by a camera), the 3D geometry compression problem is simplified to a 2D compression problem. Furthermore, instead of compressing the wavefront and its complex values, Darakis and Soraghan [13] proposed a digital hologram compression method which applies JPEG and JPEG 2000 compression directly to the camera captured interference patterns, from which a complex wavefront can later be computed. This method is flexible due to the ability to define and control the compression rates (i.e., the JPEG quality level in use). However, to achieve higher compression ratios, lower JPEG qualities are used, which causes considerable error on the reconstructed wavefront. Further, the data size is proportional to the number of phase-shifted patterns captured by the camera. If this number is increased, compressing the wavefront and its complex values may be more efficient.

Darakis and Soraghan [14] proposed a method for compressing a complex wavefront—at the object's reconstruction plane—which first quantizes the complex data and then losslessly encodes it using the Burrows–Wheeler transform [15]. This method outperforms the method directly compressing the interference patterns using JPEG that the same team proposed earlier [13]. It achieved reasonably good compression ratios [e.g., approximately 26:1 for a normalized root-mean-square (RMS) error of approximately 0.1], and retained the hologram's natural capability of being able to be reconstructed at different depths and perspectives. More thorough reviews of state-of-the-art compression methods using various digital holography approaches are given by Alfalou and Brosseau [16] and Dufaux *et al.* [17].

The physical properties of digital holography systems (e.g., lighting conditions, surface texture, speckle noise) could greatly affect the efficiency of hologram compression methods. To alleviate such potential problem, a *virtual* digital holography system can be used to create computer-generated holograms (CGHs). These are generated by numerically simulating how light reflects and propagates off of a virtual 3D object. CGH methods are advantageous as they can both represent arbitrary 3D objects and are computed within a completely ideal environment. Recently, methods have been proposed for compressing CGHs using JPEG [18] or even high efficiency video coding (HEVC) [19]. Although these compression methods are quite effective, generating the CGHs themselves is both a computationally complex and memory expensive process [20]. Graphic processing units can be used to effectively reduce the time of computing CGHs [21–23], yet such a method is still limited by the amount of on-board memory for high-resolution hologram generation. Moreover, since the viewing angle of the reconstructed image is proportional to the CGH size [23], the resulting CGH may have a very large spatial resolution compared to the number of points actually encoded. In general, all the compression approaches based on digital holograms suffer from the noise caused by speckle. The presence of speckle noise makes it difficult to fully leverage 2D lossy image compression methods, hindering the ability to achieve very high compression ratios while also preserving data quality after compression.

Compared to holography-based 3D geometry compression methods, digital fringe projection (DFP)-based 3D range data compression methods typically have the advantages of (1) one-to-one correspondence between a pixel on an image and one encoded 3D geometry point; (2) the elimination of speckle noise related problems; and (3) the ability to achieve much higher compression ratios with standard 2D image compression techniques (e.g., a magnitude higher for high-quality compression). Similar to the concept of using a virtual digital holography system to calculate CGHs, a virtual DFP system can be used to precisely and quickly encode 3D coordinates within the three channels (RGB) of a regular 2D image using phase-shifting techniques. Once 3D range geometry is encoded into a 2D image, it can then be further compressed using well-established image compression techniques, such as PNG or JPEG, and saved to disk or transmitted over a network.

Researchers have proposed different approaches to encode 3D geometry into a 2D image using the concept of a virtual DFP system along with phase-shifting principles [24–26]. These methods use all three color channels of the output RGB image to encode 3D data. Typically two of the three color channels will be used to represent the 3D data. The third color channel is used to store important *fringe order* information, which is needed for the proper recovery of the phase-shifted data within the first two channels. Although these methods are successful, using all three color channels of the output image limits the ability to save any additional information with the 3D data (e.g., a texture image). Given this, some methods have focused on encoding 3D geometry in such a way that it only uses two of the three color channels of the output 2D image.

Hou *et al.* [27] proposed a two-channel method which was able to represent 3D geometry with one single channel of the output image, still using a second channel to store the fringe order information for decoding. Using one channel instead of two to represent 3D geometry information then leaves one channel free, either to be left empty or to store additional attributes of the data. Although efficient in this regard, this method uses only a single 8-bit color channel to represent 3D geometry limiting the precision of the encoding. Further, the data this method encodes to represent 3D geometry contains very sharp discontinuities, which results in rapid intensity changes between pixels of the output image. This limits the method's

potential extension to using lossy JPEG compression to further reduce file sizes, as sharp intensity changes can cause compression artifacts within the encoded data.

