Low-Complexity, Near-Lossless Coding of Depth Maps from Kinect-Like Depth Cameras

Sanjeev Mehrotra, Zhengyou Zhang, Qin Cai, Cha Zhang, Philip A. Chou

Microsoft Research
Redmond, WA, USA
{sanjeevm,zhang,qincai,chazhang,pachou}@microsoft.com

***Abstract*—Depth cameras are gaining interest rapidly in the market as depth plus RGB is being used for a variety of applications ranging from foreground/background segmentation, face tracking, activity detection, and free viewpoint video rendering. In this paper, we present a low-complexity, near-lossless codec for coding depth maps. This coding requires no buffering of video frames, is table-less, can encode or decode a frame in close to 5ms with little code optimization, and provides between 7:1 to 16:1 compression ratio for near-lossless coding of 16-bit depth maps generated by the Kinect camera.**

I. Introduction

Over the past few years, depth cameras have been rapidly gaining popularity with devices such as Kinect becoming popular in the consumer space as well as the research space. Depth information of a scene can be used for a variety of purposes and can be used on its own or in conjunction with texture data from a RGB camera. It can be used for aiding foreground/background segmentation, performing activity detection, face tracking, pose tracking, skeletal tracking, and for free viewpoint video rendering [1], [2], [3], needed for immersive conferencing and entertainment scenarios.

Depth cameras primarily operate using one of two technologies, one is by computing the time of flight from light being emitted and the other is by observing the actual pattern that gets displayed when a known infrared pattern is projected onto the scene. The second of these two technologies requires an infrared projecter to project a known pattern onto the scene and an infrared camera to read the projected pattern. It is known that in the second of these two technologies, such as that used in the Microsoft Kinect sensors, the accuracy of the camera actually reduces with the inverse of the depth. For example, the error in the actual depth when reading a depth value which is twice as far as another one will be two to four times as large. This can be used when coding depth maps to lower the bitrate while maintaining the full fidelity of the sensor. Stereovision systems, usually using two or three video cameras [4], produce depth maps with the depth accuracy also decreasing with the inverse of the depth. Therefore, the compression technique proposed in this paper can also be applied to depth maps from stereovision systems.

In addition, depth maps typically have only a few values in a typical scene. For example in a typical scene with a background and a single foreground object, the background has no variation in depth. In fact the variation is much less than that found in texture map of the background. Even the foreground has less variation in depth when going from one pixel to the next as a greater degree of continuity naturally exists in depth when compared to texture. For example, if we look at ones clothing, the clothing itself may have significant variation in texture due to the design, but the depth will still be relatively continuous as it is still physically the same object. This fact can be used to predict the depth of a given pixel from neighboring pixels. After this prediction, simple run-length/level techniques can be used to further perform entropy coding allowing for high compression ratios.

The coding of depth maps is a relatively recent topic of interest to MPEG [5]. In this paper, we do not attempt to come up with a relatively complicated lossy compression scheme, but rather present a *low-complexity*, and *near-lossless* compression scheme for depth maps which gives a high compression ratio between 7 (2.3bpp) to 16 (1bpp) depending on the content. This coding can be used as front-end to other compression schemes, or can be used on its own for compressing depth maps. For example, it can be implemented directly on the camera and used for sending the depth map to the computer. This bitrate reduction can allow for multiple depth cameras (depth camera array) to be put onto a single USB interface or allow for higher resolution depth cameras to be used on a single USB interface.

Alternate methods for depth map encodings have been proposed [6], [7], [8]. Several methods use triangular mesh representations for the depth map. Extracting the mesh from the raw depth map takes additional complexity which we avoid by directly coding the depth map using the pixel representation. In addition, even using a mesh representation does not perform better than JPEG 2000 encoding [8], whereas our method does perform better for lossless coding. In addition, we concentrate on a lossless to near-loss encoding as opposed to a lossy encoding.

