Reversible Data Hiding for Texture Videos and Depth Maps Coding with Quality Scalability

Yuanzhi Yao(✉), Weiming Zhang, and Nenghai Yu

Department of Electronic Engineering and Information Science,
University of Science and Technology of China, Hefei 230027, China
yaoyz@mail.ustc.edu.cn, {zhangwm,ynh}@ustc.edu.cn

Abstract. To support 3-D video and free-viewpoint video applications, efficient texture videos and depth maps coding should be addressed. In this paper, a novel reversible data hiding scheme is proposed to integrate depth maps into corresponding texture video bitstreams. At the sender end, the depth video bitstream obtained by depth down-sampling and compression is embedded in residual coefficients of corresponding texture video. The data embedding is implemented by the histogram shifting technique. At the receiver end, the depth maps can be retrieved with scalable quality after data extraction, video decoding and texture-based depth reconstruction. Due to the attractive property of reversible data hiding, the texture video bitstream can be perfectly recovered. Experimental results demonstrate that the proposed scheme can achieve better video rendering quality and coding efficiency compared with existing related schemes.

Keywords: Reversible data hiding · Depth map · Depth down-sampling · Histogram shifting · Texture-based depth reconstruction

1 Introduction

Differing from traditional 2-D imaging, the emerging stereoscopic imaging can provide arbitrary views of real 3-D scenes. Due to the increasing demand for 3-D video and free-viewpoint video applications, the stereoscopic imaging has been regarded as the next generation visual technology. One of the key problems for stereoscopic imaging is how to efficiently represent real 3-D scenes. Among many 3-D scene representation technologies [1], the texture-plus-depth representation has been extensively studied because of its low computing cost and high rendering quality. In the texture-plus-depth representation [2], virtual views can be synthesized from the acquired texture videos and their corresponding depth maps. The depth map records the relative distance between the observed object and the camera. Therefore, texture-plus-depth video coding and transmission have attracted considerable research effort [3–5]. In 3-D video coding, texture videos and depth maps should be jointly considered for obtaining the synthesized virtual views with high quality. Either the distortion of texture videos or the distortion of depth maps can degrade

Y.Q. Shi et al. (Eds.): IWDW 2016, LNCS 10082, pp. 421–435, 2017.
DOI: 10.1007/978-3-319-53465-7_31

the quality of synthesized virtual views. Furthermore, the stereoscopic visual effect will no longer exist without depth maps in the texture-plus-depth representation.

With regard to the importance of depth maps, the transmission of depth maps can be achieved using the covert communication [6]. Hence, many schemes have been proposed to integrate the depth map into corresponding texture image/video using data hiding to address the security concern. Khan et al. [7] proposed to embed the depth map in its corresponding 2-D image to secure the transmission of the depth map. In [8], Tong et al. embedded the disparity of two stereo images in one of them for reducing storage space. In Wang et al.'s scheme [9], the depth map is embedded in quantized DCT coefficients of the JPEG image using the difference expansion technique [10]. Jung [11] proposed to embed the depth map in the JPEG compressed bitstream of corresponding texture image using super-pixel image segmentation and depth value clustering. Due to the spatial and temporal correlation of depth maps, the texture-plus-depth representation format can also be compressed using the efficient video coding technologies, such as H.264/AVC [12]. In [13,14], Wang et al. extended their watermarking scheme of coding depth maps from JPEG image to H.264/AVC video. They embedded the depth video bitstream obtained by compressing depth maps in quantized DCT coefficients of corresponding texture video.

The major concern of embedding depth maps in corresponding texture video is that the embedding capacity of texture video is limited due to high-efficiency video coding. Therefore, the difference expansion technique [10] is utilized in the schemes [13,14] to obtain relatively higher embedding capacity. However, data embedding in quantized DCT coefficients using the difference expansion technique can induce severe distortion in texture videos. As an alternative to the schemes [13,14], Shahid et al. [15] proposed to embed the depth video bitstream obtained by depth down-sampling and compression in quantized DCT coefficients of corresponding texture video using LSB replacement. Then, the depth maps can be reconstructed after data extraction, video decoding and depth up-sampling. The depth down-sampling is effective on reducing the size of depth video bitstream. Moreover, the depth maps can be reconstructed with scalable quality by adjusting the scaling factor. Unfortunately, the texture video cannot be losslessly recovered without reversible data hiding in [15]. Due to the motion compensation, the distribution of quantized DCT coefficients in videos can be modeled as the Laplacian probability distribution [16,17], which provides enough embedding capacity for the histogram shifting technique [18,19]. To obtain a tradeoff between the embedding capacity and the stego texture video quality, the histogram shifting and the depth down-sampling should be jointly used. In the depth down-sampling-based scheme, we should reconstruct the depth map where missing depth samples need to be estimated after data extraction and video decoding. According to the grouping principle in [20], the correlation of depth samples is associated with the similarity and the proximity of corresponding texture samples. It means that we can reconstruct the depth map with high quality based on the texture video. However, Shahid et al.'s scheme [15] does not utilize corresponding texture videos for reconstructing depth maps. Hence, new

texture videos and depth maps coding schemes which support reversible depth embedding and texture-based depth reconstruction should be sought.