Wang *et al.* [28] also proposed a method which was able to encode 3D geometry using only two color channels of an RGB image, leaving one channel open for additional information. This method also uses one channel to represent 3D geometry information and another to store the fringe order information needed for decoding. While successful, the method has the same drawback that it only uses a single color channel to represent 3D geometry, limiting the precision of the encoding. Further, it requires a post-processing error compensation framework to alleviate decoding errors.

Karpinsky *et al.* [29] proposed a method which encoded 3D geometry into three color channels and then performed dithering on each one. Using this method, each 8-bit channel could be represented with a single bit per pixel, allowing all three color channels to be represented with only three bits per pixel. This method is quite advantageous in terms of its small file sizes and large amounts of remaining space within the output image for the storage of additional information (such as a texture image). In fact, the texture image itself can be dithered along with the geometry information in order to reduce data sizes even further. The main drawback of this method was that it required the usage of a lossless image compression technique (i.e., PNG) when storing the dithered channels. Ideally, lossy image compression techniques could be used to further decrease file sizes; however, if a lossy method (i.e., JPEG) was used to store the dithered channels, the resulting file sizes were larger than PNG. Since almost all of the widely used video codecs (e.g., H.264) employ some sort of lossy image compression, this would limit this encoding method's extension to 3D video applications.

One trait that is common in the methods that use phase-shifting concepts to represent 3D data within a 2D image is the need to encode the fringe order information. This is because the fringe order value for each 3D coordinate (or its associated 2D pixel) needs to be known in order to perform proper decoding of the coordinate. If there were another means to derive the fringe order information (instead of encoding it directly within the output image), up to an entire channel could be saved or used to store additional information.

This paper proposes a novel method for 3D range geometry compression which utilizes the geometric constraints of the 3D capture system itself to derive this fringe order information, necessary for proper data decoding, in an on-demand fashion using the system's calibration parameters. The result of this is that fringe order information no longer must be stored along with the encoded 3D data within the output 2D image. This freedom allows our method the ability to precisely represent floating point 3D range geometry within two entire color channels while keeping the third color channel open for additional data storage. Further, the encoding within the two color channels is continuous in nature (i.e., no sharp intensity changes), which allows the proposed method to achieve extremely large compression ratios (i.e., smaller file sizes) via lossy JPEG encoding while maintaining very high reconstruction accuracies. For example, compression ratios of 3038:1 were achieved versus the STL format, with a RMS error of 0.47%, when the output image was compressed with JPEG 80%.

The proposed 3D range geometry encoding method can efficiently archive or transmit 3D range geometry data, which could be valuable for applications such as entertainment, security, and telecommunications. Further, given the method's ability to encode 3D range data within two color channels, a texture image can be stored in the third channel. This may be beneficial, for example, to the area of telemedicine: remote physicians could leverage both decoded 3D range geometry and 2D texture image to perform simultaneous physical measurements and visual assessments to make sound medical decisions.

Section 2 will describe the novel 3D range geometry encoding and decoding methods, specifically in how the geometric constraints of the capture system can be used to help decode geometry information stored within a 2D image. Section 3 will present various experimental results of the proposed encoding method, and Section 4 will summarize the paper.

2. PRINCIPLE

A. Phase Encoding for 3D Range Geometry Compression

A generic structured light scanner consists of one camera and one projector. The DFP technique is one of the structured light methods which uses a projector to project phase-shifted, sinusoidal fringe images onto a 3D scene. The camera will then capture the distorted fringe images projected upon the scene and can use these to compute distorted phase information. This phase information can then be used pixel-by-pixel to recover 3D coordinates if the DFP system is properly calibrated [30].

The concepts of phase shifting can also be used to encode 3D geometry into a 2D RGB image. However, as discussed in Section 1, the state-of-the-art methods require one of the three output color channels to store the fringe order information needed to properly decode the phase-shifted data. This paper presents a novel method for encoding that can recover the geometry without needing to store fringe order information. Given this, the proposed method can use two data channels to precisely encode data while having one channel free to store additional data.

The proposed method directly encodes distorted phase information as captured by a DFP system, Φ, into two color channels (e.g., red and green) of the output 2D image:

$$I_r(i,j) = 0.5 + 0.5 \times \sin[\Phi(i,j)/\mathrm{SF}], \tag{1}$$

$$I_g(i,j) = 0.5 + 0.5 \times \cos[\Phi(i,j)/\mathrm{SF}], \tag{2}$$

where (i,j) are image pixel indices and where SF is a scaling factor. This encoding is advantageous as it retains the precision of the phase map while remaining very straightforward to implement. Once the phase has been encoded into the 2D image, it can be further compressed using conventional methods, such as PNG or JPEG.