II. Depth Map Coding

An example depth map from a scene as captured from the depth camera on Microsoft's Kinect is shown in Fig. 1. Regardless of the texture, we can see that the depth map captures the outlines of the objects in the scene. It can be seen from the depth map that there are only a few distinct values in the depth map and thus we can infer that high compression ratios should be possible. In this paper, we present a low

Fig. 1. An example depth map from Set 1.

complexity, near-lossless compression scheme which consists of three components which will be described in greater detail.

1) Inverse coding of the depth map: As the depth map accuracy from the sensor decreases with distance, we can quantize depth values which are further out more.
2) Prediction: Since the depth map is relatively continuous from pixel to pixel simple prediction schemes can be applied to remove redundancy.
3) Adaptive Run-length / Golomb-Rice (RLGR) coding [9]: After prediction, there will typically be large runs of zeros. Thus we can use adaptive RLGR as an adaptive, low-complexity, table-less code to perform entropy coding.

A. Inverse Coding of Depth Map

It is well known that the sensor accuracy of depth cameras such as the Kinect has an accuracy which is at least inversely proportional to the actual depth — Kinect's depth accuracy is actually proportional to the inverse of *squared* depth. Let Z be the depth value produced by the camera. We define $Z_{\min}$ to be the minimum depth that can be produced by the sensor of the depth camera and let $Z_{\max}$ be the maximum depth. We define Z_0 to be the depth at which the accuracy is 1 unit. This value can be found by looking at the sensor specifications of the manufacturer or can be measured by placing objects at known depths and repeatedly measuring the depth values using the camera. Therefore, at distances less than Z_0, the depth map coding does not fully encode the sensor capabilities and at distances greater than Z_0, the sensor accuracy decreases with depth. For example, at $Z = 2Z_0$, the sensor accuracy is 2 units rather than 1 (if inversely proportional) or 4 units (if inversely proportional to the square). and therefore we can quantize all values $2Z_0 - 1$, $2Z_0$, and $2Z_0 + 1$ all to $2Z_0$ and still be encoding the full fidelity of the sensor.

As an alternative to non-uniform quantization, we can use something similar to that proposed in [1], and simply code the inverse of the depth (call it D),

$$D = \frac{a}{Z} + b, \tag{1}$$

where a is chosen to maintain full depth accuracy. That is

$$\begin{aligned} \frac{a}{Z_0} - \frac{a}{Z_0+1} &\geq 1, & (2) \\ a &\geq Z_0(Z_0+1). & (3) \end{aligned}$$

To minimize the number of bits in the coding while maintaining the encoding capabilities of the sensor, we can simply use $a = Z_0(Z_0+1)$. b is an arbitrary offset and can be determined by setting $D = 1$ at $Z = Z_{\max}$, which gives

$$b = 1 - \frac{a}{Z_{\max}}. \tag{4}$$

$Z = 0$ can be used as a special value to indicate missing depth values and can be coded as $D = 0$.

Decoding can simply invert Eqn. 1, and we get

$$Z = \frac{a}{D - b}. \tag{5}$$

Note that although the decoded depth value may not be the same as the original depth value, it is still "lossless" in that little to no additional noise is being introduced than that already present due to the sensor accuracy.

The dynamic range of the coefficients after applying the inverse mapping is given by

$$\Delta D = a\left(\frac{1}{Z_{\min}} - \frac{1}{Z_{\max}}\right) = \frac{a(Z_{\max} - Z_{\min})}{Z_{\min}Z_{\max}}. \tag{6}$$

If $a < Z_{\min}Z_{\max}$, then we see that the dynamic range of the coefficients after applying the inverse mapping is reduced, and thus even in the absence of other entropy coding schemes, there is a bit rate reduction of $\log_2 \frac{Z_{\min}Z_{\max}}{a}$ bits needed to represent the depth map values.