We propose a novel reversible data hiding scheme of integrating depth maps into corresponding texture video bitstreams for addressing the secure 3-D video coding in this paper. At the sender end, the depth video bitstream obtained by depth down-sampling and compression is embedded in residual coefficients of corresponding texture video. The data embedding is implemented by the histogram shifting technique. At the receiver end, the depth maps can be retrieved with scalable quality after data extraction, video decoding and texture-based depth reconstruction. The reversibility guarantees that the texture video bitstream can be losslessly recovered after data extraction. In conclusion, the highlights of this paper can be summarized as follows.

- The proposed scheme can embed the depth video bitstream in corresponding texture video with low distortion.
- The texture video bitstream can be losslessly recovered after data extraction.
- The depth maps can be reconstructed with scalable quality by adjusting the scaling factor.
- Texture-based depth reconstruction is used to obtain the depth maps with high quality.

The remainder of the paper is organized as follows. In Sect. 2, the proposed reversible data hiding scheme for texture videos and depth maps coding is elaborated. The experimental results are presented in Sect. 3. Section 4 finally concludes this paper.

2 Proposed Reversible Data Hiding Scheme for Texture Videos and Depth Maps Coding

In this section, the proposed reversible data hiding scheme for texture videos and depth maps coding is discussed in detail. Figure 1 illustrates the framework of the proposed scheme. The proposed scheme contains three components which are depth down-sampling and compression, depth video bitstream embedding and extraction, and texture-based depth reconstruction.

2.1 Depth Down-Sampling and Compression

In the 3-D video system, texture video sequences are captured by the camera array which is composed of several cameras with a certain interval. The depth maps can be acquired by depth estimation from adjacent-view texture video sequences. The real depth value is in floating-point type generally. To compress depth data using existing video coding technologies, real depth values should be quantized to the depth map in which sample values range from 0 to 255 [2]. Like the texture video sequence, the depth maps are also a series of images with spatial and temporal correlation. Afterwards, texture video sequences and depth maps can be compressed using existing video coding technologies, such as H.264/AVC [12].

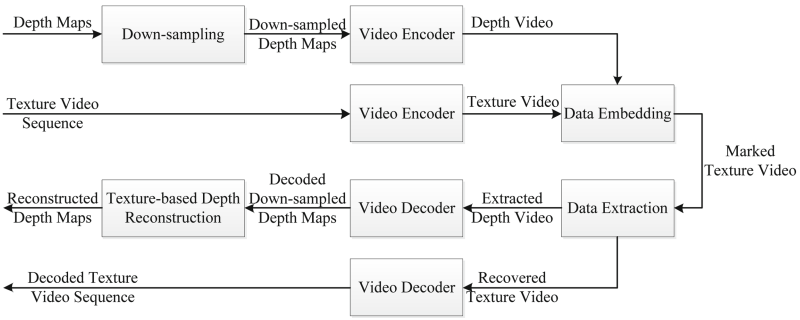


Fig. 1. Framework of the proposed reversible data hiding scheme.

To reduce the size of depth video bitstream, depth down-sampling is performed before compressing the depth maps. Differing from the texture image, the depth map usually has a well-defined object boundary. Therefore, it is important to maintain the object boundaries of the depth map for high-quality rendering at the receiver end. Considering that object boundaries are sensitive to coding errors, the edge-preserving filters [21,22] can be used to eliminate coding errors and refine object boundaries. Without loss of generality, the depth maps are down-sampled by dropping samples as follows.

$$Depth_{down}(i,j) = Depth(i \cdot step, j \cdot step) \tag{1}$$

where *step* is the scaling factor. Then, the down-sampled depth maps are compressed by the video encoder to generate the depth video bitstream.

2.2 Depth Video Bitstream Embedding and Extraction

To address the secure transmission of depth maps, embedding the depth video bitstream obtained by depth down-sampling and compression in the texture video is a feasible solution. Reversible data hiding should be used to losslessly recover the texture video after data extraction. In video data hiding, when we embed data in the current frame, the embedding distortion in the current frame will propagate to subsequent frames. This is called the inter-frame distortion drift. Therefore, how to decrease the inter-frame distortion drift caused by data embedding deserves investigation.

We only focus on the distortion drift of P-frames, because B-frames do not cause distortion drift and I-frames are not used for data embedding. We suppose that the total frame number of the texture video is N and the GOP (Group of Pictures) structure of the texture video is IPPP, in which only the first frame is encoded as an I-frame and the remaining frames are encoded as P-frames.