B. Phase Decoding and Unwrapping Using Geometric Constraints

To recover phase back from the 2D image, ϕ is computed from the encoded data stored in the two channels:

$$\phi(i,j) = \tan^{-1}\left[\frac{I_r(i,j) - 0.5}{I_g(i,j) - 0.5}\right]. \quad (3)$$

This recovered phase ϕ is bounded within the range $[-\pi, \pi)$. The original unwrapped phase, Φ, can be recovered if the 2π discontinuities within ϕ can be identified, ordered, and corrected. It is this fringe order information, denoted by K, which existing encoding methods carry along within an additional color channel. To save data and to avoid using a color channel to carry along the fringe order information (either directly or within some other encoding), the proposed method uses the geometric constraints of the DFP system to generate an artificial phase map, $\Phi_{\min}$. Then, for each pixel, $\Phi_{\min}$ can be referenced to determine the proper K value for that pixel. The following will describe the mathematical models governing the system and how they are used, as proposed by An *et al.* [31], to generate $\Phi_{\min}$.

The camera and projector within a structured light system can each be mathematically described using a pinhole model. Using this model, real-world coordinates, (x^w, y^w, z^w), can be projected onto the 2D plane, at the coordinate (u, v), using the equation

$$s[\,u \quad v \quad 1\,]^t = \mathbf{P}[\,x^w \quad y^w \quad z^w \quad 1\,]^t, \quad (4)$$

where s is a scaling factor and $\mathbf{P}$ is the projection matrix. This matrix can be described as

$$\mathbf{P} = \begin{bmatrix} f_u & \gamma & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}, \quad (5)$$

where f_u and f_v are the focal lengths along the u and v directions, respectively; γ is the skew factor of the two axes; r_{ij} and t_i are the rotation and translation parameters; and (u_0, v_0) is the principle point. This projection matrix is often simplified into a single 3×4 matrix:

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}. \quad (6)$$

If the camera and projector of a DFP structured light system are properly calibrated, we know their respective projection matrices, $\mathbf{P^c}$ and $\mathbf{P^p}$. These matrices can be used to obtain two sets of equations, one for the camera (denoted superscript c) and one of the projector (denoted superscript p) describing the DFP system:

$$s^c[\,u^c \quad v^c \quad 1\,]^t = \mathbf{P}^c[\,x^w \quad y^w \quad z^w \quad 1\,]^t, \quad (7)$$

$$s^p[\,u^p \quad v^p \quad 1\,]^t = \mathbf{P}^p[\,x^w \quad y^w \quad z^w \quad 1\,]^t. \quad (8)$$

Equations (7) and (8) provide six equations yet there are seven unknowns: s^c, s^p, x^w, y^w, z^w, u^p, and v^p. To solve for the unknowns, one more equation is needed; typically, the linear relationship between some known absolute phase value, Φ, and a projector line is used to resolve this by providing an additional equation to solve for u^p or v^p (depending on the direction of Φ). At this point, the unknowns can be solved for, and a 3D coordinate for each camera pixel can be derived.

Similarly, consider if the absolute phase value is unknown yet the depth value z^w is known for a pixel. For a given z^w then, an artificial phase value can be determined. Further, if $z^w = z_{\min}$, the artificial phase map is a minimum phase map, denoted $\Phi_{\min}$. This map can formally be defined by a function taking inputs $z_{\min}$, the minimum z value; T, the fringe width on the projector used to capture the original data; and the respective projection matrices for the camera and projector, $\mathbf{P^c}$ and $\mathbf{P^p}$. Based on the fringe width T used by the DFP system, the minimum phase map may have a limited working depth range [31]. To ensure that $\Phi_{\min}$ can be used to properly unwrap the decoded ϕ, T_s is used to derive $\Phi_{\min}$, and it is defined as $T \times \mathrm{SF}$; thus,

$$\Phi_{\min}(u^c, v^c) = f(z_{\min}, T_s, \mathbf{P^c}, \mathbf{P^p}), \quad (9)$$

is of a function of $z_{\min}$, T_s, $\mathbf{P^c}$ and $\mathbf{P^p}$.