B. Prediction

D is coded in the integer domain, that is we simply take the round(.) of Eqn. 1. To take advantage of the continuity of the depth map, D is predicted using the previous neighboring value. We simply raster scan the depth map to create a one-dimensional array. Other scanning methods such as the well known zig-zag scan are also possible, although the raster scan takes advantage of cache locality and is thus faster in implementation, and also gets most of the prediction gains. After raster scan, the nth pixel gets coded using

$$C_n = D_n - D_{n-1}. \tag{7}$$

More complicated schemes can also be used for prediction such as adaptive linear prediction techniques and various directional prediction schemes as used in the video coding literature. However, again, for purposes of simplicity and since we already get good compression efficiency, we choose to just use a one step prediction.

For low memory usage, we also choose to only do spatial prediction and do not rely on any temporal or motion prediction schemes. Thus we do not require any additional frame buffer for encoding or decoding. Because there is significant spatial continuity in the depth map, additional temporal prediction will provide additional gain primarily at the boundaries of the objects. The pixels internal to the objects is already being predicted fairly well.

C. Adaptive RLGR Coding

We use the adaptive Run-Length / Golomb-Rice code from [9] to code the resulting C_n values. Since this entropy coding method uses a backward adaptive method for parameter adaptation, no additional side information needs to be sent, and no tables are needed to perform the encoding or decoding. Although the original depth map, Z, and the inverse depth map, D, consist of positive values, the predicted values, C, can be negative or positive. If we assume that C_n is distributed according to a Laplacian distribution, then, we can apply the interleave mapping to get a source with exponential distribution as needed by Golomb-Rice code,

$$F_n = \begin{cases} 2C_n & \text{if } C_n \geq 0 \\ -2C_n - 1 & \text{if } C_n < 0 \end{cases} . \tag{8}$$

After this, the codec operates in either the "no-run" mode or the "run" mode. There are two parameters in the codec which are adapted, k and k_R. In the no-run mode ($k = 0$), each symbol F_n gets coded using a Golomb-Rice coding, represented as $GR(F_n, k_R)$. In the the run mode ($k \neq 0$), a run of $m = 2^k$ zeros gets coded using a 0, and a run of $m < 2^k$ symbols gets coded using 1 followed by the binary representation of m using k bits followed by the level $GR(F_n, k_R)$.

The Golomb-Rice coding is simply given by

$$GR(F_n, k_R) = \underbrace{11\ldots1}_{p\text{-bit prefix}} \; \underbrace{b_{k_R-1}b_{k_R-2}\ldots b_0}_{k_R\text{-bit suffix}}, \tag{9}$$

where there are $p = \lfloor \frac{u}{2^{k_R}} \rfloor$ prefix bits and the suffix value is the remainder of $\frac{u}{2^{k_R}}$.

Instead of keeping k and k_R fixed parameters, they are adapted using the backward adaptation similar to that described in [9]. Although the adaptive RLGR coding on its own is table-less, we have optimized the decoding using a trivial amount of memory of 3KB. With minimal optimizations, we can easily encode a 640x480 16-bit per pixel frame in close to 5ms on a single-core 3GHz core.

D. Lossy Coding

The proposed coding scheme is numerically lossless if only the prediction and adaptive RLGR entropy coding is performed. It is also practically lossless, because of the sensor accuracy, if the inverse depth coding is done using $a = Z_0(Z_0 + 1)$. To introduce some small amount of loss, we can choose one of three methods

1) $a < Z_0(Z_0 + 1)$ can be chosen. This will introduce loss in that the sensor capabilities will not be fully encoded.
2) C_n can be mildly quantized. In this case the coefficient that gets coded becomes $C_n = Q(D_n - D'_{n-1})$, where $Q(.)$ is the quantization operation, and D'_{n-1} is the previous decoded pixel. The decoded pixel can be found by $D'_n = IQ(C_n) + D'_{n-1}$, where $IQ(.)$ is the inverse quantization operation.
3) The level in the adaptive RLGR coding can be quantized.

In this paper, we explore lossy coding using the first of these three options, that is choosing $a < Z_0(Z_0 + 1)$.

