Compression of computer-generated holograms in image projection

Paula Kochańska, Michał Makowski

Faculty of Physics, Warsaw University of Technology, Koszykowa 75, 00-662 Warszawa

Received March 14, 2017; accepted April 20, 2017; published June 30, 2017.

Abstract—Computer-generated holography is a technique of lossless and lens-less forming of images. Methods that use local devices to compute such holograms are very power- and time-consuming. In order to make it possible to transfer the calculations to the cloud, it is necessary to elaborate efficient algorithms of lossless compression. In this paper two methods of compression are presented and supported by both simulation and experimental results. A lossy compression method omitting certain bit-planes of holographic data is also presented, which allows a insignificant loss of information, while achieving a greater compression ratio.

Computer-generated holograms (CGH) displayed on a spatial light modulator (SLM) are successfully used for real-time projection of images by lossless, diffractive redirection of light beams to certain areas on the screen or retina [1,2]. Such an approach to image formation allows a simple optical setup with greater efficiency, but on the other hand, the required calculations are very resource-consuming [3], especially when time-averaging of speckle noise is used [4]. For this reason, it is questionable whether mobile devices could potentially handle real-time calculations involving Fast Fourier Transforms (FFT) and high-resolution arrays. In this paper, we propose the calculations of holograms on a remote server, i.e. in the cloud [5]. To this end, an efficient compression method must be established, which would not destroy high-frequency fringes stored in holograms.

CGHs referred in this paper are calculated using the *Random Phase Free* algorithm [6]. The holographic data is stored as a two-dimensional matrix in the form of a bitmap (BMP) file with an 8-bit grayscale. This paper presents two algorithms of lossless compression of computer-generated holograms. Further, the analysis of holograms reconstructed from an incomplete number of bitplanes is shown.

Both proposed algorithms implement the extracting of partial bit-planes from entry bitmaps which contain the distribution of phase encoded modulo 2π, i.e. a kinoform-like encoding. Extracting bitplanes is achieved by performing a specific bitwise operation of logical AND on each pair of a pixel and mask. Mask m is defined as an 8-bit value where the *n* bit is set to 1 and the rest is set to 0, where *n* is the number of the bitplane. Adequate mathematical formula defining mask: $m = 0000\,0001 << n$.

After iterating each pixel, a two-dimensional array with 8 corresponding bitplanes is created. Subsequently, each bitplane is subjected to compression. In Fig.1, an example of input bitmap and that of extracted bit-planes are shown.

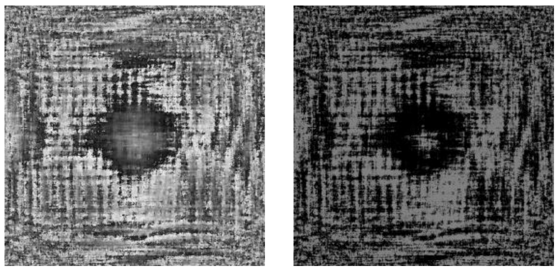


Fig. 1. Example of the input hologram and the extracted bitplane.

The first algorithm is based on run-length encoding (RLE) compression [7]. For each bitplane the number of consecutive pixels with the same level is counted. In the data, the brightness of the first pixel in sequence is written, having value 0 for black pixels and no information for bright ones. Therefore, the information of the gray level of the other pixels may be omitted, as corresponding sequences of bright and black pixels follow each other. The use of a random-phase-free algorithm allows slowly varying phase patterns which produce a long series suitable for RLE compression. The exemplary sequence shown in Fig. 2. will be coded as '*03212'*.

The data obtained from the RLE compression is then recalculated to a 4-bits representation. It means that one byte of data is used to keep the information about two 4-bits numbers. The 4-bit data may only store numbers smaller or equal to 15, which is a result of its binary representation – *00001111*. In one byte it is possible to write only two numbers lower than 15, which is a common case for the lower bitplanes. If the number is greater than 15, then it has to be written in two bytes of data.

Decompression starts with changing a 4-bits data representation to 8-bits. Afterwards pixels are being decoded. First, it is checked what is the first encoded number – if it is 0, then the first pixel is black, otherwise its brightness is calculated as 2^n, where n is the bitplane number. Then every pixel is written into an array and all the decoded bitplanes are joint into an output bitmap.

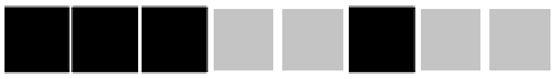


Fig. 2. Sequence of the pixels in the bitplane n=6.

The second method is based on data reorganization rather than compression. The idea is simple – after extracting bitplanes each one is encoded by writing a sequence of 8 pixels in one byte of data. If the pixel is bright then it is encoded as 1, otherwise it is represented by 0. The exemplary sequence shown in Fig. 2 using this byte-wise algorithm is encoded as *'00011011'*.

