Joint Virtual Reality Conference of EuroVR - EGVE (2011)
R. Blach, S. Coquillart, M. D'Cruz, A. Steed, and G. Welch (Editors)

Adapting Standard Video Codecs for Depth Streaming

Fabrizio Pece Jan Kautz Tim Weyrich

{f.pece, j.kautz, t.weyrich}@cs.ucl.ac.uk
Department of Computer Science, University College London, UK

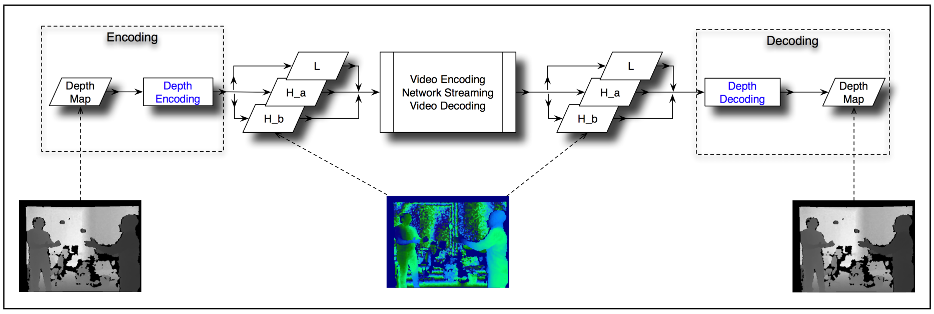


Figure 1: *Graphical overview of the proposed method. The original 16-bit depth map is encoded in an 8-bit, three-channel image and is then processed by a video encoder and transferred over the network. When received, the three-channel image is decoded through the video decoder and is then processed by our method to reconstruct the original 16-bit depth map.*

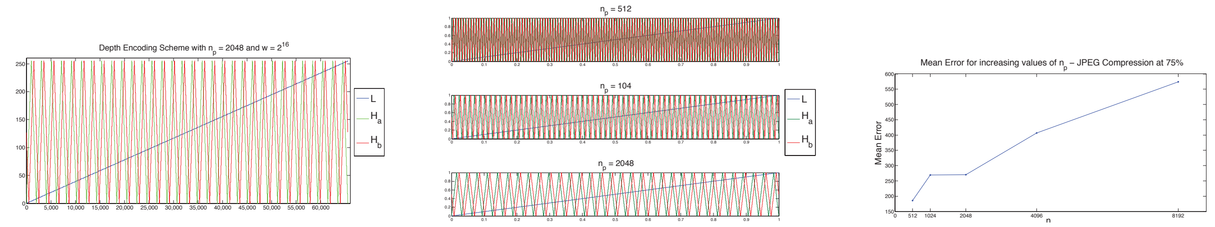
Abstract

Cameras that can acquire a continuous stream of depth images are now commonly available, for instance the Microsoft Kinect. It may seem that one should be able to stream these depth videos using standard video codecs, such as VP8 or H.264. However, the quality degrades considerably as the compression algorithms are geared towards standard three-channel (8-bit) colour video, whereas depth videos are single-channel but have a higher bit depth. We present a novel encoding scheme that efficiently converts the single-channel depth images to standard 8-bit three-channel images, which can then be streamed using standard codecs. Our encoding scheme ensures that the compression affects the depth values as little as possible. We show results obtained using two common video encoders (VP8 and H.264) as well as the results obtained when using JPEG compression. The results indicate that our encoding scheme performs much better than simpler methods.

Categories and Subject Descriptors (according to ACM CCS): I.4.2 [Image Processing and Computer Vision]: Compression (Coding)—Approximate methods

1. Introduction

In the last few years depth acquisition has become a popular topic of research, and this has reflected in a larger availability of depth cameras that allow direct acquisition of scenes' depth information. While there is a large number of applications that can take advantage of this, new problems are introduced. For instance, streaming the information available from depth cameras is a non-trivial task due to the type of data employed by these units (16 bits per depth or higher) and the required bandwidth. While some work has been done to develop ad-hoc depth encoders that allow streaming of 3D content, we have no knowledge of working solutions that adapt existing video encoders (i.e., VP8 or H.264) to depth streaming. Such a solution is highly desirable for applications that are being built today, as special depth compression codecs are not generally available and, consequently, have not been widely adopted. Furthermore, being able to use the same video codec for transferring both colour and depth frames enhances consistency and simplifies the streaming architecture.



(a) $L(d)$ (blue), $H_a(d)$ (green) and $H_b(d)$ (red) with $w = 2^{16}$. For illustration, $n_p = 2048$ is set unusually large, and the ordinate shows integer values rather than [0,1]-normalised values.

(b) Plots of $L(d)$ (blue), $H_a(d)$ (green) and $H_b(d)$ (red) for increasing values of n_p. Here, input depths and output channel values are normalised to lie within $0 \ldots 1$.

(c) Mean Error for increasing n_p values.

