Three dimensional range geometry and texture data compression with space-filling curves

XIA CHEN AND SONG ZHANG*  
School of Mechanical Engineering, Purdue University, West Lafayette, Indiana 47907, USA  
*szhang15@purdue.edu

Abstract: This paper presents a novel method to effectively store three-dimensional (3D) data and 2D texture data into a regular 24-bit image. The proposed method uses the Hilbert space-filling curve to map the normalized unwrapped phase map to two 8-bit color channels, and saves the third color channel for 2D texture storage. By further leveraging existing 2D image and video compression techniques, the proposed method can achieve high compression ratios while effectively preserving data quality. Since the encoding and decoding processes can be applied to most of the current 2D media platforms, this proposed compression method can make 3D data storage and transmission available for many electrical devices without requiring special hardware changes. Experiments demonstrate that if a lossless 2D image/video format is used, both original 3D geometry and 2D color texture can be accurately recovered; if lossy image/video compression is used, only black-and-white or grayscale texture can be properly recovered, but much higher compression ratios (e.g., 1543:1 against the ASCII OBJ format) are achieved with slight loss of 3D geometry quality.

© 2017 Optical Society of America

OCIS codes: (100.6890) Three-dimensional image processing; (120.0120) Instrumentation, measurement, and metrology; (100.2960) Image analysis; (100.5070) Phase retrieval.

References and links
1. Introduction

With the rapid development of three-dimensional (3D) measurement technique, high resolution and high quality 3D range data can be acquired at real time [1]. The advances on 3D range imaging have been accelerating in the past few years propelled by the availability of commercial 3D sensors (e.g., Microsoft Kinect, Intel RealSense). Yet, there is one fundamental issue that has not been fully addressed: how can one effectively store and deliver such enormously large 3D data?

Conventional 3D geometry representation methods including OBJ, PLY, and STL can represent arbitrary 3D geometry and texture data. They typically store data as a sequence of vertices, connectivity between vertices, and often \((u, v)\) coordinate or direct color information to colorize each vertex. These formats have been extensively used to store a single 3D geometry data, yet the file size is enormous (typically at least one order of magnitude larger than 2D counter parts) [2]. Therefore, it is challenging for these formats to represent high-resolution 3D geometry videos for efficient storage; thus it is extremely difficult, if at all possible, to use these formats for 3D video communication across the standard wireless networks that are currently available.

While 3D image data compression is still in its infancy, compression techniques for 2D images, however, are quite mature. As a result, researchers have developed a variety of methods to store 3D data as standard 2D images such that 2D image compression techniques can be leveraged. In the field of holography, 3D images can be reconstructed if both the amplitude and the phase information of the wavefront reflected by the object are recorded [3]. Inspired by a common 2D image compression standard, JPEG2000, Alkholidi et al. [4] introduced a compression method of holograms by implementing the 2D wavelet transform optically to simulate the first few stages of JPEG2000 algorithms. The remaining step of compression (i.e., the entropy coding) is processed digitally. The testing results show that the 3D holograms that are recovered from the 2D images compressed by this method preserve high quality. As discussed by Alfalou and Brosseau [3] and Dufaux et al. [5], the holography-based compression methods can successfully reduce data size for data encryption and secure transmission. However, the compression methods based on digital holograms have three major limitations: 1) the achieved compression ratio is not high (typically less than 30:1) for high-quality 3D representation because the lossy 2D image compression typically does not naturally work effectively for image with random speckle noise [3]; 2) the recovered data has speckle noise even though they can be reduced by changing the compression domain [6]; and 3) the computational cost is typically very high especially when the computer generated hologram is used, albeit the speed can be significantly improved by an advanced graphics processing unit (GPU) [7–9].
Another state-of-the-art conversion method from 3D data to 2D image is to build a virtual structured light system, and project virtual fringe patterns onto the 3D object. Since the virtual settings can be precisely determined, three 8-bit channels of a standard 2D color image are sufficient to represent 3D range data. Karpinsky and Zhang [2] developed a method to encode two channels with sine and cosine functions of the phase map, and a third channel with fringe order information for pixel-by-pixel phase unwrapping. Later Karpinsky and Zhang [10,11] slightly modified this method by encoding the fringe order information in the third channel with smooth cosine function to enable 3D range video compression. Wang et al. [12] developed a two-channel method that utilized one channel to record the cosine function of the phase and a second channel to store fringe order information, and they demonstrated the success of such an approach with lossless formats. However, due to the sharp changes of the second channel, it is difficult for this approach to store the encoded 2D images in a lossy format. Hou et al. [13] developed a method to directly encode the wrapped phase into one channel and the fringe order into the second channel. Similarly, this method can only work for lossless 2D image representation approach because the $2\pi$ discontinuities of the wrapped phase map can cause problems if the information is slightly lost. All these virtual structured light based 3D-to-2D conversion methods can work well, but require the creation of another virtual structured light system that introduces additional problems associated with resampling and triangulation (e.g., the points that cannot be “seen” by either the virtual projector or the virtual camera). Furthermore, the spatial resolution of these methods is typically limited by the resolution that a single video card can support, and thus the recovered 3D data resolution may be lower than that of the original data.