According to the inter-frame distortion drift analysis in [23], the overall distortion $D(n)$ of the nth video frame at the receiver end is given by

$$D(n) = D_s(n) + D_e(n) \tag{2}$$

where $D_s(n)$ is the source distortion caused by lossy compression and $D_e(n)$ is the embedding distortion caused by embedding data in quantized DCT coefficients of the nth video frame. When the texture video bitstream is given, $D_s(n)$ is constant. Therefore, to decrease the overall distortion $D(n)$ and improve the decoded texture video quality, the left problem is to decrease $D_e(n)$ during data embedding [23]. We can rewrite Eq. (2) as follows.

$$\begin{aligned} D(n) &= D_s(n) + D_e(n) \\ &= D_s(n) + \alpha \cdot D_e(n-1) + p \cdot \mathrm{E}\{[\tilde{r}(n,i) - \hat{r}(n,i)]^2\} \end{aligned} \tag{3}$$

where α is a constant relating to the video content, p is the probability of embedding data in the nth frame ($n \geq 2$), and $E\{x(n,i)\}$ is denoted as the average (over all pixels) expected value of the random variable $x(n,i)$. In Eq. (3), $\tilde{r}(n,i)$ and $\hat{r}(n,i)$ represent the ith reconstructed residues of the nth video frame with embedded data and without embedded data respectively. It can be implied from Eq. (3) that we should decrease $\mathrm{E}\{[\tilde{r}(n,i) - \hat{r}(n,i)]^2\}$ to decrease the embedding distortion in the nth frame and the embedding distortion of each frame can accumulate as the frame index increases.

According to the property of integer 4×4 transform in H.264/AVC [24], the (i,j)th quantized DCT coefficient in the 4×4 block can be classified according to its induced independent embedding distortion[1] as follows.

$$\begin{cases} C_1 = \{(2,2),(4,2),(2,4),(4,4)\} \\ C_2 = \{(1,2),(2,1),(1,4),(2,3),(3,2),(4,1),(3,4),(4,3)\} \\ C_3 = \{(1,1),(3,1),(1,3),(3,3)\} \end{cases} \tag{4}$$

In Eq. (4), C_1, C_2 and C_3 represent the coefficient classification sets, in which modifying the coefficients in the given set can induce approximately equal embedding distortions. It has been verified that embedding data into coefficients in C_1 can induce the minimum distortion and embedding data into coefficients in C_3 can induce the maximum distortion [23]. Therefore, it is necessary to embed data using coefficient selection, which means that data should be embedded in the coefficients in C_1 with highest priority, the coefficients in C_2 with medium priority and the coefficients in C_3 with lowest priority.

The depth video bitstream is the to-be-embedded message sequence $\mathbf{b} = (b_1, b_2, \cdots, b_m)$ with length m. The Laplacian probability distribution of quantized DCT coefficients in videos [16,17] provides enough embedding capacity for the histogram shifting technique [18]. Therefore, we denote $T_p = 1$ and $T_n = -1$

[1] The independent embedding distortion is calculated by adding an error δ to the (i,j)th quantized DCT coefficient in the 4×4 block using the MSE (Mean Squared Error) measure.

as the two highest bins in the histogram of quantized luma DCT coefficients in the texture video. (Zero coefficients are not used for data embedding.) The embedding capacity is the sum of the number of bin T_p and the number of bin T_n. We describe the data embedding algorithm as follows.

– *Case 1*: If $|z_{ij}| > 1$, shift z_{ij} by 1 unit to the right or the left respectively.

$$z'_{ij} = \begin{cases} z_{ij} + 1 & \text{if } z_{ij} > 1 \\ z_{ij} - 1 & \text{if } z_{ij} < -1 \end{cases} \tag{5}$$

where z_{ij} is the (i, j)th original quantized DCT coefficient in the 4×4 block and z'_{ij} is the corresponding stego coefficient.

– *Case 2*: If $|z_{ij}| = 1$, modify z_{ij} according to the to-be-embedded message bit b_k.

$$z'_{ij} = \begin{cases} z_{ij} & \text{if } |z_{ij}| = 1 \text{ and } b_k = 0 \\ z_{ij} + 1 & \text{if } z_{ij} = 1 \text{ and } b_k = 1 \\ z_{ij} - 1 & \text{if } z_{ij} = -1 \text{ and } b_k = 1 \end{cases} \tag{6}$$

– *Case 3*: If $z_{ij} = 0$, z_{ij} remains unchanged.

$$z'_{ij} = z_{ij} \tag{7}$$

Using the above data embedding algorithm, the following steps should be conducted to complete embedding the depth video bitstream in the corresponding texture video.

– *Step 1*: Partial decoding is performed on the texture video bitstream to obtain the quantized luma DCT coefficients. The data embedding starts in the Nth frame.
– *Step 2*: Embed message bits in inter-coded macroblocks (excluding P_Skip mode coded macroblocks) within the current frame using coefficient selection. It means that data should be embedded into the coefficients in C_1 with highest priority, the coefficients in C_2 with medium priority and the coefficients in C_3 with lowest priority [23]. During data embedding process, record the number of embedded message bits k.
– *Step 3*: If $k < m$, return to *Step 2* to embed data in previous frames one by one until all message bits have been embedded.
– *Step 4*: Partial encoding is performed on the quantized luma DCT coefficients to generate the stego texture video bitstream.