Figure 2: *Analysis of the three functions, L, H_a and H_b, used for depth encoding.*

This work presents a novel scheme to encode 16-bit depth maps into three-channel, 8-bit images. Once encoded, the depth maps can be transferred using standard video encoders with no further changes. We show how our scheme successfully encodes and decodes various depth configurations using three different compression schemes, JPEG, VP8 and H.264, at different levels of compression. The proposed technique has been successfully tested on a variety of scenes and depth configurations, and it is now used at the core of an Immersive Collaborative Virtual Environment (ICVE) platform. The applications for the proposed algorithm are numerous: 3D video, Video Mediated Communication (VMC) Systems or ICVEs are some of them.

2. Related Work

Depth streaming is a novel problem with very few, ad-hoc solutions so far. While some work has been done to develop specific depth codecs, the same cannot be said for the task of adapting depth maps to conventional video streaming. Depth streaming is a central topic in free viewpoint video (FVV) and 3D television (3DTV) [KAF*07] applications. An interesting overview of suitable technology for such applications is given by Smolic and Kauff [SK05]. A popular format for 3DTV uses a conventional monoscopic colour video and an associated per pixel depth image corresponding to a single, central viewing position. This format, named "video-plus-depth", has been adopted by the ATTEST system [RdBF*02], one of the first European project that could demonstrate the feasibility of a 3DTV processing chain. By employing such format, the ATTEST system is able to obtain backwards compatibility to existing 2D services for digital video broadcast, efficient compression capabilities and a high adaptability to 3D display properties and viewing conditions [Feh04]. While the monoscopic video stream is encoded with the standard MPEG video coding, the auxiliary depth information is compressed by using an adapted version of the H.264/AVC standard [MWS06]. As a first step towards standardisation of technologies for 3DTV and FVV applications, a new standard addressing algorithms for Multi-view video (MVV) data compression — Multi-view Video Coding (MVC) — has been developed by the Joint Video Team (JVT) of VCEG and MPEG [IMYV07]; however, MVC is intended to encode stereoscopic (two views) images by adapting the H.264 codec [MBX*06], it does not lend itself for direct depth encoding.

Merkle et al. [MMS*09] acknowledge the need of special solutions to enable video codecs, such as H.264, to depth compression. Video codecs are often optimised for image statistics and human perception, and thus a naïve adaption of such codecs to the depth case is not sufficient. In this work, the authors present a different depth-optimised encoding for adaptive pixel blocks that are separated by a single edge, and assign to such block a constant or linear depth approximation. Pajak et al. [PHE*11] present an automatic solution for efficient streaming of frames rendered from a dynamic 3D model. The proposed algorithm is based on an efficient scheme that relies on inter frame prediction, avoiding any future frame prediction. Maitre and Do [MD08] present a different approach based on joint colour/depth compression. The authors exploit the strong correlation between colour and depth to develop an ad-hoc codec that relies on a shape-adaptive wavelet transform and an explicit representation of the locations of major depth edges. However, this solution is limited by its semiautomatic approach. Also region-of-interest specifications and depth-value redistribution can improve depth compression and transmission quality, as showed by Krishnamurthy et al. [CSSH04].

Finally, interesting solutions for depth compression have been developed for telepresence and video-conferencing systems. Lamboray et al. [LWG04] propose a communication framework for distributed real-time 3D video rendering and reconstruction. They introduce several encoding techniques and analyse their behaviour with respect to resolution, bandwidth and inter-frame jitter. Also Würmlin et al. [WLG04] propose a point-based system for real-time 3D reconstruction, rendering and streaming. As their system operates on arbitrary point clouds, no object shape assumptions are made, and topological changes are handled efficiently.

Even if the works presented in this section provide solutions for depth streaming based on already existing codecs (mainly H.264), none of them can be used with the original implementations of such codecs. In fact, they all rely on strong changes on the original video codec and thus on modified implementation. In contrast, our solution can be used with any existing, *unmodified* codec implementation, as it is completely independent from the video encoding technique.

3. Depth Encoding and Decoding

In this section we describe the depth encoding and decoding scheme presented in this work. Our aim is to encode depth maps acquired from depth cameras (i.e., Microsoft Kinect, PMD Camcube or PointGrey Bumblebee), with depths typically described with 16-bit precision, such that they can be streamed using existing video codecs.

Our goal is to reconstruct the original depth values as accurately as possible after compression/decompression. Compression schemes for videos are highly tuned for colour video, taking into account human perception, e.g., by spending fewer bits on colour than luminance information, and so forth. Of course, these insights do not apply to depth compression. On the plus side, video codecs compress 24 bits of data per pixel (3×8 bits), whereas we only have 16 bits per pixel as input. As we will demonstrate in Section 4, naïvely multiplexing the 16-bit depth values into two 8-bit values and passing those into a video codec (leaving the third channel empty) creates severe artefacts; duplicating some of the bits in order to fill the available 3×8 bits does not improve quality much.

We propose a robust encoding of 16-bit depth values into 3×8 bits, such that the decoded depth maps suffer from very few compression artefacts, see Figure 1 for an overview. The scheme is designed to be resilient to quantisation, and comparatively robust against down-sampling (convolution) and altered intensities due to lossy compression.