To alleviate some of the aforementioned problems, Zhang [14] developed a direct depth encoding method that samples the depth map of captured 3D geometry by leveraging the OpenGL rendering pipeline; the sampled data is further encoded as regular 2D images using sine and cosine functions. This method allows the use of an arbitrary high resolution image for 3D data representation, yet still requires a resampling process before encoding. Ou and Zhang [15] developed a native compression method that encodes scaling factor ($s$) map of the structured light with sine and cosine functions, achieving a resolution as high as the camera’s, without resampling. However, this approach requires reconstructing 3D geometry point by point twice: the first time is to obtain $s$ map before encoding, and the second time is to reconstruct 3D geometry from the decompressed $s$ map. Overall, one common issue of all aforementioned compression methods is that when the image is stored in lossy compression formats, substantial filtering is required to reduce phase unwrapping artifacts during the decoding process.

Recognizing this problem, Bell and Zhang [16] proposed a two-frequency depth encoding method that stores two channels with sine and cosine functions of high-frequency phase map, and a third channel with normalized low-frequency phase map for phase unwrapping. Since all three channels are smooth, this method requires little to no filtering even stored as lossy formats. However, similar to the direct depth encoding method developed by Zhang [14], this method also requires resampling before encoding.

All the 3D geometry compression methods mentioned so far ignore the image texture (i.e., 2D photograph). To address such a problem, Karpinsky et al. [17] applied a dithering technique to the encoded image such that only three bits are required to store 3D range data, and it saves the rest bits for texture storage. This method demonstrated its success by using lossless image formats (e.g., PNG) with a reasonably high compression ratio. However, the nature of dithering is not suitable for the use of any lossy 2D image/video formats (e.g., JPEG, H.264) for further compression.

This paper presents a novel method for representing both high-quality 3D range geometry and 2D texture image within a regular 24-bit 2D color image. The proposed method uses the Hilbert space-filling curve [18] to map the normalized unwrapped phase map to two 8-bit color channels, and saves the third color channel for 2D texture storage. Experiments demonstrated that
Section 5 summarizes this paper. The rest of this paper is structured as follows: Section 2 explains the principles of the proposed method. Section 3 shows some experimental data to verify the performance of the proposed method. Section 4 discusses advantages and limitations of the proposed method, and finally, Section 5 summarizes this paper.

2. Principle

This section will introduce the pipeline and principles of the proposed compression method. We will discuss the structured light system and phase-based 3D shape measurement first, and later focus on how to encode the phase information into one or multiple channels.

2.1. Phase-based 3D absolute shape measurement

Phase-based 3D absolute shape measurement is one of the most popular 3D measurement methods because of its robustness and accuracy [19]. Phase-based method applies the structured light techniques that usually contain one camera and one projector [20]. The pinhole camera model [21], which describes the geometric relationship between camera pixel coordinates \((u, v)\) and 3D world coordinates \((x, y, z)\), can be mathematically represented as

\[
\begin{bmatrix}
    u \\
    v \\
    1
\end{bmatrix}
= \begin{bmatrix}
    f_u & \gamma & u_0 \\
    0 & f_v & v_0 \\
    0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    r_{11} & r_{12} & r_{13} & t_1 \\
    r_{21} & r_{22} & r_{23} & t_2 \\
    r_{31} & r_{32} & r_{33} & t_3
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    z \\
    1
\end{bmatrix}
\]