The number of embedded message bits m in the stego texture video bitstream should be synchronized between the sender end and the receiver end. Before data extraction, partial decoding is performed on the stego texture video bitstream to obtain the quantized luma DCT coefficients. The data extraction is the inverse process of data embedding. We can extract the embedded depth video bitstream $\mathbf{b} = (b_1, b_2, \cdots, b_m)$ as depicted in Eq. (8).

$$b_k = \begin{cases} 0 & \text{if } |z'_{ij}| = 1 \\ 1 & \text{if } |z'_{ij}| = 2 \end{cases} \tag{8}$$

where z'_{ij} is the (i, j)th stego quantized DCT coefficient in the 4×4 block.

The original quantized DCT coefficient recovery is described as follows.

$$z_{ij} = \begin{cases} z'_{ij} & \text{if } z'_{ij} \in \{-1, 0, 1\} \\ z'_{ij} - 1 & \text{if } z'_{ij} \geq 2 \\ z'_{ij} + 1 & \text{if } z'_{ij} \leq -2 \end{cases} \tag{9}$$

Since the data embedding process is reversible, the texture video can be losslessly recovered after data extraction.

2.3 Texture-Based Depth Reconstruction

After extracting the embedded depth video bitstream and decoding it, we can only obtain the decoded down-sampled depth maps. To reconstruct the depth map, the missing samples whose horizontal or vertical coordinates are multiples of the scaling factor need to be estimated. According to the analysis in [20], the correlation of depth samples is associated with the similarity and the proximity of corresponding texture samples. It is an effective way to reconstruct the depth map to its original resolution with high quality based on the texture video.

In the texture-based depth reconstruction, each existing depth sample has different weight on estimating missing depth samples based on the similarity and the proximity of corresponding texture samples. Moreover, the similarity and the proximity of texture samples can be measured separately. Therefore, the weight of existing depth sample can be calculated by

$$\omega(p, q) = \exp[-(\frac{\triangle c_{pq}}{\lambda_c} + \frac{\triangle g_{pq}}{\lambda_g})] \tag{10}$$

where $\triangle c_{pq}$ and $\triangle g_{pq}$ represent the distance between the colors of texture samples p and q, and the distance between texture samples p and q respectively [20]. λ_c and λ_g are two parameters to control the strength of similarity and the strength of proximity respectively. In Eq. (10), the $\triangle c_{pq}$ is calculated from the corresponding texture video using RGB color space in the ℓ_2-norm as follows.

$$\triangle c_{pq} = \sqrt{\sum_{C \in \{R,G,B\}} (C_p - C_q)^2} \tag{11}$$

where C_p and C_q are the color component values of texture samples p and q respectively. In Eq. (10), the $\triangle g_{pq}$ is calculated in the ℓ_1-norm[2] as follows.

$$\triangle g_{pq} = |i_p - i_q| + |j_p - j_q| \tag{12}$$

where (i_p, j_p) and (i_q, j_q) are the coordinates of texture samples p and q respectively.

[2] In [20], it is suggested that $\triangle g_{pq}$ should be calculated in the ℓ_2-norm. Actually, the better reconstructed depth map quality can be obtained if $\triangle g_{pq}$ is calculated in the ℓ_1-norm according to our experiments.

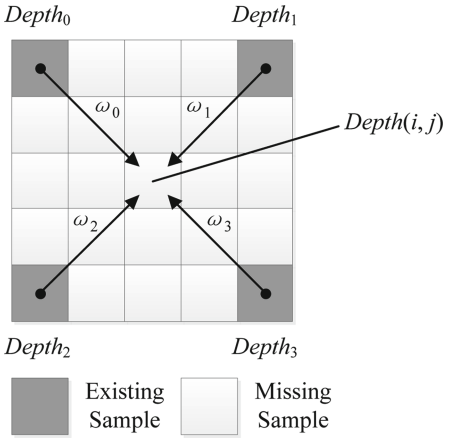


Fig. 2. Illustration of texture-based depth reconstruction.

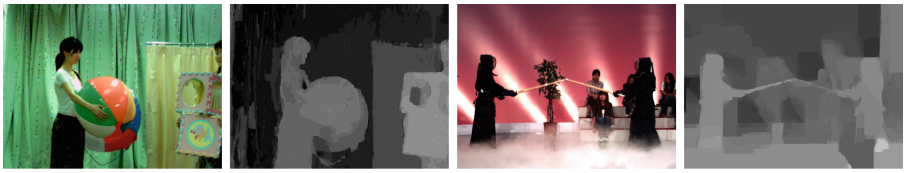
The texture-based depth reconstruction algorithm can be illustrated in Fig. 2, in which the scaling factor is set as 4. To estimate the missing depth sample $Depth(i, j)$, we can locate four nearest existing depth samples which are $Depth_0$, $Depth_1$, $Depth_2$ and $Depth_3$ respectively. Using texture-based depth reconstruction, we can calculate the weight ω_k for the missing depth sample $Depth(i, j)$ and the nearest existing depth sample $Depth_k$ where $k \in \{0, 1, 2, 3\}$ as depicted in Eq. (10). When we have four weights, we can estimate the missing depth sample $Depth(i, j)$ using texture-based depth reconstruction as depicted in Eq. (13).

$$Depth(i,j) = \frac{\omega_0 \cdot Depth_0 + \omega_1 \cdot Depth_1 + \omega_2 \cdot Depth_2 + \omega_3 \cdot Depth_3}{\omega_0 + \omega_1 + \omega_2 + \omega_3} \tag{13}$$

Using the above texture-based depth reconstruction algorithm, the down-sampled depth map can be reconstructed to its original resolution with high quality.