We express our scheme as a mapping from integer depth values $d \in \{0, \ldots, w-1\}$ ($w = 2^{16}$ for a 16-bit depth map) to three $[0,1]$-normalised (colour) channels $L(d)$, $H_a(d)$ and $H_b(d)$. $L(d)$ is a linear mapping of d into $[0,1]$ and, since subject to quantisation, is interpreted as a low-depth-resolution representation of d,

$$L(d) = (d + {}^{1}\!/\!{}_{2}) \,/\, w \,,$$

while H_a and H_b are chosen as fast-changing, piece-wise linear functions (triangle waves) whose slopes are high enough to be expressed in the low-precision output representation:

$$H_a(d) = \begin{cases} \left(\frac{L(d)}{\frac{p}{2}} \bmod 2\right) & \text{if } \left(\frac{L(d)}{\frac{p}{2}} \bmod 2\right) \leq 1 \\ 2 - \left(\frac{L(d)}{\frac{p}{2}} \bmod 2\right) & \text{otherwise} \end{cases} \,,$$

$$H_b(d) = \begin{cases} \left(\frac{L(d) - \frac{p}{4}}{\frac{p}{2}} \bmod 2\right) & \text{if } \left(\frac{L(d) - \frac{p}{4}}{\frac{p}{2}} \bmod 2\right) \leq 1 \\ 2 - \left(\frac{L(d) - \frac{p}{4}}{\frac{p}{2}} \bmod 2\right) & \text{otherwise}\,. \end{cases}$$

n_p is the integer period of H_a and H_b in the input depth domain and needs to be at most twice the number of output quantisation levels ($n_p \leq 512$ for 8-bit output); $p = \frac{n_p}{w}$ is this period normalised to a $0 \ldots 1$ depth range. Thus designed to be resilient to quantisation, H_a and H_b will be used to decode fine-grain depth variations, while L will anchor these variations in the global depth frame.

In practice, $L(d)$, $H_a(d)$ and $H_b(d)$ can be tabulated for any d in the input depth range, reducing depth encoding to a simple look-up with negligible computational overhead.

As shown in Figure 2(a), H_a and H_b are triangle waves with equal period and different phase. The phases are chosen, so that for any depth value $\bar{d}$ encoded by L, either H_a or H_b is linear within $\bar{d} \pm {}^{p}\!/\!{}_{4}$. Accordingly, given an encoded triple $(\bar{L}, \bar{H}_a, \bar{H}_b)$, the original depth value $\bar{d}$ can be decoded by determining a depth offset L_0 from L and adding a fine-scale depth correction δ:

$$\bar{d}(\bar{L}, \bar{H}_a, \bar{H}_b) = w \cdot \left[L_0(\bar{L}) + \delta(\bar{L}, \bar{H}_a, \bar{H}_b)\right] \,,$$

$$\delta(\bar{L}, \bar{H}_a, \bar{H}_b) = \begin{cases} \frac{p}{2}\bar{H}_a & \text{if } m(\bar{L}) = 0 \\ \frac{p}{2}\bar{H}_b & \text{if } m(\bar{L}) = 1 \\ \frac{p}{2}(1 - \bar{H}_a) & \text{if } m(\bar{L}) = 2 \\ \frac{p}{2}(1 - \bar{H}_b) & \text{if } m(\bar{L}) = 3 \end{cases} \,,$$

with

$$L_0(\bar{L}) = \bar{L} - \left(\bar{L} - \frac{p}{8} \bmod p\right) + \frac{p}{4} m(\bar{L}) - \frac{p}{8} \,,$$

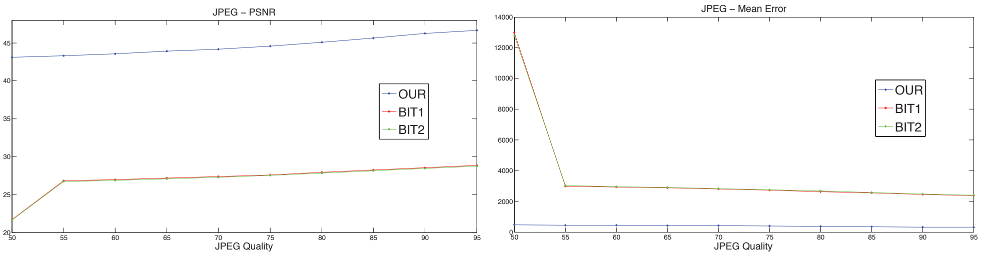
$$m(\bar{L}) = \left\lfloor 4 \frac{L(\bar{d})}{p} - 0.5 \right\rfloor \bmod 4 \,.$$

H_a and H_b are chosen to be triangle waves to be robust against spatial filtering; alternative choices, such as a saw-tooth wave, would have suffered from strong distortions at their discontinuities. While other mappings may still be possible, we argue that C^0 continuity is a desirable property, in particular where the codec downsamples individual colour channels. When increasing n_p above its maximum value, the slopes of $H_a(d)$ and $H_b(d)$ are decreased (Figure 2(b)), gradually becoming subject to quantisation. Figure 2(c) shows how the reconstruction error increases accordingly. For the results shown in this paper we set $n_p = 512$ and $w = 2^{16}$.