To actually determine $\Phi_{\min}$, x^w and y^w are first computed for each camera pixel (u^c, v^c) via

$$[\,x^w \quad y^w\,]^t = A^{-1}b, \quad (10)$$

where

$$A = \begin{bmatrix} p^c_{31}u^c - p^c_{11} & p^c_{32}u^c - p^c_{12} \\ p^c_{31}v^c - p^c_{21} & p^c_{32}v^c - p^c_{22} \end{bmatrix}, \quad (11)$$

$$b = \begin{bmatrix} p^c_{14} - p^c_{34}u^c - (p^c_{33}u^c - p^c_{13})z_{\min} \\ p^c_{24} - p^c_{34}v^c - (p^c_{33}v^c - p^c_{23})z_{\min} \end{bmatrix}. \quad (12)$$

Knowing x^w and y^w, (u^p, v^p) can be found for each camera pixel, similar to Eq. (8):

$$s^p[\,u^p \quad v^p \quad 1\,]^t = \mathbf{P}^p[\,x^w \quad y^w \quad z_{\min} \quad 1\,]^t. \quad (13)$$

Finally, the artificial phase value, $\Phi_{\min}$ can be determined via

$$\Phi_{\min}(u^c, v^c) = u^p \times 2\pi / T_s. \quad (14)$$

This specific equation will provide phase assuming the fringe patterns are projected along the v^p direction; to obtain phase along the other direction, the u^p and v^p values can simply be swapped.

Once the artificial phase map has been derived, it can be used to determine the fringe order information, K, as

$$K(i,j) = \mathrm{Ceil}\left[\frac{\Phi_{\min}(i,j) - \phi(i,j)}{2\pi}\right]. \quad (15)$$

The fringe order information is then used to unwrap ϕ in order to recover the originally encoded phase information, Φ, via

$$\Phi(i,j) = [\phi(i,j) + 2\pi \times K(i,j)] \times \mathrm{SF}. \quad (16)$$

Now that the originally encoded phase, Φ, has been decoded and recovered, (x^w, y^w, z^w) coordinates can be reconstructed with Eqs. (7) and (8) as described above.

3. EXPERIMENTS

To test the proposed method, several different objects were captured with a DFP system. Comparisons made were between the 3D geometry reconstructed from the original unwrapped phase versus the 3D geometry reconstructed from the decoded, recovered unwrapped phase. The hardware system included a digital light processing projector (Texas Instruments LightCrafter 4500) and a camera (PointGrey Flea3 FL3-U3-13Y3M-C) with an 8 mm lens (Computar M0814-MP2). The resolutions

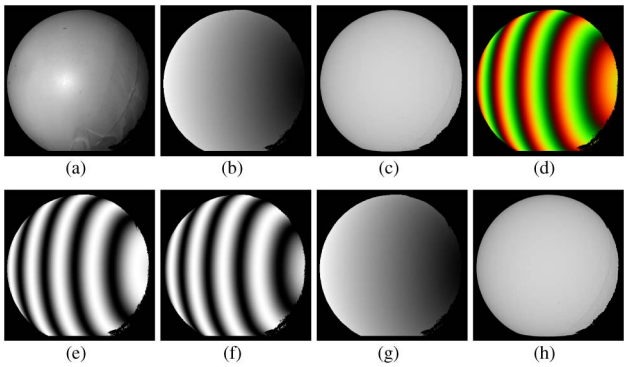

Fig. 1. Experimental results of capturing, encoding, and decoding a 4 in. (101.60 mm) diameter sphere. (a) A 2D texture image of the sphere; (b) original absolute phase of the sphere; (c) original 3D geometry reconstructed from (b); (d) encoded phase stored in a lossless PNG image via the proposed method, cropped for visualization from its original 480 × 640 resolution; (e) the red channel of (d); (f) the green channel of (d); (g) the decoded absolute phase from (d); (h) recovered 3D geometry reconstructed from the decoded (g).

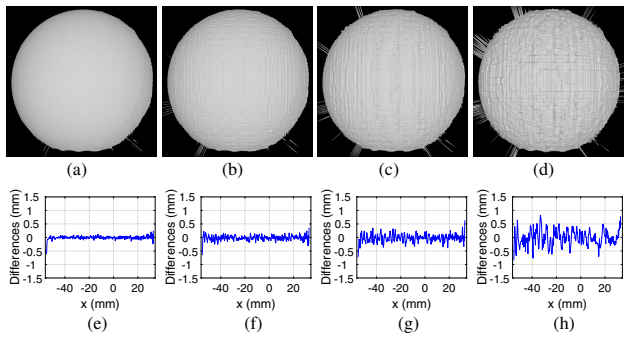

Fig. 2. Results of the sphere's phase being encoded into two 2D color channels and saved with different JPEG qualities (using MATLAB 2014b). (a)–(d) 3D reconstructed results using the decoded phase from JPEG qualities 100%, 80%, 60%, and 20%, respectively; (e)–(h) Difference in z^w between the original sphere and the recovered sphere for a cross section. RMS errors for (e)–(h) are 0.17 mm (0.35%), 0.23 mm (0.47%), 0.31 mm (0.61%), and 0.43 mm (0.85%), respectively, after removing boundary outliers with a simple threshold.

of the camera and projector were 480 × 640 and 912 × 1140, respectively. For all mentioned experiments, the fringe width used was $T = 36$ pixels. The system was calibrated following the method proposed by Li *et al.* [30], and only 553 bytes were required to store the resulting calibration parameters.