where \(r_{ij}\) and \(t_i\) are the rotational and translational transform parameters from the 3D world coordinate to the camera coordinate relatively; \(f_u\) and \(f_v\) are the focal lengths along \(u\) and \(v\) direction; \(\gamma\) is the skew factor of the camera’s axes; \((u_0, v_0)\) is called principle point where the optical axis intersects the camera’s imaging plane. This equation can be further simplified by defining a projection matrix \(P\):

\[
P = \begin{bmatrix}
    f_u & \gamma & u_0 \\
    0 & f_v & v_0 \\
    0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    r_{11} & r_{12} & r_{13} & t_1 \\
    r_{21} & r_{22} & r_{23} & t_2 \\
    r_{31} & r_{32} & r_{33} & t_3
\end{bmatrix}
\]

Since a projector is physically identical to a camera except that it projects, instead of capturing, images. Thus the camera and the projector can be modeled by rewriting Eq. (1) as:

\[
s^c\begin{bmatrix}
    u^c \\
    v^c \\
    1
\end{bmatrix}^T = P^c\begin{bmatrix}
    x \\
    y \\
    z \\
    1
\end{bmatrix}^T
\]

\[
s^p\begin{bmatrix}
    u^p \\
    v^p \\
    1
\end{bmatrix}^T = P^p\begin{bmatrix}
    x \\
    y \\
    z \\
    1
\end{bmatrix}^T
\]

Where superscript \( ^c \) denotes camera, and \( ^p \) denotes projector. From Eq. (2)–(4), we have 6 equations with 7 unknowns \((x, y, z, s^c, s^p, u^p, v^p)\) once the projection matrices are pre-calibrated or known. Thus one additional equation is necessary to solve for \((x, y, z)\) coordinates corresponding to a camera pixel for 3D shape measurement [22]. In a phase-based 3D measurement method, the absolute phase \(\Phi(u^c, v^c)\) is often used to provide one additional constraint equation (i.e., a mapping from one point on the camera plane to one line on the projector plane):

\[
\Phi(u^c, v^c) = u^p_{u^c, v^c}
\]
2.2. Phase normalization based on minimum and maximum phase

As an object to be measured is always within a certain depth range from the camera, we can create a virtual plane at the smallest depth ($z$) value (i.e., $z = z_{\text{min}}$) to generate a corresponding virtual phase map, denoted as $\Phi_{\text{min}}(u^c, v^c)$ [23]. From Eqs. (3)–(4), if $z = z_{\text{min}}$ for each camera pixel $(u^c, v^c)$ is known, the corresponding coordinate $(u^p, v^p)$ on the projector plane is uniquely determined. Thus we can calculate the corresponding minimum phase value $\Phi_{\text{min}}(u^c, v^c)$ that is a function of depth plane $z_{\text{min}}$ and the projection matrices $P^c$ and $P^p$:

$$
\Phi_{\text{min}}(u^c, v^c) = f(z_{\text{min}}; P^c, P^p)
$$

Similarly, we can define the maximum phase map for the object to be measured as at a virtual plane where $z = z_{\text{max}}$:

$$
\Phi_{\text{max}}(u^c, v^c) = f(z_{\text{max}}; P^c, P^p)
$$

We then normalize the unwrapped phase map $\Phi(u^c, v^c)$ by mapping each phase value into a real number between within the range of $[0, 1]$:

$$
\Phi_n(u^c, v^c) = \frac{\Phi(u^c, v^c) - \Phi_{\text{min}}(u^c, v^c)}{\Phi_{\text{max}}(u^c, v^c) - \Phi_{\text{min}}(u^c, v^c)}
$$

This normalized phase map can represent the original phase map for 3D reconstruction if it is stored properly along with the calibration data of a structured light system, $z_{\text{min}}$, and $z_{\text{max}}$.

2.3. Proposed 3D geometry and texture encoding

As what has been discussed before, we need to find a way to effectively encode the normalized phase map, $\Phi_n(u^c, v^c)$, for 3D data compression, and preferably as a 2D image. A standard 2D color image usually has three color channels (R, G, B), each represented by an unsigned 8-bit integer (0–255) for a single pixel. Meanwhile, to store the normalized phase in $k$-bit, a simple conversion can be used to map the real number between 0 and 1 to an unsigned $k$-bit integer linearly:

$$
\Phi_{n,k\text{-bit}}(u^c, v^c) = \text{Round} \left[ \Phi_n(u^c, v^c) \times 2^k \right],
$$

where Round($x$) is the operator to obtain closely integer number for $x$.