In order to decode the information, every encoded byte of bitplane data is being extracted and results in getting 8 pixels. E.g. byte *'11111000'* is decoded as a sequence of 5 bright pixels and 3 black. Again, the brightness of a pixel is calculated like in the 4-bits algorithm described above.

The last part of the decoding process is common for both methods. In this step bit-planes are joint into one output array. It is done by adding the corresponding pixel values together and writing them into an array. In this step is it possible to define the number of the bitplanes to join, so there is no need to reconstruct all of them.

The data compression ratios for the holograms presented in Fig. 3. are shown in Table 1.

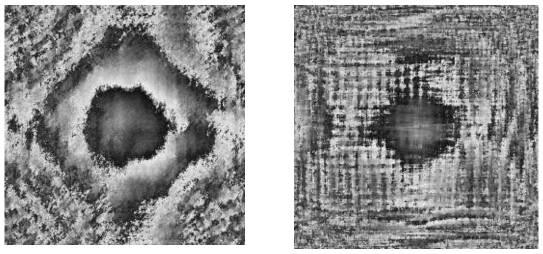


Fig. 3. Tested holograms, a) -400mm, b) -4000mm.

Algorithm	4-bits		Bytes	
Bitplane	size [kB]		size [kB]	
	a)	b)	a)	b)
0	259	259	128	128
1	259	259	128	128
2	259	259	128	128
3	260	259	128	128
4	260	257	128	128
5	254	228	128	128
6	220	168	128	128
7	159	138	128	128
sum	1930	1827	1024	1024
k	0.532	0.562	1.002	1.002

Table 1. Compression ratios for 4-bits and bytes algorithms.

Those holograms are holograms of the same object, but calculated for two different propagation distances a) 400mm, b) 4000mm. In Table 2 and Fig. 4 there are shown compression ratios and error of the reconstruction calculated as a root mean square error between an ideal reconstruction and a given hologram reconstructed with different methods of compression. Byte-wise and 4-bits algorithms were used for reconstructing only 3 or 4 most significant bit-planes. Clearly, the high compression ratios of JPEG algorithm suppress higher frequency components of the CGHs, leading to a huge root mean square error (RMSE) of over 20, which is unacceptable. On the other hand, the proposed byte-wise algorithm provides compression of approx. 2 with relatively low RMSE, as can be deducted from Fig. 4.

Hologram	Method	k	RMSE
a)	JPEG (Q=1)	18.159	43.187
	JPEG (Q=50)	2.651	20.765
	4-bits (3)	1.621	21.675
	4-bits (4)	1.149	11.279
	bytes (3)	2.672	21.675
	bytes (4)	2.004	11.279
b)	JPEG (Q=1)	27.003	33.705
	JPEG (Q=50)	3.523	19.929
	4-bits (3)	1.921	25.501
	4-bits (4)	1.297	13.584
	bytes (3)	2.672	25.501
	bytes (4)	2.004	13.584

Table 2. Compression ratios for different compression methods.

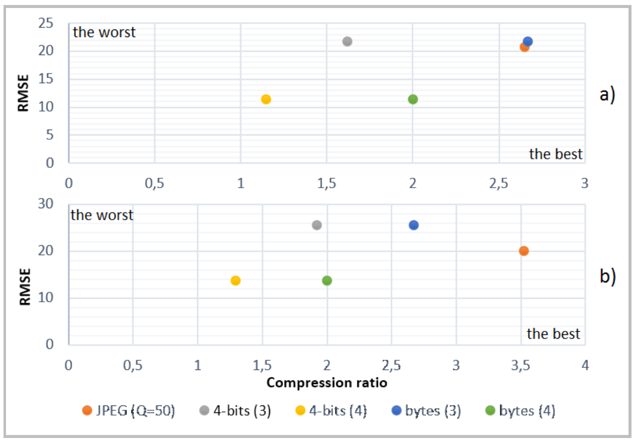


Fig. 4. Compression ratios for tested holograms.

Holograms with a reduced number of bitplanes were reconstructed in a numerical simulation and the results are shown in Fig. 5, where the first column ("bit-planes") shows which bit-planes were used for compression. RMSE as an error of reconstruction was calculated only on given analyzed fragments. The object represented in hologram a) in Fig. 3 was also reconstructed from an incomplete number of bit-planes in an experimental optical setup using a spatial light modulator and an He-Ne laser [1], which is shown in Fig. 6.