Lossy coding can also be done using more complex methods and different schemes may be used depending on the actual application of the depth map. For example, if the coded depth map is being used to reconstruct different views of a video scene, then the criteria to be optimized is the actual video reconstruction and not the accuracy of the coded depth map.

III. Results

We present two sets of results, one is to show the computational complexity and compression efficiency of the proposed depth map coding with an implementation done using minimal optimizations. The other set of results compares the reconstruction of video from alternate viewpoints when using the texture map and a *coded* depth map from a given viewpoint. The depth map is coded using near-lossless and slightly lossy compression using the method proposed.

A. Coding Efficiency

We compress two sets of captured depth maps from the Kinect camera. These two sets of depth maps are taken from multiple video sequences, the total consisting of close to 165 seconds of video and depth maps captured at 30fps, 640x480, and 16 bits/pixel (bpp). The first of these two sets is a more complicated depth map (Fig. 1 is a depth map from this set) with multiple depth values, the second is a relatively simpler depth map consisting of only a single background and foreground object (Fig. 5(b)).

The coding is done using the following five methods. In the Microsoft Kinect sensors, we measure and use the following parameters for the inverse coding presented in Sec. II-A, $Z_{\min} = 300mm$, $Z_0 = 750mm$, and $Z_{\max} = 10000mm$.

1) Without inverse coding: This method just uses the entropy coding (prediction plus adaptive RLGR coding) and produces numerically lossless results.
2) With inverse coding (lossless): This method uses the inverse coding method with $Z_0 = 750mm$. This value of Z_0 is the measured depth value at which the sensor accuracy is 1 unit. Thus, this produces results which fully encode the sensor capabilities.
3) With inverse coding (lossy): We reduce Z_0 to $Z_0 = 600mm$. This induces mild compression, with a quantization factor of 1.25.
4) With inverse coding (lossy): We further reduce to $Z_0 = 300mm$. This results in stronger compression as the depth values are quantized by a factor 2.5.
5) With JPEG 2000 [10] lossless mode encoding. We use this state of the art lossless image codec to compare the efficiency of our compression scheme.

The coding efficiency results for these two sets of depth maps are summarized in Table I when using methods 1 to 4. In Table. I, we list the 10-th percentile, 50-th percentile (median), and 90-th percentile results for the compression ratio per frame, encode time per frame, and decode time per frame. We do not show results for the *mean* compression ratio as they

TABLE I

SUMMARY OF COMPRESSION RATIO AND CPU UTILIZATION RESULTS FOR BOTH WITHOUT AND WITH INVERSE CODING. THIS TABLE SHOWS THE 10-TH PERCENTILE, 50-TH PERCENTILE (MEDIAN), 90-TH PERCENTILE VALUES FOR COMPRESSION RATIO (CR), ENCODE TIME (LISTED AS "ENC") PER FRAME, AND DECODE TIME (LISTED AS "DEC") PER FRAME. WITHOUT INV. CODING IS NUMERICALLY LOSSLESS, $Z_0 = 750mm$ IS NEARLY LOSSLESS BECAUSE OF THE SENSOR ACCURACY, $Z_0 = 600mm$ AND $Z_0 = 300mm$ ARE LOSSY DUE TO ADDITIONAL QUANTIZATION OF DEPTH VALUES BY A FACTOR OF 1.25 AND 2.50 RESPECTIVELY.