On first glance, our code bears resemblance to phase-shift encoding, due to the undulating H_a and H_b with ${}^{\pi}\!/\!{}_{4}$ phase shift. Our decoding scheme, however, does not employ quadrature decoding but rather evaluates only one $H_{\{a,b\}}$ at the time (depending on $L(d)$).

4. Results

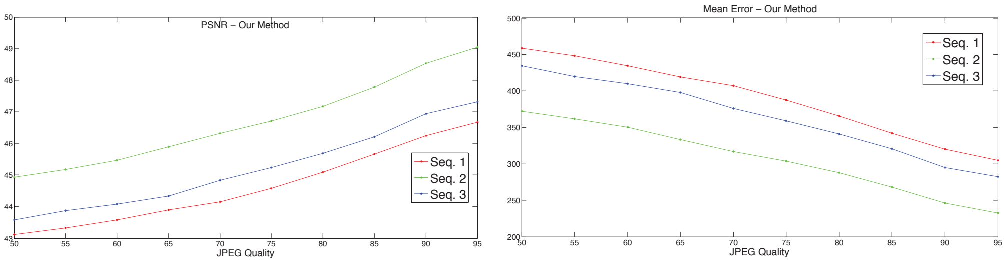
In this section we present the results obtained on a variety of depth-plus-colour videos acquired with a *Microsoft Kinect* unit. We tested three dynamic sequences with a number of frames between 300 and 450 (for each test all the frames have been used to compute the evaluation metrics),



(a) PSNR – JPEG compression

(b) Mean Error – JPEG compression

Figure 4: *Results of the different depth encoding schemes using JPEG compression. Note how our encoding scheme yields a much better PSNR and a much lower mean error. Results are computed on 450 frames with a resolution of 640×480 pixels.*



(a) PSNR of our method – JPEG compression

(b) Mean error of our method – JPEG compression

Figure 5: *Results of our technique using JPEG compression for the three sequences. 300–450 frames, 640×480 pixels.*

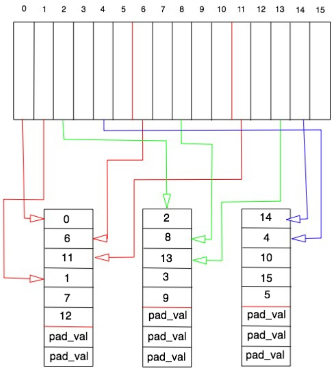
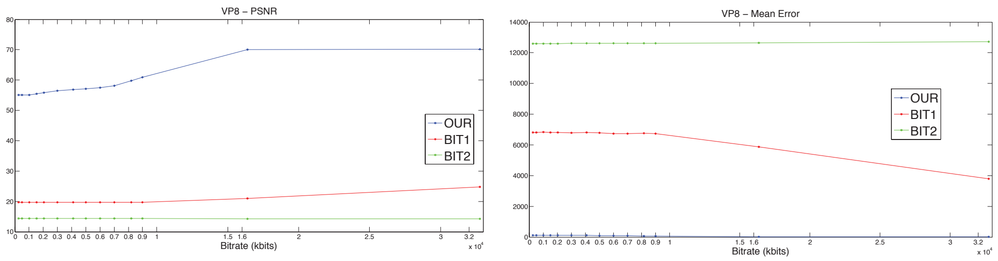


Figure 3: *BIT1 interleaving scheme. Please note that each value in the 8-bit variable cells refers to the corresponding bit index in the 16-bit variable.*

and with a resolution of 640×480 pixels. As quality metrics we decided to compute the Peak-Signal-to-Noise-Ratio (PSNR) and the absolute value of the mean error (ME). To integrate the results analysis we also show point-cloud renderings of the depth maps before and after the transmission. For comparison purpose, we implemented two depth encoding schemes based on "bit multiplexing". In both cases we split the original 16-bit buffer in three chunks with varying sizes, but never bigger than 8 bits, and we then pack them in a three-channel image. In the first case (which we will call BIT1) we interleave the original bit sequence with the scheme shown in Figure 3. For the second case (which we will call BIT2) we store the first six most important bits in the first six most important bits of the first channel, the subsequent five bits in the five most important bits of the second channel, and the final five bits in the five most important bits of the third channel. We then pad the remaining bits with zeros. We decided to employ both JPEG and VP8/H.264 compression to show the results of our encoding scheme with different compression techniques. While JPEG's compression is purely based on the image statistics, VP8 [BWX11] and H.264 [Ric03] encoders take advantage of both temporal and spatial properties of the input sequence.