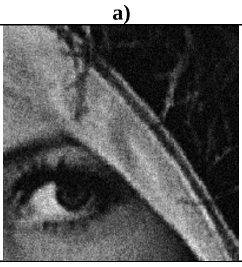
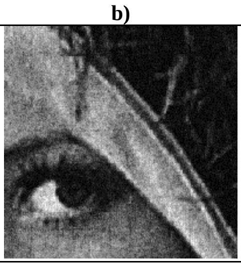
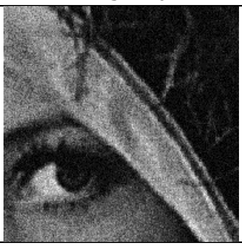
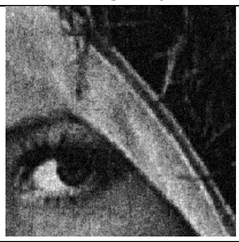
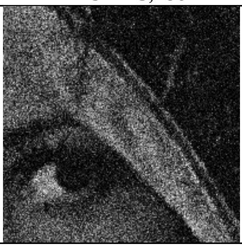
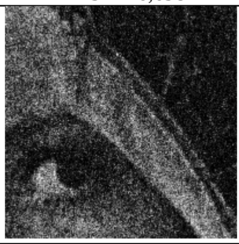
Bit-planes	**a)**	**b)**
1111 1111	 RMSE = 0	 RMSE = 0
1111 0000	 RMSE = 5,786	 RMSE = 6,698
1100 0000	 RMSE = 21,675	 RMSE = 24,501

Fig. 5. Simulated reconstructions with different number of bit-planes.

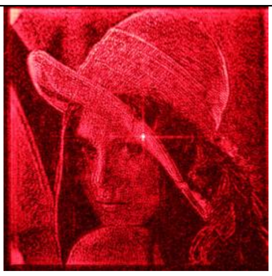
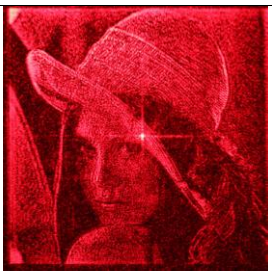
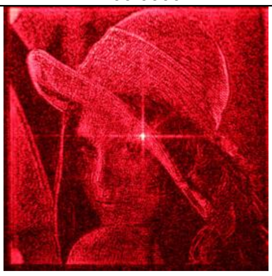
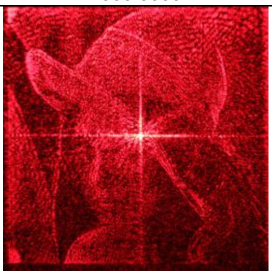
1111 1111	1110 0000
 c = 8.39	 c = 8.04
1100 0000	1000 0000
 c = 7.61	 c = 6.78

Fig. 6. Experimental reconstructions with a different number of used bit-planes.

For given reconstructions, contrast *c* was calculated as a merit of reconstruction quality. As can be clearly seen, the image quality does not rapidly decrease with the loss of bit-planes. Hence, this makes it possible to limit seriously the required data streams for holographic transmission.

In conclusions, compression methods based on bitplane extraction have advantages compared to other lossy and lossless algorithms. Dividing the data into separate binary files during compression and further transmission allows to recover the full encoded image if the transmission was interrupted. It is also important that this method allows, on the client side, to decide about the number of reconstructed bit-planes without engaging the server, which takes advantage of the unique features of computer holography.

Those advantages are crucial for holographic transmission in real-time. Slow data receiving could cause a delay in displaying the images or even improper reconstruction. The method proposed in this paper would make it possible to send only some of the compressed bitplanes, which would increase the number of transmitted holograms per time unit by causing only slight deterioration of reconstruction quality.

This work was supported by the Polish National Centre for Research and Development (No. LIDER/013/469/L-4/12/NCBR/2013) and the Polish National Science Centre (No. 2015/17/B/ST7/03754).

References

[1] M. Makowski, Opt. Expr. **20**, 25130 (2012).
[2] M. Makowski, I. Ducin, K. Kakarenko, J. Suszek, A. Kowalczyk, Phot. Lett. Poland **8**, 26 (2016).
[3] A. Kowalczyk, M. Bieda, M. Makowski, I. Ducin, K. Kakarenko, J. Suszek, A. Sobczyk, Phot. Lett. Poland **6**, 84 (2014).
[4] M. Makowski, Opt. Expr. **21**, 29205 (2013).
[5] H. Niwase, N. Takada, H. Araki, Y. Maeda, M. Fujiwara, H. Nakayama, T. Kakue, T. Shimobaba, T. Ito, Opt. Eng. **55**, 093108 (2016).
[6] T. Shimobaba, T. Ito, Opt. Expr. **23**(7) 9549 (2015).
[7] S.R. Kodituwakku, Indian J. Computer Science and Engineering, **1**(4), 416 (2010).