	Set1				Set2			
	W/O Inv. Coding	Inv. Coding $Z_0 = 750mm$	Inv. Coding $Z_0 = 600mm$	Inv. Coding $Z_0 = 300mm$	W/O Inv. Coding	Inv. Coding $Z_0 = 750mm$	Inv. Coding $Z_0 = 600mm$	Inv. Coding $Z_0 = 300mm$
10-th% CR	4.4279	5.7479	6.3408	13.3827	12.6955	15.2030	16.6360	33.1284
50-th% CR	4.8855	6.1754	6.8012	15.1446	12.9683	15.4539	16.8819	34.1922
90-th% CR	5.2915	7.2276	7.9825	16.6762	13.5149	16.0821	17.5864	35.8543
10-th% Enc	5.213ms	6.648ms	6.585ms	4.728ms	3.186ms	3.797ms	3.753ms	2.909ms
50-th% Enc	5.833ms	7.429ms	7.415ms	5.177ms	3.523ms	3.901ms	3.857ms	2.961ms
90-th% Enc	6.472ms	7.900ms	7.966ms	5.553ms	4.109ms	3.978ms	4.138ms	3.010ms
10-th% Dec	7.027ms	7.871ms	8.196ms	4.763ms	3.538ms	3.943ms	3.911ms	2.550ms
50-th% Dec	7.862ms	9.097ms	9.248ms	5.362ms	3.883ms	4.036ms	4.013ms	2.601ms
90-th% Dec	8.831ms	9.704ms	10.047ms	5.813ms	4.590ms	4.210ms	4.470ms	2.698ms

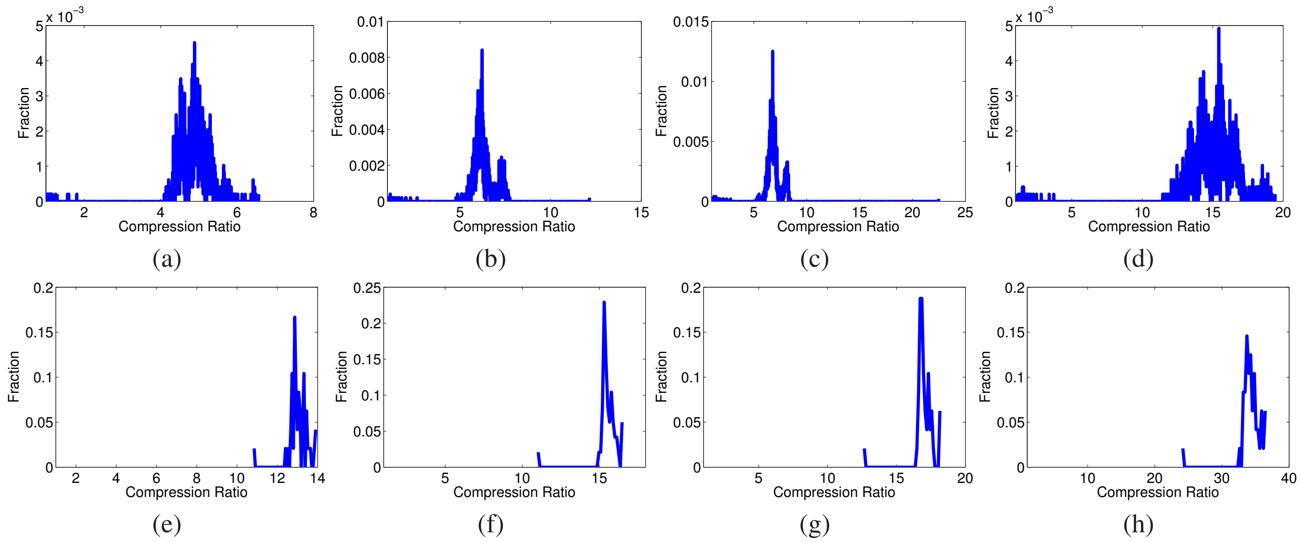


Fig. 2. PDF of compression ratios of two sets of depth maps. (a) Numerically lossless coding (without inverse coding), (b) Lossless coding of sensor capabilities ($Z_0 = 750mm$), (c) Lossy coding with additional quantization factor of 1.25 ($Z_0 = 600mm$), (d) Lossy coding with additional quantization factor of 2.5 ($Z_0 = 300mm$). (a)-(d) are results for Set 1 and (e)-(h) are for Set 2.

get skewed towards the high side since some frames (with a constant depth value) get compressed to very few bytes (as little as 5 bytes) resulting in very high compression ratios.