3 Experimental Results

The proposed reversible data hiding scheme for texture videos and depth maps coding has been implemented in the H.264/AVC reference software JM 10.2 [25]. Besides, the schemes proposed by Wang et al. [14] and Shahid et al. [15] are also implemented for performance comparison. As shown in Fig. 3, two free-viewpoint video sequences [26] which are Akko&Kayo with resolution 640×480 and Kendo with resolution 1024×768 respectively are used in the experiments. We use commonly adopted measurements PSNR and SSIM [27] to evaluate the objective visual quality of texture videos and depth maps. The PSNR (dB) and the SSIM are calculated by comparing the decoded reconstructed video sequence with the original video sequence. Texture video sequences are encoded using QP



(a) (b) (c) (d)

Fig. 3. Free-viewpoint video sequences used in the experiments. (a) Akko&Kayo, (b) Depth map of Akko&Kayo, (c) Kendo, and (d) Depth map of Kendo.

16 and depth maps are encoded using QP 28. (QP is the quantization parameter.) The first 100 frames of each texture video sequence and depth maps are encoded at frame rate 30 fps. In the texture-based depth reconstruction in Eqs. (10) and (13), the parameter λ_c is set as the maximum value of color distances in the texture video[3] and the parameter λ_g is set as the scaling factor.

3.1 Visual Quality of the Stego Texture Video

In the experiments, video sequences Akko&Kayo in view 27 and Kendo in view 1 are used for comparing the visual quality of stego texture videos. The corresponding depth maps of each texture video sequence are down-sampled with different scaling factors respectively. Afterwards, the depth video bitstreams are embedded in the corresponding texture videos using Wang et al.'s scheme [14], Shahid et al.'s scheme [15] and our proposed scheme respectively.

The comparison of average luma PSNR (dB) of stego texture videos is listed in Table 1. The visual quality of cover texture videos without embedded data is also presented in Table 1. The bold digits in Table 1 mean that the corresponding scheme can achieve the best visual quality of stego texture videos. Our proposed scheme focuses on decreasing the inter-frame distortion drift so that it outperforms Wang et al.'s scheme and Shahid et al.'s scheme in the measurement of average luma PSNR. The notation "–" in Table 1 means that the stego texture video cannot contain the corresponding depth video bitstream. In Wang et al.'s scheme, the difference expansion technique is used to embed data so that the highest embedding capacity can be guaranteed. However, severe distortions are induced in stego texture videos.

Figure 4 shows the subjective results of the 16th frame, 31st frame, 46th frame, 61st frame and 76th frame of decoded texture video sequence Akko&Kayo in view 27 with scaling factor 4. Obvious visual distortion can be seen using Wang et al.'s scheme. Shahid et al.'s scheme and our proposed scheme can achieve better subjective visual quality.

[3] The color distance in the texture video is calculated between two texture samples which correspond to the missing depth sample and the nearest existing depth sample respectively.

Table 1. Comparison of average luma PSNR (dB) of stego texture videos.

Video sequence	Scaling factor	PSNR (dB)			
		Cover	Wang et al. [14]	Shahid et al. [15]	Proposed
Akko&Kayo	2	46.54	21.21	–	–
	4	46.54	21.20	31.70	**40.49**
	8	46.54	20.73	36.61	**44.64**
	16	46.54	21.60	38.54	**45.93**
Kendo	2	47.43	25.63	34.75	**38.23**
	4	47.43	25.34	36.01	**44.67**
	8	47.43	25.03	40.14	**46.46**
	16	47.43	24.95	42.79	**47.18**

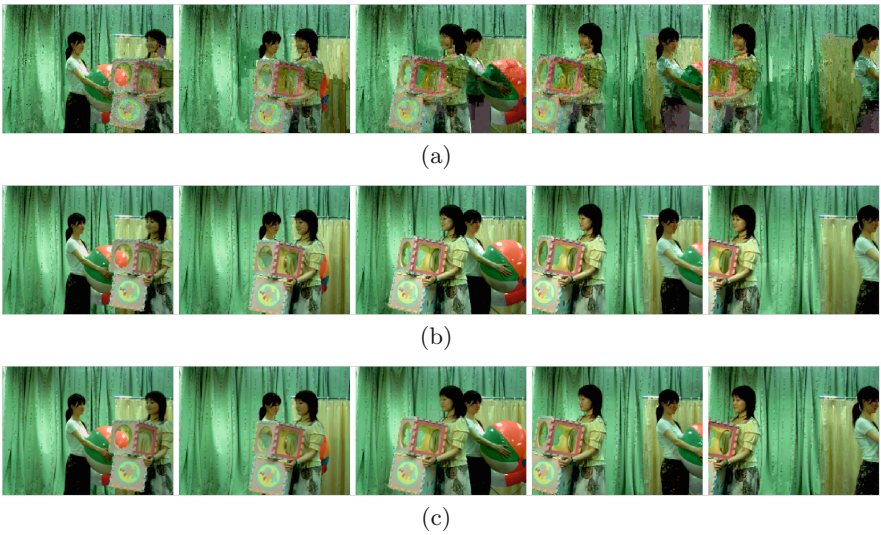


Fig. 4. Subjective results of decoded texture video sequence Akko&Kayo in view 27 with scaling factor 4. (a) Wang et al. [14], (b) Shahid et al. [15], and (c) Proposed.