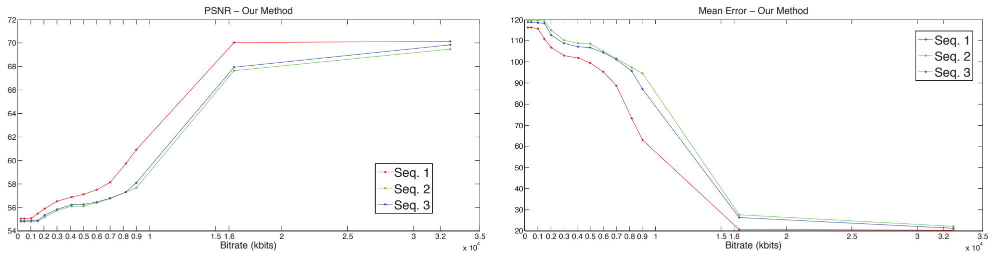
4.1. JPEG Compression

As first test, we combined our depth encoding scheme with the JPEG compression algorithm and compared our solution with the two bit-multiplexing schemes. Hence, we first encoded the video depth maps in an RGB image using either our compression algorithm or one of the bit-multiplexing schemes, then we applied JPEG compression with a certain quality level q, and finally we de-compressed the JPEG image and decoded the resulting RGB into a single-channel, 16-bit map.



(a) PSNR – VP8 compression (b) Mean error – VP8 compression

Figure 6: *Results of the different depth encoding schemes using VP8 compression. Note how our encoding scheme yields a much better PSNR and a much lower mean error. Results computed on 450 frames with a resolution of 640×480 pixels.*



(a) PSNR score of our method – VP8 compression (b) Mean error of our method – VP8 compression

Figure 7: *Results of our technique using VP8 compression for the three sequences. 300–450 frames, 640×480 pixels.*

The result of this test, which we ran on the first video sequence, are shown in Figure 4. The experiment has been conducted with increasing quality for the JPEG compression (quality level of 50 – 95). The performance of the proposed method is clearly superior to the bit-multiplexing schemes. Both PSNR and mean error show how our method is able to compress and decompress the depth range without losing much precision. These results are also supported by the analysis of a point cloud of one of the compressed depth maps. Figure 11 shows the decoded depth maps obtained with the three methods. The depth maps transmitted using our method are superior to the ones obtained with the bit-multiplexing schemes. In fact, while bit multiplexing leads to many grossly corrupted depth values, the quality of the depths obtained with our algorithm compares favourably to the ground truth. These results are confirmed by the tests run on the other two sequences (Figures 5 and 12, second column).

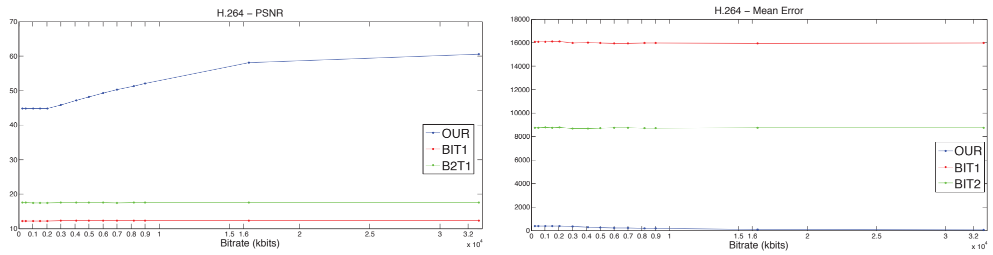
4.2. VP8/H.264 Compression

The tests run on JPEG compression indicate that our depth encoding algorithm generates 3×8-bit images that, when compressed with the JPEG algorithm, do not lose information that will be needed for the reconstruction of the original maps. However, the vast majority of the codecs used for streaming, in contrast to the JPEG standard, are based not only on image statistics, but also on temporal and spatial features. Therefore, we run other tests on our depth encoder (similarly to the ones described in Section 4.1) using two of the most common codecs used for real-time streaming, VP8 and H.264. For these tests, and for both codecs, we have used the codec implementations included in *ffmpeg* (`www.ffmpeg.org`). Both VP8 and H.264 perform a colour-space transformation (RGB to YUV422) before starting the frame encoding, with higher precision in the Y channel. To ensure that the information contained in $L(d)$ is transferred as accurately as possible, we pack the encoded triples $L(d)$, $H_a(d)$ and $H_b(d)$ into Y, U, and V channel, respectively, and feed them directly to the *ffmpeg* encoder. Similarly for the bit-multiplexing techniques, we distribute values over Y, U and V according to their significance. We encoded the depth as the most significant 8 bits in the Y channel, and the remaining bits in the chroma channels.

Note that all codecs considered (including JPEG) down-sample colour information spatially, which is another reason to store data of higher significance in the luminance channel. It further implies that our experiments also test for resilience to (moderate) spatial down-sampling and respective pre-convolution of the chromaticity of the image.

4.2.1. VP8

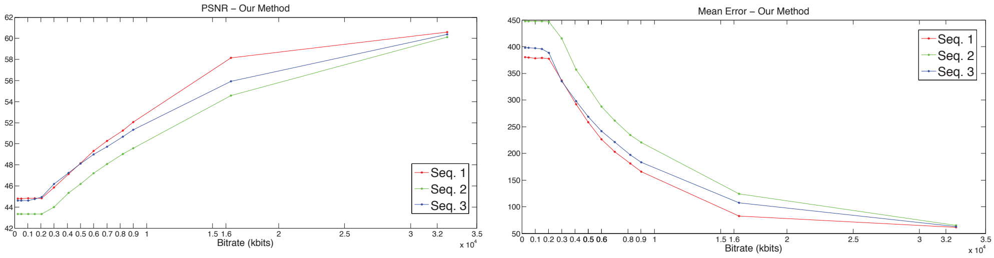
Similarly to what we did for the JPEG case, we run a test on the first of three sequences using our depth encoding scheme and the two bit-multiplexing techniques with VP8 compres-