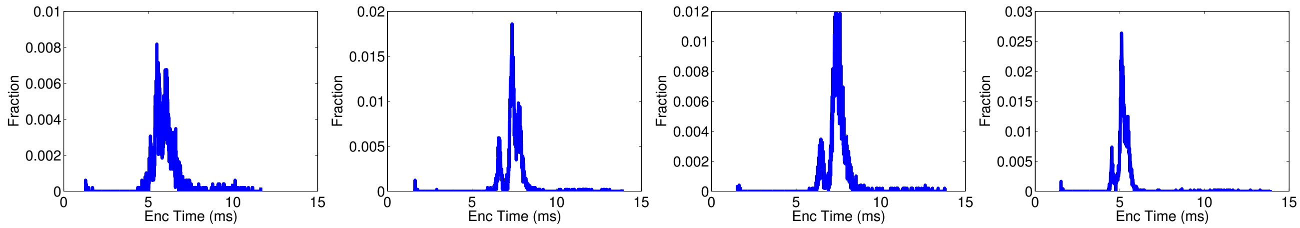
For the first set, we see that we can achieve up to a ratio of 5.3 for numerically lossless compression, and 7.2 (2.2bpp) for near-lossless compression (lossless up to the sensor accuracy) by using inverse coding. For the second set (simpler set), we can achieve lossless compression ratios of up to 16 (1 bpp). The higher compression ratio for the second set is due to the simplicity of the scene where most of the background is out of sensor range and thus does not have a valid depth value. This can be seen by comparing the depth map in Fig. 1 (from the first set) with the one in Fig. 5 (from the second set). If lossy coding is allowed, with $Z_0 = 300mm$, we achieve compression ratios of 15 (for the first set) up to 30 (for the second set) resulting in 0.5-1bpp. The histogram (PDF) of the compression ratio for the first four coding methods are also plotted in Fig. 2 for the two data sets.

For comparison, JPEG 2000 [10] lossless mode encoding gives 10%, 50%, and 90% compression ratios of 3.52, 3.73, and 3.98 respectively for Set 1, and 9.37, 9.70, and 10.18 for Set 2. This makes our lossless compression efficiency – even without inverse coding – more than 30% better (see Table I) than the JPEG 2000 lossless codec. Additionally, the encoding and decoding complexity for JPEG 2000 is noticeably higher.

B. Computational Complexity

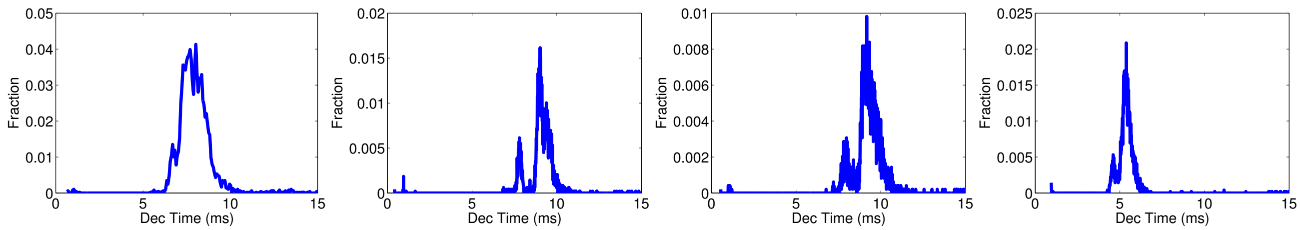
The implementation is done in pure C-code on a PC (no assembly code written). We measure the encode/decode times per frame when running on a single core 3GHz PC. The results are summarized in Table I, showing the 10-th%, 50-th% (median), and 90-th% encode and decode times. The PDF of the encode and decode times are plotted in Fig. 3 and Fig. 4 respectively for the first data set.