3.2 Impacts on the Coding Efficiency

We measure the coding efficiency using the video bit rate increment BR_{inc} as follows.

$$BR_{inc} = \frac{BR_{texture+depth} - BR_{texture}}{BR_{texture}} \times 100\% \tag{14}$$

where $BR_{texture+depth}$ is the video bit rate used for coding both texture videos and depth maps and $BR_{texture}$ is the video bit rate used for coding texture videos. In data hiding-based schemes, $BR_{texture+depth}$ is equal to the bit rate of stego texture video. In the experiments, video sequences Akko&Kayo in view

Table 2. Comparison of video bit rate increments.

Video sequence	Scaling factor	BR_{inc}(%)			
		Separate coding	Wang et al. [14]	Shahid et al. [15]	Proposed
Akko&Kayo	2	17.07	**13.20**	–	–
	4	17.07	5.41	6.93	**5.07**
	8	17.07	2.28	2.61	**1.92**
	16	17.07	1.29	1.20	**0.87**
Kendo	2	10.69	**8.86**	12.35	9.02
	4	10.69	3.76	5.00	**3.59**
	8	10.69	1.81	2.19	**1.55**
	16	10.69	1.00	0.97	**0.65**

27 and Kendo in view 1 are used. The comparison of video bit rate increments among Wang et al.'s scheme [14], Shahid et al.'s scheme [15] and our proposed scheme is listed in Table 2, in which bold digits mean that the corresponding scheme can achieve the smallest video bit rate increment. The notation "–" in Table 2 means that the stego texture video cannot contain the corresponding depth video bitstream. Separate coding denotes coding texture videos and depth maps independently without depth down-sampling and data hiding. Data hiding-based schemes can obtain smaller BR_{inc} and have higher coding efficiency compared with separate coding.

3.3 Visual Quality of the Reconstructed Depth Maps

To compare the visual quality of reconstructed depth maps, original depth maps of video sequences Akko&Kayo in view 27 and Kendo in view 1 are used. Without down-sampling depth maps, the average luma PSNR values of depth videos obtained by compressing depth maps of video sequences Akko&Kayo and Kendo are 41.03 dB and 47.23 dB respectively. Because the depth reconstruction algorithm is not explicit in Shahid et al.'s scheme [15], the decoded down-sampled depth maps are reconstructed using nearest-neighbor interpolation, bilinear interpolation, Wildeboer et al.'s scheme [28] and our proposed scheme respectively in the experiments. Nearest-neighbor interpolation and bilinear interpolation are implemented using OpenCV 2.4.9 [29]. A variant of nearest-neighbor interpolation which considers the texture sample distance and the color distance of texture samples is utilized in Wildeboer et al.'s scheme.

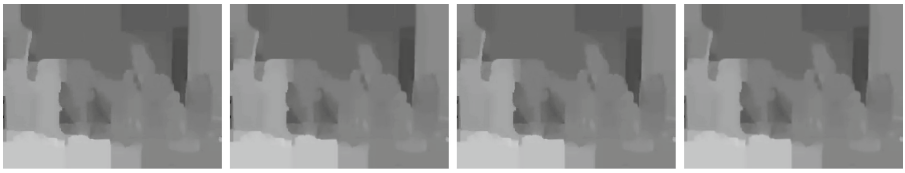
The comparison of average luma PSNR (dB) and average luma SSIM of reconstructed depth maps is listed in Table 3, in which bold digits mean that the corresponding scheme can achieve the best visual quality of reconstructed depth maps. Moreover, the subjective quality of the 51st reconstructed depth maps of video sequences Akko&Kayo in view 27 and Kendo in view 1 with scaling

Table 3. Comparison of average luma PSNR (dB) and average luma SSIM of reconstructed depth maps.

Video sequence	Scaling factor	PSNR (dB)				SSIM			
		Nearest-neighbor	Bilinear	Wildeboer et al. [28]	Proposed	Nearest-neighbor	Bilinear	Wildeboer et al. [28]	Proposed
Akko & Kayo	2	34.39	35.30	34.38	**36.07**	0.9196	0.9254	0.9188	**0.9297**
	4	29.99	31.17	30.04	**32.82**	0.8589	0.8774	0.8604	**0.8921**
	8	26.20	27.45	26.08	**29.54**	0.8143	0.8305	0.8156	**0.8572**
	16	22.76	24.04	22.56	**26.28**	0.8047	0.7891	0.8043	**0.8241**
Kendo	2	42.10	42.69	42.03	**43.17**	0.9798	0.9815	0.9795	**0.9822**
	4	37.83	38.68	37.85	**39.94**	0.9646	0.9701	0.9646	**0.9731**
	8	33.95	34.96	33.96	**36.90**	0.9495	0.9575	0.9495	**0.9634**
	16	30.42	31.58	30.43	**33.76**	0.9481	0.9451	0.9481	**0.9558**

(a) (b) (c) (d)

Fig. 5. Reconstructed depth maps of video sequence Akko&Kayo in view 27 with scaling factor 4. (a) Nearest-neighbor interpolation with PSNR = 27.940 dB and SSIM = 0.8380, (b) Bilinear interpolation with PSNR = 29.164 dB and SSIM = 0.8555, (c) Wildeboer et al.'s scheme [28] with PSNR = 27.962 dB and SSIM = 0.8395, and (d) Proposed scheme with PSNR = 30.999 dB and SSIM = 0.8781.