(a) PSNR – H.264 compression

(b) Mean error – H.264 compression

Figure 8: *Results of the different depth encoding schemes using H.264 compression. Note how our encoding scheme yields a much better PSNR and a much lower mean error. Results computed on 450 frames with a resolution of 640×480 pixels.*



(a) PSNR score of our method – H.264 compression

(b) Mean error of our method – H.264 compression

Figure 9: *Results of our technique using H.264 compression for the three sequences. 300–450 frames, 640×480 pixels.*

sion. Figure 6 shows the results of this initial test. The experiment has been conducted with increasing bit-rate (256 kbit – 32768 kbit) using *ffmpeg* with default parameters. Our compression scheme yields the best performance for both PSNR and mean error, in contrast to the two bit-multiplexing techniques. Moreover, our method generates depth maps that are almost identical to the original ones (Figure 12(c)). Figure 7 shows the performance obtained by our algorithm for the other two video sequences, confirming the results of the previous test. The error introduced by our compression scheme is low, as is also clear from the point clouds showed in the third column of Figure 12.

4.2.2. H.264

As a last test we combined our encoding scheme with the H.264 video compressor. As done in the previous experiments, we run an initial test on the first of three sequences using our depth encoding scheme and the two bit-multiplexing techniques. The results of this experiment (Figure 8) revealed that our technique yields the best performance for both mean error and PSNR. Moreover, the amount of error introduced in the reconstructed maps do no seem to adversely affect the reconstructed depth maps (Figure 12(d)). This is also the case for the last two sequences (Figure 9, Figure 12(h) and Figure 12(l)).

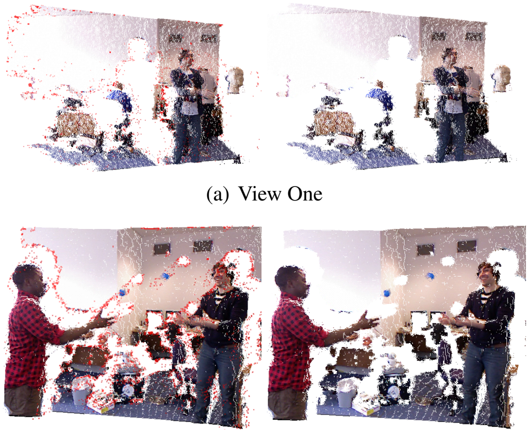
As with VP8, the overall scene's details are well preserved, and the error is mostly located around the edges. From this, we can conclude that our solution can be used successfully with both VP8 and H.264 compression for depth streaming.

4.3. Discussion

The results obtained during our tests show that the proposed solution successfully adapts standard video codecs to depth map streaming. Limited amount of noise is introduced during compression, and the mean error shows that our method affects the depth values very little. The majority of the errors occupies the regions around depth discontinuities. This, however, has been already noticed in previous works [MMS*09, CSSH04, PJO*09, PHE*11], and thus it has to be expected when depth discontinuities are not dealt with separately. These limitations can be partially solved by filtering the decoded depth maps, as shown in Figure 10. Filtering these depth samples (left) based on local point-cloud density helps removing outliers and improves the quality of the reconstruction considerably (right).

5. Conclusion

We presented an efficient solution to adapt video codecs designed for 3×8-bit images to 16-bit depth maps. Our solution requires negligible computational overhead (see Table 1), and works well with several compression algorithms



(a) View One

(b) View Two

Figure 10: *Initial decoded depth map (left) with outliers marked in red. Filtered point cloud of depth samples (right).*

Input resolution	Encoding (ms)	Decoding (ms)
320 × 240	7.9791	10.7116
640 × 480	29.4461	32.7017
1280 × 960	94.6789	106.6898

Table 1: *Computational times of our encoding/decoding scheme on an Intel(R) Core(TM) i7 @ 2.93GHz*

such as JPEG, VP8 and H.264. The proposed method allows the use of the same codec for both colour and depth frames. This simplifies streaming 3D videos, as colour and depth frames can be compressed and transferred using the same video codec, simplifying implementation in applications such as 3D Video, Virtual Environments and Video Mediated Communications. Our scheme is independent of the video codec employed and therefore does not require any modification of the compression algorithm itself (see Figure 1). This distinguishes our work from previous solutions for depth streaming. Finally, our method introduces a small amount of error and noise in the reconstructed depth maps. The vast majority of the noise lies around the depth discontinuities present in the original map, as our method does not explicitly treat them. A post-decoding filtering step seems to be sufficient to remove sporadic noise, however, this solution is semiautomatic and cannot cope with clustered noise. Therefore, a possible extension to our work could be a solution to automatically improve precision around the edges.