Thus, if we encode the normalized phase in a single channel (e.g., the red channel), 8 bits will be used. If we encode the data in two channels (e.g., the red and green channels), it will be represented by a total of 16 bits. Using more bits naturally results in higher resolution; Table 1 provides an example of achievable depth resolution for an object with a depth range $(z_{\text{min}} - z_{\text{max}})$ of 1,000 mm or 1 meter. This table indicates that if a scanner’s depth resolution is lower than 0.015 mm (or 15 $\mu$m), using more than 16 bits is not necessary. Therefore, in this paper, we mainly use 16 bits for our 3D geometry storage since a typical structured light scanner with 1 meter depth sensing range cannot achieve a resolution higher than 0.015 mm. It should be noted that it is possible to encode the phase map into three channels (24 bits), but it will not be discussed in this paper since the resolution is excessively high for practical use, although such a method is still valuable for 3D scanners with higher depth resolutions.

<table>
<thead>
<tr>
<th>No. of 8-bit channels</th>
<th>No. of bits</th>
<th>Resolution of depth (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$n = 8$</td>
<td>3.90</td>
</tr>
<tr>
<td>2</td>
<td>$n = 16$</td>
<td>0.015</td>
</tr>
<tr>
<td>3</td>
<td>$n = 24$</td>
<td>0.000060</td>
</tr>
</tbody>
</table>

Table 1. Resolution of depth when using different numbers of channels of a 24-bit 2D image, when the depth range is 1,000 mm or 1 meter.
When only one channel is used to store the normalized phase map, the encoding is straightforward: to directly store $\Phi_{n,8\text{-bit}}$ for each point (i.e., each pixel on the encoded 2D image). However, storing the data in multiple channels requires a mapping from a one-dimensional domain ($\Phi_{n,k\text{-bit}}$) to a multi-dimensional domain (i.e., $(\mathbf{R}, \mathbf{G}, \ldots)$ in the case of color channels). Taking 2-channel encoding as an example, if the red and the green channels are used, the following one-to-one mapping needs to be established:

$$\Phi_{n,16\text{-bit}} \mapsto (R, G)$$

(10)

where $\Phi_{n,16\text{-bit}}$ is an unsigned 16-bit integer ranging 0–65535 and $\mathbf{R}, \mathbf{G}$ are both unsigned 8-bit integers ranging 0–255.

Space-filling curves (SFCs) have been extensively adopted to create a mapping between a one-dimensional domain and a multi-dimensional domain [24]. Essentially, a SFC goes through all points in a multi-dimensional space for one and only one time each. When traversing a multi-dimensional space along an SFC, one can generate a linear order of all points, based on which the a mapping function can be established [25]. For example, to create a mapping between a 4-bit one-dimensional space $\mathbf{D}$ to a 2-bit two-dimensional space $(\mathbf{A}, \mathbf{B})$, a 4 $\times$ 4 SFC is necessary. The left part of Fig. 1 illustrates an example of such an SFC that travels along $\mathbf{0} \rightarrow \mathbf{1} \rightarrow \mathbf{2} \rightarrow \cdots$. At point $\mathbf{0}$, the 4-bit binary form of the integer number 0 is 0000, and the corresponding 2-bit two-dimensional coordinate is (00, 00). Therefore, the mapped value of 0 is (0, 0). Similarly, take point 2 for another example, its two-dimensional coordinate is (01, 01), thus the mapped value of 2 is (1, 1). By this means, we can obtain a one-to-one mapping between one-dimensional 4-bit data and two-dimensional 2-bit data, i.e., $\mathbf{D} \mapsto (\mathbf{A}, \mathbf{B})$.