First, a matte white spherical object with a 4 in. (101.60 mm) diameter was captured by a DFP system and had its phase encoded into a 480 × 640 lossless PNG image using the proposed method. From this 2D image, the phase was decoded and used to reconstructed 3D coordinates. In this first experiment, no additional texture information was stored in the output 2D image: the phase was encoded into the red and green channels and the blue channel remained empty. Figure 1 illustrates this entire process: Fig. 1(a) shows a texture image of the sphere, Fig. 1(b) shows the original absolute phase, and Fig. 1(c) shows the sphere's 3D geometry. The phase from Fig. 1(b) was then encoded into a PNG image (cropped for visualization), shown in Fig. 1(d). Figures 1(e) and 1(f), respectively, show the red and green color channels of the PNG image, which contain encoded phase information. These channels are then decoded to recover the absolute phase data, displayed in Fig. 1(g), which is used to recover the 3D sphere, shown in Fig. 1(h).

Figure 2 shows the reconstructed results when the output image shown in Fig. 1(d) was stored with different lossy image qualities (JPEG 100%, 80%, 60%, and 20%) using MATLAB 2014b. One may notice that the reconstructed 3D geometry quality is fairly high if the JPEG 100% was used, as shown in Fig. 2(e), and the associated phase RMS error is small (0.17 mm or 0.35%). Even if JPEG 20% was used the quality of the reconstructed 3D geometry is still reasonably good, and the error is still pretty small (0.85%).

It is important to note that these results were obtained without a kernel-based, post-processing filter or error compensation framework. The only post-processing performed was a simple threshold to remove significant boundary outliers.

The original 3D capture of the 4 in. (101.60 mm) diameter sphere required 65.0 MB, 9.0 MB, and 8.4 MB to store in the common mesh formats STL, OBJ, and PLY, respectively, in their ASCII formats. When storing the encoded sphere into a PNG image, the proposed method was capable of approximately a 688:1 compression ratio, with an RMS error of 0.02 mm (0.033%), versus the original geometry stored in the STL format. To obtain higher compression ratios, lossy JPEG was used to store the output image. For example, when JPEG 80% was used, a 3038.3:1 compression ratio was achieved with an RMS error of 0.23 mm (0.466%). Even when saving out the encoded sphere at the low image quality of JPEG 20%, the RMS error was only 0.42 mm (0.854%) and achieved a 6241.9:1 compression ratio versus STL. Table 1 shows the overall compression ratios when Fig. 1(d) was encoded into different image qualities and compared against the common mesh formats.

Another experiment was performed to evaluate the proposed method's ability to properly encode phase of multiple, more complex, geometries. In this experiment, a scene consisting of a cat sculpture and a dog sculpture was captured and had its phase encoded into a PNG image. This PNG image was then decoded to recover the phase from which 3D coordinates were reconstructed. Figure 3 demonstrates that the proposed method was indeed able to properly encode and decode phase containing multiple, complex, geometries. Figure 3(a) shows the original 3D geometry and Fig. 3(b) shows the 3D geometry

Table 1. Compression Ratios of the Encoded Sphere Using PNG and Different JPEG Levels Versus Common 3D Mesh Formats

	PNG	JPG100	JPG80	JPG60	JPG40	JPG20
STL	688.0: 1	856.5: 1	3038.3: 1	3983.6: 1	4795.6: 1	6241.9: 1
OBJ	99.9: 1	124.3: 1	441.0: 1	578.2: 1	696.0: 1	905.9: 1
PLY	88.6: 1	110.2: 1	391.1: 1	512.7: 1	617.3: 1	803.4: 1