References

[BWX11] Bankoski J., Wilkins P., Xu Y.: Technical overview of vp8, an open source video codec for the web. *International Workshop on Acoustics and Video Coding and Communication* (2011). 4

[CSSH04] Chai B.-B., Sethuraman S., Sawhney H. S., Hatrack P.: Depth map compression for real-time view-based rendering. *Pattern Recognition Letters 25* (May 2004), 755–766. 2, 6

[Feh04] Fehn C.: Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* (May 2004), A. J. Woods, J. O. Merritt, S. A. Benton, & M. T. Bolas, (Ed.), vol. 5291 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pp. 93–104. 2

[IMYV07] Ince S., Martinian E., Yea S., Vetro A.: Depth estimation for view synthesis in multiview video coding. *3DTV Conference (3DTV-CON)* (2007). 2

[KAF*07] Kauff P., Atzpadin N., Fehn C., MÂŸller M., Schreer O., Smolic A., Tanger R.: Depth map creation and image-based rendering for advanced 3dtv services providing interoperability and scalability. *Signal Processing: Image Communication 22*, 2 (2007), 217 – 234. Special issue on three-dimensional video and television. 2

[LWG04] Lamboray E., WÃIJrmlin S., Gross M.: Real-time streaming of point-based 3d video. In *In To appear in: Proceedings of IEEE Virtual Reality* (2004), IEEE Computer Society Press, pp. 91–98. 2

[MBX*06] Martinian E., Behrens A., Xin J., Vetro A., Sun H.: Extensions of h.264/avc for multiview video compression. In *IEEE International Conference on Image Processing* (2006). 2

[MD08] Maitre M., Do M. N.: Joint encoding of the depth image based representation using shape-adaptive wavelets. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on* (October 2008), pp. 1768 –1771. 2

[MMS*09] Merkle P., Morvan Y., Smolic A., Farin D., Mueller K., de With P. H. N., Wiegand T.: The effects of multiview depth video compression on multiview rendering. *Singal Processing: Image Communication 24*, 1-2 (2009), 73–88. 2, 6

[MWS06] Marpe D., Wiegand T., Sullivan G. J.: The h.264/mpeg4 advanced video coding standard and its applications. *Communications Magazine, IEEE 44*, 8 (aug. 2006), 134 –143. 2

[PHE*11] Pająk D., Herzog R., Eisemann E., Myszkowski K., Seidel H.-P.: Scalable remote rendering with depth and motion-flow augmented streaming. *Computer Graphics Forum 30*, 2 (2011). Proceedings Eurographics 2011. 2, 6

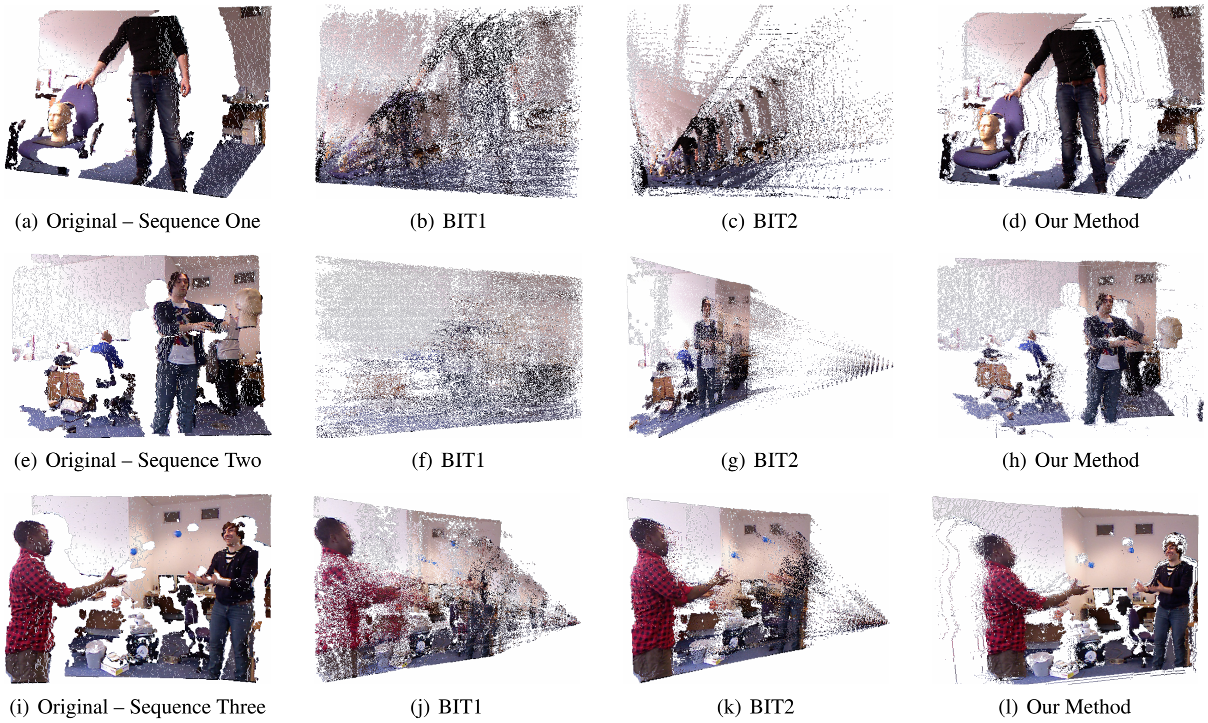
[PJO*09] Park Y. K., Jung K., Oh Y., Lee S., Kim J. K., Lee G., Lee H., Yun K., Hur N., Kim J.: Depth-image-based rendering for 3dtv service over t-dmb. *Signal Processing: Image Communication 24*, 1-2 (2009), 122 – 136. Special issue on advances in three-dimensional television and video. 6