There are different SFCs for different purposes with their own merits and limitations. Figure 2 shows another two representative two-dimensional SFCs and the corresponding binary bit conversation table. The sweep curve shown in Fig. 2(a) represents a simple mapping function which is equivalent to putting the most significant half of bits into the first channel and the least significant half of bits into the second channel. Figure 2(b) shows the Lebesgue curve [26] that can be recursively constructed by dividing a big square into subsquares. This SFC is widely employed since the corresponding one-dimensional value for a Lebesgue curve can be easily calculated by interleaving the binary representation of two-dimensional coordinates [24], as illustrated on the binary conversion table shown in Fig. 2(b). Figure 1 shows the Hilbert SFC curve [18] that was introduced by Hilbert, who had many remarkable achievements in different areas of mathematics in late 19th century and early 20th century. He was the first person to generalize a method to generate an entire class of SFCs geometrically [24]. The Hilbert curve can also be recursively constructed, but the mapping can not be simply represented by bit operations.
like the two aforementioned SFCs. In practice, a look-up-table (LUT) can be conveniently created to establish the one-to-one mapping. For example, to convert the 16-bit phase map into two 8-bit color channels, we generated a $256 \times 256$ LUT that only consumes 128 KB memory. It is also possible to encode the normalized phase map into three color channels based on a three-dimensional Hilbert curve [24], however it is not necessary because its resolution goes far beyond standard 3D scanners can achieve, as discussed earlier.

![Two other common two-dimensional space-filling curves demonstrated by a 4 x 4 grid. (a) Sweep curve; (b) Lebesgue curve.](image)

We used different SFCs to convert the normalized phase map of a complex statue into two channels for comparison. Figure 3 shows the result. Figure 3(e) shows the original 3D geometry and Fig. 3(a) shows the normalized phase map to be encoded. The normalized phase map is then mapped to the red and the green channels of a regular color image using different SFCs. Figure 3(b) shows the resultant image of the sweep curve, depicting many random structures. The random structures are inherent to this method since it simply encodes the most significant 8 bits into the red channel and the least significant 8 bits into the green channel. In comparison, when the Lebesgue curve is used, the encoded image has much fewer random structures, as shown in Fig. 3(c), albeit there are still some sharp edges. Figure 3(d) shows the encoded image for the Hilbert curve, which is apparently the smoothest among these three different SFCs. The choice of SFC does not matter when using a lossless image format such as PNG, since the exact same phase value can be accurately recovered for 3D reconstruction, as shown in Figs. 3(f)–3(h).

To further compress the data, lossy compression methods (e.g., JPEG) are necessary. In this case, however, the choice of SFC affects the quality of recovered 3D geometry. The differences can be significant especially when highly lossy compression is applied. Figure 4(a)–4(c) show the 3D reconstruction result after lossy compression with the sweep, Lebesgue, and Hilbert
Fig. 3. Encoded phase map and recovered 3D geometry of a statue using different SFCs. (a) Original normalized phase map; (b)–(d) normalized phase map encoded in the red and the green channels using sweep, Lebesgue, and Hilbert curves respectively; (e) original 3D geometry; (f)–(h) recovered 3D geometry from (b)–(d) when stored in lossless PNG format.

curves respectively, and Figs. 4(d)–4(f) show the corresponding close-up views. Clearly, due to the random structures, the quality of the reconstructed 3D geometry is very low for the sweep curve; in contrast, the recovered data has the highest quality when the Hilbert curve is employed. The image encoding and compression in this paper are processed by MATLAB 2017a, where a ‘JPG85’ (e.g.) format stands for JPEG encoding with a quality factor of 85%. (It is worth noting that a JPEG image of 100% quality is still a lossy compression format.) From the comparison, we can conclude that the Hilbert curve has the best performance in term of fidelity. Consequently, we choose the Hilbert curve to convert the normalized phase map to 16-bit data stored in two channels for the proposed compression method.

Since only two color channels are used to encode 3D geometry, the third color channel can be used to store additional information such as texture. For a lossless compression (e.g., PNG), the third channel can be used to store the Bayer-coded color texture. The Bayer-coding [27] is extensively employed on single-chip image sensors. It uses four local pixels to represent one color pixel. Essentially, each pixel has a color filter on top of the photo sensor such that the sensor only responds to one specific color spectrum of light (red, green, or blue). For example,
Fig. 4. Raw 3D geometry recovered from the encoded images shown in Figs. 3(b)–3(d) when stored in lossy JPG85 format. (a)–(c) 3D reconstruction using sweep, Lebesgue, and Hilbert curves, respectively; (d)–(f) corresponding zoom-in view of the 3D reconstruction above.

for an 8-bit camera, only one 8-bit image is created, from which a 24-bit color image can be reconstructed through the debayering/demosaicing process. However, lossy formats (e.g., JPEG) typically cannot preserve Bayer-coded color texture information when the intensities of three color components are drastically different. This is because the local 2×2 pixels of the Bayer-coded intensity image contains sharp edges (i.e., high frequency components in frequency domain) that are typically smoothed out by the lossy compression algorithm (i.e., the high frequency components are reduced). As a result, the color information is partially and sometimes completely lost depending on the level of lossy compression. Therefore, only grayscale texture can be stored and properly recovered when a lossy format is used to compress the encoded image.