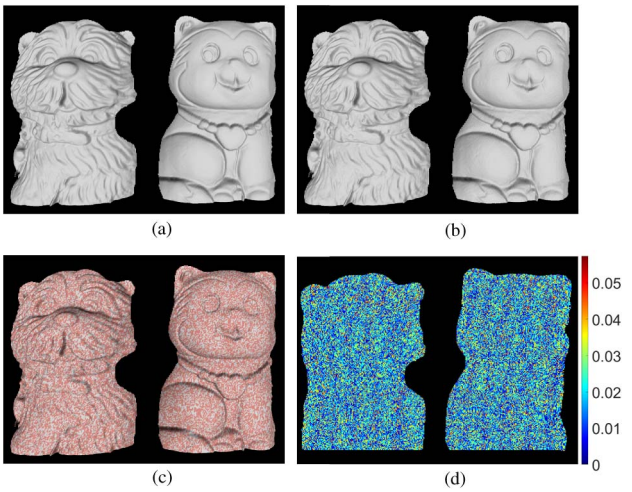

Fig. 3. Visual demonstration of reconstruction results when the scene contains multiple, complex geometries. (a) Original 3D geometry; (b) 3D geometry recovered from decoded phase in the red and green channels of a PNG image; (c) overlay of the original and reconstructed 3D geometries (gray color represents recovered geometry, red represents the original geometry); (d) error map, in mm, between the original and recovered geometry (RMS error of 0.02 mm).

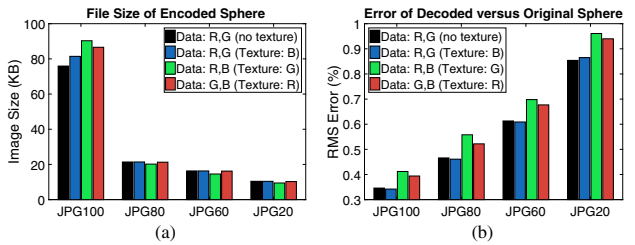

Fig. 4. Storing encoded phase data and texture in different channels for various levels of JPEG compression. (a) Comparison of JPEG file sizes when the texture is not used or stored in the blue, green, or red channels; (b) comparison of reconstructed 3D geometry error (versus the original sphere) when phase data is encoded into various color channels, potentially along with a texture image in the remaining channel.

reconstructed from a PNG image. Figure 3(c) is a rendered image that overlaps the two geometries together, showing that the difference between them is very small. Figure 3(d) provides an error map between the two reconstructions. The RMS error for this geometry reconstructed from a PNG image was 0.02 mm.

In the previous experiments, the encoded phase data was stored in the red and green channels; this would allow the blue channel to store texture information when desired. If the 2D RGB image is stored with a lossless compression method (e.g., PNG), it does not matter which respective channels the data and texture reside within. However, due to how most JPEG encoders typically perform their image encoding, different color channels end up being encoded at varying levels of fidelity.

During JPEG encoding, an image's RGB values are transformed into $Y'C_BC_R$ color values, where Y' represents the luma component and where C_B and C_R represent the blue and red chroma components, respectively. The human visual system (HVS) typically is more sensitive to changes in luminance as opposed to changes in color [32]. Given this, JPEG encoders maintain high fidelity in the Y' component and usually downsample the C_B and C_R components in aims to reduce the file size while minimizing impact on the perceptual quality of the reconstructed image. In the RGB to $Y'C_BC_R$ transformation used by JPEG, the highly preserved Y' component is primarily influenced by the green channel, followed by the red and then blue channel values [33]. This is done to further mimic the HVS as humans are typically more sensitive to green, red, and then blue light, respectively.

Another experiment was conducted to evaluate the impact this color JPEG encoding has on the proposed phase encoding method. In this experiment, the same 4 in. (101.60 mm) sphere from the previous experiments was used. The sphere's phase was encoded into the different channel arrangements (red and green, red and blue, green and blue) while storing the sphere's grayscale texture image in the remaining channel (blue, green, red, respectively). The RGB images were then stored using MATLAB's JPEG encoder, which uses chroma subsampling, at various compression levels. From the compressed JPEG images, phase and geometries were reconstructed and compared against the original 3D capture of the sphere.

Figure 4(a) compares the file sizes across the various color channel arrangements. When JPEG 100% was used to store the 480×640 2D image, file sizes of 76 KB were obtained when encoding the phase into the red and green channels (leaving the blue channel empty); 81 KB when encoding into the red and green channels (storing texture in the blue channel); 90 KB when encoding into the red and blue channels (storing texture in the green channel); and 87 KB when encoding into the green and blue channels (storing texture in the red channel). Overall, the selection of which color channels to store data and textures does not drastically affect the resulting file size of the 2D image, especially when using higher levels of JPEG compression, as the file sizes all become near equivalent.

Figure 4(b) compares reconstructed 3D geometry accuracies, versus the original sphere, when the different color channel arrangements are used. When JPEG 100% was used, the reconstruction errors were 0.346% when encoding phase data into the red and green channels (leaving the blue channel empty); 0.342% when encoding into the red and green channels (texture in the blue channel); 0.412% when encoding into the red and blue channels (texture in the green channel); and 0.394% when encoding into the green and blue channels (texture in the red channel). The overall trend was that the lowest errors could be achieved by encoding phase data into the red and green channels, storing texture in the blue channel, if desired.