(a) (b) (c) (d)

Fig. 6. Reconstructed depth maps of video sequence Kendo in view 1 with scaling factor 4. (a) Nearest-neighbor interpolation with PSNR = 38.442 dB and SSIM = 0.9674, (b) Bilinear interpolation with PSNR = 39.266 dB and SSIM = 0.9717, (c) Wildeboer et al.'s scheme [28] with PSNR = 38.497 dB and SSIM = 0.9676, and (d) Proposed scheme with PSNR = 40.592 dB and SSIM = 0.9757.

factor 4 can be seen in Figs. 5 and 6 respectively, in which the luma PSNR value and the luma SSIM value of each frame are presented. Through comparisons, it can be observed that our proposed scheme can achieve the best visual quality of reconstructed depth maps. By adjusting the scaling factor, the depth maps can be reconstructed with scalable quality.

3.4 Visual Quality of the Synthesized View

Before view synthesis, we firstly down-sample original depth maps of video sequences Akko&Kayo in view 27 and view 29 with scaling factor 4 and compress them. Then, the depth video bitstreams are embedded in corresponding texture videos. After data extraction, video decoding and depth reconstruction, the reconstructed depth maps in view 27 and view 29 can be obtained. The texture video sequence Akko&Kayo in view 28 is synthesized from adjacent view 27 and view 29 using the depth estimation and view synthesis software [30].

The comparison of visual quality of the 51st synthesized frames in view 28 is given in Fig. 7, in which the luma PSNR value and the luma SSIM value of each synthesized frame are presented. The visual quality of synthesized frames depends on the decoded texture video sequence and the reconstructed depth maps. The 51st original frame in view 28 captured by the camera array is presented as the ground truth. We set two comparison targets as follows. In comparison target Tar1, Shahid et al.'s scheme [15] is used for data embedding and bilinear interpolation is used for depth reconstruction. In comparison target Tar2, our proposed data embedding algorithm and Wildeboer et al.'s depth reconstruction algorithm [28] are used. The average luma PSNR values of all synthesized frames in view 28 using Tar1, Tar2 and our proposed scheme are 27.49 dB, 30.03 dB and 30.05 dB respectively. The average luma SSIM values of all synthesized frames in view 28 using Tar1, Tar2 and our proposed scheme are 0.8692, 0.8945 and 0.8970 respectively. In Tar1, the decoded texture video sequence cannot be losslessly recovered after data extraction, which is permanently distorted due to data embedding. In Tar2, the visual quality of reconstructed depth maps is lower than that in our proposed scheme. Due to the superiority of data embedding and depth reconstruction in our proposed scheme, the best visual quality of synthesized view 28 can be obtained as shown in Fig. 7.

(a) (b) (c) (d)

Fig. 7. Synthesized view rendering of video sequence Akko&Kayo in view 28 with scaling factor 4. (a) Original frame, (b) Tar1 with PSNR = 26.488 dB and SSIM = 0.8545, (c) Tar2 with PSNR = 29.650 dB and SSIM = 0.8875, and (d) Proposed scheme with PSNR = 29.702 dB and SSIM = 0.8913.

4 Conclusion and Future Work

In this paper, a novel reversible data hiding scheme for texture videos and depth maps coding is proposed. At the sender end, the depth video bitstream obtained

by depth down-sampling and compression is embedded in residual coefficients of corresponding texture video. At the receiver end, the depth maps can be retrieved with scalable quality after data extraction, video decoding and texture-based depth reconstruction. The experimental results demonstrate the effectiveness of our proposed scheme. Our future work is to design the rate-distortion optimization criterion for texture videos and depth maps coding using reversible data hiding.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China under Grant 61572452, in part by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDA06030601, and in part by the China Scholarship Council Program under Grant 201506340006.