The decoding process is rather straightforward: an encoded 2D color image is first used to recover the normalized phase map and the texture image; then the normalized phase is converted to the absolute unwrapped phase; finally we are able to compute the \((x, y, z)\) coordinates of objects point by point from the unwrapped phase map (the same as a standard 3D reconstruction process of the phase-based 3D shape measurement technique).

3. Experimental results

The proposed method was tested with both ideal 3D data and several different 3D objects captured by a real-time phase-based structured light system developed in our laboratory. First, an ideal sphere model with a 100 mm diameter was used to quantify errors and to compare between encoding phase into one channel and two channels with lossless and lossy compression. The
one-channel encoding uses the red channel to store the phase map and the blue channel to store the gray scale texture image, which leaves the green channel empty. Figure 5(a) shows the encoded image. The two-channel encoding uses both the red and the green channels to store the phase map encoded by the Hilbert curve and the blue channel to store texture, and its encoded image is shown in Fig. 5(e). From these encoded images, 3D geometry and texture can be recovered. Figures 5(b) and 5(c) show the 3D geometry recovered from Fig. 5(a) when it is compressed as lossless PNG and lossy JPG85 format respectively. Figure 5(d) shows the texture image decoded from Fig. 5(a) stored in JPG85 format. Figures 5(f)–5(h) are the corresponding recovered geometries and texture from Fig. 5(e). These results show that for lossless compression, there are some small but noticeable artifacts on the reconstructed surface when only one channel is used to store the phase map. The low resolution of 8-bit data causes these artifacts, and using 2 channels with 16-bit representation is sufficient to eliminate them. When the coded image is stored in lossy format, the two-channel encoding preserves evidently higher quality in term of 3D geometry, while there is slight difference in the recovered texture images. It should be noted that some post-processing techniques were applied on the phase map to improve the visual quality of the geometry recovered from lossy images: a $3 \times 3$ median filter and a $3 \times 3$ Gaussian filter were used to alleviate spikes and random noise, and a Laplace filter was applied to detect abrupt surface changes of small bumpy artifacts that were further removed by interpolation. For all the data presented in the rest of this paper, the same post-processing procedures were employed for reconstructions from lossy compression formats.

Fig. 5. Results when an ideal sphere of 100 mm diameter is encoded with texture using one or two channels storing its geometry and further compressed in PNG or JPG85 format. (a) Encoded color image with one channel storing geometry; (b) recovered 3D geometry from (a) compressed in PNG; (c) recovered 3D geometry from (a) compressed in JPG85; (d) recovered texture from (a) compressed in JPG85; (e) encoded color image with two-channel geometry encoding using the Hilbert curve; (f) recovered 3D geometry from (e) compressed in PNG; (g) recovered 3D geometry from (e) compressed in JPG85; (h) recovered texture from (e) compressed in JPG85.

We then calculated the compression error of the reconstructed 3D geometry (an ideal sphere
in this case). We calculated the root-mean-square (RMS) error of the depth value between the original and the reconstructed 3D data. Table 2 summarizes the RMS percent error for different image formats with one or two channels storing the phase map. The data shown on the table clearly demonstrate that the two-channel encoding preserves higher data quality regardless of the image format. One can see that even with a low-quality JPEG compression (e.g., JPG50), only 0.027% RMS error is introduced for the two-channel encoding method. Notably, when lossless PNG format is used, the two-channel encoding can achieve high precision with only an RMS error of 0.000037% (almost 100% accurate). Therefore, for the rest of this paper, we encode our data with two channels storing 3D geometries.

Table 2. RMS percent error of the depth value between the original and the reconstructed ideal sphere when the coded images are compressed in different image formats, using one or two channels to store the geometry.