A final experiment used the proposed phase encoding method to compress a dynamic sequence of 3D data frames along with their associated color textures. For this experiment, a color camera (PointGrey Grasshopper3 GS3-U3-23S6C) was used. Each captured frame's phase was encoded into the red and green channels of an output 2D image with the proposed method. Each frame's associated texture—before color

Fig. 5. Reconstructions of 3D data from a dynamic sequence (associated with Visualization 1). Each column, from left to right respectively, represents reconstructions from various levels of compression used to store the output 2D image: PNG, JPEG 100%, JPEG 95%, JPEG 90%, and JPEG 85%. First row: reconstructed 3D geometry from the compressed images. Second row: reconstructed 3D geometry with small median and Gaussian filters applied. Third row: filtered reconstructed 3D geometry with color texture mapping applied.

demosaicing—was placed in the blue channel of its respective output image. Each output RGB image was then compressed using various levels of image compression: PNG, JPEG 100%, JPEG 95%, JPEG 90%, and JPEG 85%. It should be noted that chroma subsampling was not used for the JPEG encodings in this experiment.

Figure 5 shows reconstructions from the encoded images for one of the dynamic frames stored using various levels of 2D image compression: PNG, JPEG 100%, JPEG 95%, JPEG 90%, and JPEG 85%, from left to right. Visualization 1 shows several seconds of the decoded dynamic sequence. The first row shows the reconstructed 3D geometry without any post-processing or filtering. The second row shows the same reconstructed 3D with small median and Gaussian filters applied to remove noise around the edges and to reduce blocking artifacts imposed by JPEG. The third row shows the filtered reconstructed 3D geometry with color texture mapping applied. Color texture maps were obtained by demosaicing the texture stored within the blue channel of the encoded output images.

It is important to know that there is a trade-off between accuracy and depth range. As previously mentioned in Section 2, the minimum phase unwrapping method has a limited working depth range [31] that was ensured by using a scaling factor, SF, in our proposed method. Increasing the SF extends the depth range but reduces its accuracy. Conversely, decreasing the scaling factor increases the accuracy but reduces the effective depth range of the encoding. Therefore, in practice, the selection of SF should be tailored for a given application where the depth range can be pre-defined.

4. SUMMARY

This paper presented a novel method for the compression of 3D range geometry into a regular 24-bit 2D RGB image which utilized geometric constraints of the 3D scanning device itself to reduce the amount of data that need be stored. The proposed method used two color channels to precisely represent 3D geometry information while leaving one channel free to store additional attributes about the data (such as a texture image). Our experiments demonstrated the overall efficiency and robustness of the proposed method. When PNG was used to store the encoded output image, compression ratios of approximately 688:1 were achieved versus the STL format with an RMS error of only 0.033%. Additional experiments highlighted the proposed method's resiliency to lossy JPEG image compression. For example, compression ratios of 3038:1 were achieved versus STL with an RMS error of 0.47% when the

encoded image was compressed with JPEG 80%. Lastly, it was shown that the proposed method could reconstruct complex 3D geometry and color texture information from a single, JPEG compressed 2D RGB image, which may be useful within applications such as communications and telemedicine.