[RdBF*02] Redert A., de Beeck M. O., Fehn C., IJsselsteijn W., Pollefeys M., Gool L. V., Ofek E., Sexton I., Surman P.: Attest: Advanced three-dimensional television system technologies. *3D Data Processing Visualization and Transmission, International Symposium on 0* (2002), 313. 2

[Ric03] Richardson I. E.: *H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia*, 1 ed. Wiley, August 2003. 4

[SK05] Smolic A., Kauff P.: Interactive 3-d video representation and coding technologies. *Proceedings of the IEEE 93*, 1 (January 2005), 98 –110. 2

[WLG04] WüRmlin S., Lamboray E., Gross M. H.: 3d video fragments: dynamic point samples for real-time free-viewpoint video. *Computers Graphics* (2004), 3–14. 2



(a) Original – Sequence One

(b) BIT1

(c) BIT2

(d) Our Method

(e) Original – Sequence Two

(f) BIT1

(g) BIT2

(h) Our Method

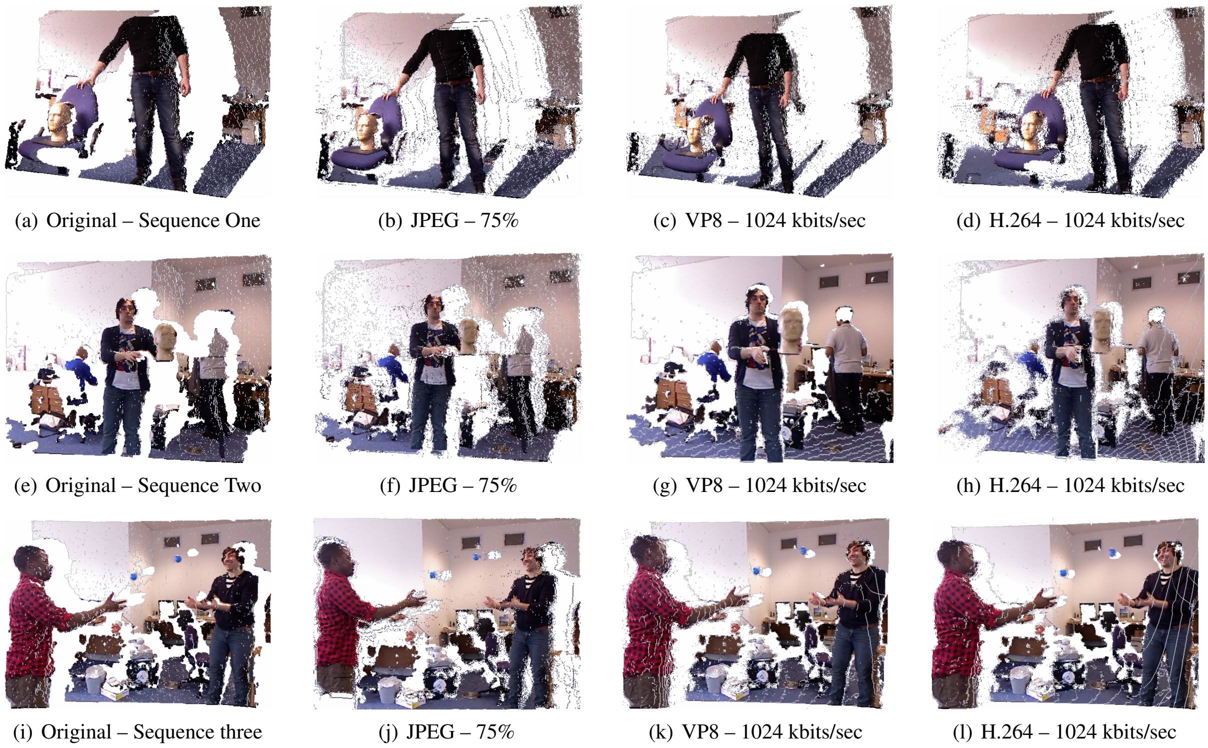
(i) Original – Sequence Three

(j) BIT1

(k) BIT2

(l) Our Method

Figure 11: *Comparison of reconstructed depth maps using different depth coding strategies and JPEG compression (75%).*



(a) Original – Sequence One

(b) JPEG – 75%

(c) VP8 – 1024 kbits/sec

(d) H.264 – 1024 kbits/sec

(e) Original – Sequence Two

(f) JPEG – 75%

(g) VP8 – 1024 kbits/sec

(h) H.264 – 1024 kbits/sec

(i) Original – Sequence three

(j) JPEG – 75%

(k) VP8 – 1024 kbits/sec

(l) H.264 – 1024 kbits/sec

Figure 12: *Depth maps reconstructed using our method. (Point cloud renderings.)*