<table>
<thead>
<tr>
<th></th>
<th>PNG</th>
<th>JPG100</th>
<th>JPG85</th>
<th>JPG70</th>
<th>JPG50</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-channel (%)</td>
<td>0.0089</td>
<td>0.020</td>
<td>0.042</td>
<td>0.059</td>
<td>0.078</td>
</tr>
<tr>
<td>2-channel (%)</td>
<td>0.000037</td>
<td>0.013</td>
<td>0.017</td>
<td>0.021</td>
<td>0.027</td>
</tr>
</tbody>
</table>

To further test the proposed compression method’s ability to preserve complex 3D geometries, a plaster statue was captured and compressed in different formats. The hardware system includes a digital light processing (DLP) projector (Texas Instruments LightCrafter 4500) and a camera (Point Grey Research Flea3 FL3-U3-13Y3M-C) with an 8 mm lens (Computar M0814-MP2). The resolutions of the camera and the projector are 480 × 640 and 912 × 1140, respectively. For all captured data, the enhanced two-frequency phase-shifting method [23] was employed for 3D reconstruction; and we used a fringe period of 24 pixels for the higher frequency fringe pattern, and a fringe period of 240 pixels for the lower frequency fringe patterns. Figure 6 shows the reconstructed 3D geometry using the lossless PNG format and different levels of lossy JPEG formats. One can see that a higher quality image has less random noise, less noticeable artifacts, and more detailed 3D geometry, yet no severe issues are visually detected even for high-level lossy compression.

![Fig. 6. Results when the a complex plaster statue is encoded with the proposed method and stored in different image formats. (a)–(e) 3D reconstruction from encoded 2D images stored in PNG, JPG100, JPG85, JPG70, and JPG50 respectively.](image)

We then computed the compression ratios by comparing the sizes of images stored in different formats against common 3D mesh formats (i.e., STL, OBJ, and PLY). Table 3 summarizes the data. All these meshes store the data in ASCII files including colorization information (UV coordinates and color for each pixel). The compression ratio are very high even for the lossless
PNG format. The lossy JPEG formats drastically increase the compression ratios. For example, the encoded output image has a size of only 59.3 KB for JPG85 format, while the original OBJ mesh file occupies 36.6 MB space, leading to a compression ratio of 618:1.

Table 3. Compression ratios of the proposed method when the coded images are stored in different image formats versus some standard mesh formats for the captured statue shown in Fig. 3(e).

<table>
<thead>
<tr>
<th></th>
<th>PNG</th>
<th>JPG100</th>
<th>JPG85</th>
<th>JPG70</th>
<th>JPG50</th>
</tr>
</thead>
<tbody>
<tr>
<td>STL</td>
<td>283:1</td>
<td>438:1</td>
<td>1589:1</td>
<td>2430:1</td>
<td>3308:1</td>
</tr>
<tr>
<td>OBJ</td>
<td>98:1</td>
<td>170:1</td>
<td>618:1</td>
<td>944:1</td>
<td>1285:1</td>
</tr>
<tr>
<td>PLY</td>
<td>42:1</td>
<td>74:1</td>
<td>269:1</td>
<td>411:1</td>
<td>559:1</td>
</tr>
</tbody>
</table>

Furthermore, another statue with complex colorful texture was captured to examine whether the proposed method can well preserve the texture data. Figure 7 shows the results. For lossless PNG compression, the exact pixel value of the raw Bayer-coded texture captured by the image sensor can be directly recorded in the third channel, and the full color image can be recovered after the debayering/demosaicing process. However, lossy JPEG formats cannot preserve Bayer-coded color information in only one channel, and the recovered texture appears to be grayscale texture as shown in Figs. 7(d) and 7(h). It is important to note that texture does not have any obvious impacts on recovered 3D geometry though they are compressed together in the same color images.

We also evaluated the performance of the proposed method with two separate statues. Figure 8 shows the result after lossy JPG85 compression. Obviously, both 3D geometry and texture can be properly recovered, verifying that the proposed method can work well for multiple objects.