REFERENCES

1. S. Zhang, "Recent progresses on real-time 3D shape measurement using digital fringe projection techniques," Opt. Laser Eng. **48**, 149–158 (2010).
2. M. Deering, "Geometry compression," in *22nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, New York, New York (ACM, 1995), pp. 13–20.
3. M. M. Chow, "Optimized geometry compression for real-time rendering," in *8th Conference on Visualization*, Los Alamitos, California (IEEE Computer Society, 1997), pp. 347–354.
4. C. L. Bajaj, V. Pascucci, and G. Zhuang, "Single resolution compression of arbitrary triangular meshes with properties," Comput. Geom. **14**, 167–186 (1999).
5. G. Taubin and J. Rossignac, "Geometric compression through topological surgery," ACM Trans. Graph. **17**, 84–115 (1998).
6. C. Touma and C. Gotsman, "Triangle mesh compression," in *Proceedings of Graphics Interface* (1998), pp. 26–34.
7. P. Alliez and M. Desbrun, "Valence-driven connectivity encoding for 3D meshes," Comput. Graph. Forum **20**, 480–489 (2001).
8. S. Gumhold and W. Straßer, "Real time compression of triangle mesh connectivity," in *25th Annual Conference on Computer Graphics and Interactive Techniques SIGGRAPH*, New York, New York (ACM, 1998), pp. 133–140.
9. J. Rossignac, "Edgebreaker: connectivity compression for triangle meshes," IEEE Trans. Vis. Comput. Graphics **5**, 47–61 (1999).
10. J. Peng, C.-S. Kim, and C.-C. J. Kuo, "Technologies for 3D mesh compression: a survey," J. Visual Commun. Image Represent. **16**, 688–733 (2005).
11. A. Maglo, G. Lavoué, F. Dupont, and C. Hudelot, "3D mesh compression: Survey, comparisons, and emerging trends," ACM Comput. Surv. **47**, 1–41 (2015).
12. B. Kronrod and C. Gotsman, "Optimized compression of triangle mesh geometry using prediction trees," in *1st International Symposium on 3D Data Processing Visualization and Transmission* (2002), pp. 602–608.
13. E. Darakis and J. J. Soraghan, "Compression of interference patterns with application to phase-shifting digital holography," Appl. Opt. **45**, 2437–2443 (2006).
14. E. Darakis and J. J. Soraghan, "Reconstruction domain compression of phase-shifting digital holograms," Appl. Opt. **46**, 351–356 (2007).
15. M. Burrows and D. J. Wheeler, "A block-sorting lossless data compression algorithm," Tech. Rep. (Digital Systems Research Center, 1994).
16. A. Alfalou and C. Brosseau, "Optical image compression and encryption methods," Adv. Opt. Photon. **1**, 589–636 (2009).
17. F. Dufaux, Y. Xing, B. Pesquet-Popescu, and P. Schelkens, "Compression of digital holographic data: an overview," Proc. SPIE **9599**, 95990I (2015).
18. Y. Xing, B. Pesquet-Popescu, and F. Dufaux, "Compression of computer generated hologram based on phase-shifting algorithm," in *European Workshop on Visual Information Processing (EUVIP)* (2013), pp. 172–177.
19. Y. Xing, B. Pesquet-Popescu, and F. Dufaux, "Compression of computer generated phase-shifting hologram sequence using AVC and HEVC," Proc. SPIE **8856**, 88561M (2013).
20. T. Nishitsuji, T. Shimobaba, T. Kakue, and T. Ito, "Fast calculation techniques for computer-generated holograms," in *IEEE 14th International Conference on Industrial Informatics (INDIN)* (2016), pp. 550–555.
21. T. Shimobaba, T. Ito, N. Masuda, Y. Ichihashi, and N. Takada, "Fast calculation of computer-generated-hologram on AMD HD5000 series GPU and OpenCL," Opt. Express **18**, 9955–9960 (2010).
22. 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).
23. J. Weng, T. Shimobaba, N. Okada, H. Nakayama, M. Oikawa, N. Masuda, and T. Ito, "Generation of real-time large computer generated hologram using wavefront recording method," Opt. Express **20**, 4018–4023 (2012).
24. N. Karpinsky and S. Zhang, "Composite phase-shifting algorithm for three-dimensional shape compression," Opt. Eng. **49**, 063604 (2010).
25. S. Zhang, "Three-dimensional range data compression using computer graphics rendering pipeline," Appl. Opt. **51**, 4058–4064 (2012).
26. T. Bell and S. Zhang, "Multiwavelength depth encoding method for 3D range geometry compression," Appl. Opt. **54**, 10684–10691 (2015).
27. Z. Hou, X. Su, and Q. Zhang, "Virtual structured-light coding for three-dimensional shape data compression," Opt. Laser Eng. **50**, 844–849 (2012).
28. Y. Wang, L. Zhang, S. Yang, and F. Ji, "Two-channel high-accuracy holoimage technique for three-dimensional data compression," Opt. Laser Eng. **85**, 48–52 (2016).
29. N. Karpinsky, Y. Wang, and S. Zhang, "Three-bit representation of three-dimensional range data," Appl. Opt. **52**, 2286–2293 (2013).
30. B. Li, N. Karpinsky, and S. Zhang, "Novel calibration method for structured-light system with an out-of-focus projector," Appl. Opt. **53**, 3415–3426 (2014).
31. Y. An, J.-S. Hyun, and S. Zhang, "Pixel-wise absolute phase unwrapping using geometric constraints of structured light system," Opt. Express **24**, 18445–18459 (2016).
32. R. Hunt, *The Reproduction of Colour*, 6th ed. (Wiley, 2005).
33. Telecommunication Standardization Sector of ITU (ITU-T), Still-image compression–JPEG-1 Extensions, "Information technology–digital compression and coding of continuous-tone still images: JPEG file interchange format (JFIF)," ITU-T Recommendation T.871.