References

1. Alatan, A.A., Yemez, Y., Gudukbay, U., Zabulis, X., Muller, K., Erdem, C.E., Weigel, C., Smolic, A.: Scene representation technologies for 3DTV-a survey. IEEE Trans. Circ. Syst. Video Technol. **17**(11), 1587–1605 (2007)
2. Fehn, C.: Depth-image-based rendering (DIBR), compression and transmission for a new approach on 3D-TV. In: Proceedings of SPIE Stereoscopic Displays and Virtual Reality Systems XI, pp. 93–104 (2004)
3. Yuan, H., Chang, Y., Huo, J., Yang, F., Lu, Z.: Model-based joint bit allocation between texture videos and depth maps for 3-D video coding. IEEE Trans. Circ. Syst. Video Technol. **21**(4), 485–497 (2011)
4. Kim, W.-S., Ortega, A., Lai, P., Tian, D.: Depth map coding optimization using rendered view distortion for 3D video coding. IEEE Trans. Image Process. **24**(11), 3534–3545 (2015)
5. Shao, F., Lin, W., Jiang, G., Yu, M.: Low-complexity depth coding by depth sensitivity aware rate-distortion optimization. IEEE Trans. Broadcast. **62**(1), 94–102 (2016)
6. Cao, Y., Zhang, H., Zhao, X., Yu, H.: Covert communication by compressed videos exploiting the uncertainty of motion estimation. IEEE Commun. Lett. **19**(2), 203–206 (2015)
7. Khan, A., Mahmood, M.T., Ali, A., Usman, I., Choi, T.-S.: Hiding depth map of an object in its 2D image: reversible watermarking for 3D cameras. In: Proceedings of International Conference on Consumer Electronics, pp. 1–2 (2009)
8. Tong, X., Shen, G., Xuan, G., Li, S., Yang, Z., Li, J., Shi, Y.Q.: Stereo image coding with histogram-pair based reversible data hiding. In: Proceedings of International Workshop on Digital-Forensics and Watermarking, pp. 201–214 (2014)
9. Wang, W., Zhao, J., Tam, W.J., Speranza, F., Wang, Z.: Hiding depth map into stereo image in JPEG format using reversible watermarking. In: Proceedings of International Conference on Internet Multimedia Computing and Service, pp. 82–85 (2011)
10. Tian, J.: Reversible data embedding using a difference expansion. IEEE Trans. Circ. Syst. Video Technol. **13**(8), 890–896 (2003)
11. Jung, S.-W.: Lossless embedding of depth hints in JPEG compressed color images. Sig. Process. **122**, 39–51 (2016)

12. Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A.: Overview of the H.264/AVC video coding standard. IEEE Trans. Circ. Syst. Video Technol. **13**(7), 560–576 (2003)
13. Wang, W., Zhao, J., Tam, W.J., Speranza, F.: Hiding depth information into H.264 compressed video using reversible watermarking. In: Proceedings of ACM Multimedia International Workshop on Cloud-based Multimedia Applications and Services for E-health, pp. 27–31 (2012)
14. Wang, W., Zhao, J.: Hiding depth information in compressed 2D image/video using reversible watermarking. Multimed. Tools Appl. **75**(8), 4285–4303 (2016)
15. Shahid, Z., Puech, W.: Synchronization of texture and depth map by data hiding for 3D H.264 video. In: Proceedings of International Conference on Image Processing, pp. 2773–2776 (2011)
16. Bellifemine, F., Capellino, A., Chimienti, A., Picco, R., Ponti, R.: Statistical analysis of the 2D-DCT coefficients of the differential signal for images. Sig. Process: Image Commun. **4**(6), 477–488 (1992)
17. Gormish, M.J., Gill, J.T.: Computation-rate-distortion in transform coders for image compression. In: Proceedings of SPIE Image and Video Processing, pp. 146–152 (1993)
18. Ni, Z., Shi, Y.-Q., Ansari, N., Su, W.: Reversible data hiding. IEEE Trans. Circ. Syst. Video Technol. **16**(3), 354–362 (2006)
19. Xu, D., Wang, R.: Efficient reversible data hiding in encrypted H.264/AVC videos. J. Electron. Imaging **23**(5), 053022-1–053022-14 (2014)
20. Yoon, K.-J., Kweon, I.S.: Adaptive support-weight approach for correspondence search. IEEE Trans. Pattern Anal. Mach. Intell. **28**(4), 650–656 (2006)
21. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Proceedings of International Conference on Computer Vision, pp. 839–846 (1998)
22. Oh, K.-J., Vetro, A., Ho, Y.-S.: Depth coding using a boundary reconstruction filter for 3-D video systems. IEEE Trans. Circ. Syst. Video Technol. **21**(3), 350–359 (2011)
23. Yao, Y., Zhang, W., Yu, N.: Inter-frame distortion drift analysis for reversible data hiding in encrypted H.264/AVC video bitstreams. Sig. Process. **128**, 531–545 (2016)
24. ITU-T Recommendation: Advanced video coding for generic audiovisual services. ISO/IEC (2012)
25. The H.264/AVC joint model (JM), ver. 10.2. http://iphome.hhi.de/suehring/tml/download/old_jm/
26. Free-viewpoint video sequences. http://www.tanimoto.nuee.nagoya-u.ac.jp/mpeg/mpeg_ftv.html
27. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
28. Wildeboer, M.O., Yendo, T., Tehrani, M.P., Fujii, T., Tanimoto, M.: Color based depth up-sampling for depth compression. In: Proceedings of Picture Coding Symposium, pp. 170–173 (2010)
29. OpenCV, ver. 2.4.9. http://opencv.org/
30. Depth estimation view synthesis software. http://www.tanimoto.nuee.nagoya-u.ac.jp/mpeg/mpeg_ftv.html