Finally, we extended our proposed compression method to video encoding. We captured a video sequence of a hand with different gestures and another video sequence of various face expressions. Each frame of the encoded video contains 3D geometry and grayscale texture, and all encoded frames were stacked to generate a standard video using the H.264 codec. Figure 9 and the associated Visualization 1 and Visualization 2 show the rendered 3D results from the encoded H.264 video. The video compression normally has a better performance than plain image sequence and it has more adjustable parameters which can be optimized for this proposed method. For example, the video of human face achieves an additional compression ratio of 22.3:1 against frame-by-frame lossless PNG format (1543:1 against OBJ mesh sequence); the file size of this H.264 video is also 57% smaller than that of frame-by-frame JPG85 image sequence, yet the video has higher data quality as it features better compression algorithms for consecutive frames and uses a 4:4:4 instead of a 4:2:0 chroma sub-sampling.

4. Discussion

Compared with other state-of-the-art 2D image based 3D range data compression methods, the proposed method has the following merits:

- **High-quality geometry and texture representation.** The proposed method can represent high-quality and high-resolution 3D geometry and 2D texture within a regular 24-bit color image.

- **Fast implementation.** The proposed method only requires simple computation: phase normalization and one-to-one mapping based on LUT. This allows compression and decompression to be processed at a high frame rate.

- **No phase unwrapping.** The proposed method directly encodes unwrapped phase, and the decoding stage does not require the often complex phase unwrapping and time-consuming processing.
Fig. 7. Results when a colorful statue is encoded in lossless PNG and lossy JPG85 format, with two channels representing its geometry and one channel representing its texture. (a) Recovered 3D geometry from the encoded PNG image; (b) recovered color texture from the encoded PNG image; (c) recovered 3D geometry from the encoded JPG85 image; (d) recovered texture from the encoded JPG85 image; (e)–(h) zoomed-in view of (a)–(d) respectively.

Fig. 8. Results when a scene of two objects is encoded in JPG85 format. (a) Encoded output image; (b) recovered 3D geometry from (a); (c) recovered texture from (a).
Fig. 9. Several representative frames of the reconstructed 3D geometry from the H.264 videos (Associated Visualization 1 and Visualization 2). The video is encoded by FFmpeg codec as a ‘.mp4’ file, with the quality factor (CRF) equal to 18 and the frame rate equal to 24 frame per second. (a)–(d) Frames of 3D reconstruction from a video of different human hand gestures; (e)–(h) frames of 3D reconstruction from a video of various human facial expressions.

arctangent function calculations. Moreover, this method eliminates any artifacts introduced by incorrect phase unwrapping.

• Versatility. The proposed method can be extended to using any number of channels or bits to store 3D coordinates depending on the data precision requirements. For example, if the texture is not useful, one can use all three channels to encode 3D geometry for even higher resolution data representation with a three-dimensional Hilbert curve.

• Standard video compression. Unlike any of the previous methods, the proposed method allows the use of standard video compression techniques (e.g., H.264 codec) to encode a 3D video with grayscale textures and does not need to make any additional changes to the frames. Thus, this method enables direct 3D video/image conversion.

However, the proposed method still is limited to encode grayscale texture when using a lossy image or video compression method.

5. Summary

This paper has presented a method to effectively store three-dimensional (3D) data and 2D texture data into a regular 24-bit image. The one-to-one mapping from the normalized unwrapped phase into two 8-bit color channels is established by space-filling curves (SFCs), and thus leaves one channel for 2D texture storage. We have successfully demonstrated that high compression ratios can be achieved by leveraging existing 2D image and video compression techniques. For example, by using a JPG85 compression provided by MATLAB, we can achieve a compression ratio of 618:1 comparing against the standard ASCII OBJ file format, and a
A compression ratio of 1543:1 can be achieved without apparent compression artifacts if a H.264 codec is used. Furthermore, our experiments demonstrated that if a lossless 2D image/video format is used, both original 3D geometry and 2D color texture can be accurately recovered; and if a lossy image/video format is used, higher compression ratios can be achieved with slight loss of data quality, albeit only black-and-white or grayscale texture can be properly recovered.

**Funding**

Directorate for Engineering (ENG), National Science Foundation (NSF), (CMMI-1531048)

**Acknowledgments**

This study was partially sponsored by the startup funds and the Summer Undergraduate Research Fellowship (SURF) of Purdue University. The authors would like to thank Tyler Bell for some valuable discussions, and Chufan Jiang and Jae-sang Hyun for facilitating real-time 3D video recording system development and acquisition.