From the results, we see that for we can simultaneously encode and decode close to 70fps-200fps (encode plus decode time is between 5ms-15ms). The encode and decode time with inverse coding is somewhat higher since there is a divide that



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

Fig. 3. PDF of encode time per frame for the first data set. (a) Numerically lossless coding (without inverse coding), (b) Lossless coding of sensor capabilities ($Z_0 = 750mm$), (c) Lossy coding with additional quantization factor of 1.25 ($Z_0 = 600mm$), (d) Lossy coding with additional quantization factor of 2.5 ($Z_0 = 300mm$).



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

Fig. 4. PDF of decode time per frame for the first data set. (a) Numerically lossless coding, (b) Lossless coding of sensor capabilities ($Z_0 = 750mm$), (c) Lossy coding with $Z_0 = 600mm$, (d) Lossy coding with $Z_0 = 300mm$.

takes place, but this can easily be reduced by using a look up table to perform the division. Other optimizations can easily further reduce the computational cost so that it can even be directly implemented on the camera itself.

C. Viewpoint Reconstruction using Coded Depth Map

We also use the coded depth map and texture map to reconstruct alternate viewpoints to measure the effect of coding on the reconstruction. Let viewpoint A be the viewpoint from which we capture video and depth which is used to reconstruct an alternate viewpoint, B. As an example, the texture map and depth map from viewpoint A are shown in Fig. 5(a) and (b). The reconstructed video from viewpoint B using the uncoded depth map and the coded depth map (using $Z_0 = 750mm$) is shown in Fig. 5(d) and (f) respectively. The coded depth map is shown in Fig. 5(e). For comparison, the actual video captured from viewpoint B is shown in Fig. 5(c).

The video used in this experiment consists of 47 frames. To measure the accuracy of the coded depth maps, we reconstruct video from five alternate viewpoints using the texture and depth map from viewpoint A. At the same time, we also capture video directly from these five alternate viewpoints. The cameras at these five alternate viewpoints (viewpoint B) are calibrated to the camera at viewpoint A, i.e. we know the translation and rotation.

We measure the PSNR between the captured video and the reconstructed video. These results are shown in Fig. 6(a) for the four different encoding methods explained before — numerically lossless, sensor-wise lossless ($Z_0 = 750mm$), and lossy using $Z_0 = 600mm$ and $300mm$. The five alternate viewpoint PSNR results (shown for the luminance component (Y) only) are all shown together in the Figure (each viewpoint consists of 47 frames in the figure). The best viewpoint reconstruction results in a PSNR of about 21dB and the worst is about 14.5dB. However, we note that regardless of which method is being used to encode the depth map, the PSNR results are all very similar. Thus, even mildly lossy compression (with $Z_0 = 300mm$) gives a bitrate reduction of 15 to 30 times, but produces no *additional* artifacts in reconstruction when compared to the actual video. That is most of the additional errors introduced by the mild quantization of the depth map are masked by the errors introduced by reconstruction inaccuracies when compared to the actual captured video. Only for the fifth viewpoint do we see additional errors introduced by quantization of the depth map of about 0.5dB in PSNR.

In Fig. 6(b), we show the PSNR results between the numerically lossless video reconstruction and the reconstructions with the inverse coding for the various Z_0 values. This result shows the reconstruction errors introduced by the depth map quantization when comparing to the reconstruction obtained using an *unquantized depth map* as opposed to the *actual captured video* as in Fig. 6(a). We see that $Z_0 = 750mm$ produces a reconstruction which is very close to that obtained when using the numerically lossless depth map (PSNR $>$ 40dB). Even the $Z_0 = 300mm$ encoding produces good results with PSNR close to 32dB.

IV. Conclusion

In this paper, we have presented a low-complexity, high-compression efficiency, near-lossless coding of depth maps. The coding takes advantage of the sensor accuracy, and continuity in the depth map to encode. The encoding consists of three components, inverse depth coding, prediction, and adaptive RLGR coding. The coding provides a high compression ratio even for a near-lossless representation of the depth map. In addition, reconstruction of alternative viewpoints from the coded depth map does not result in additional artifacts. It can be implemented as either a front-end to more complicated

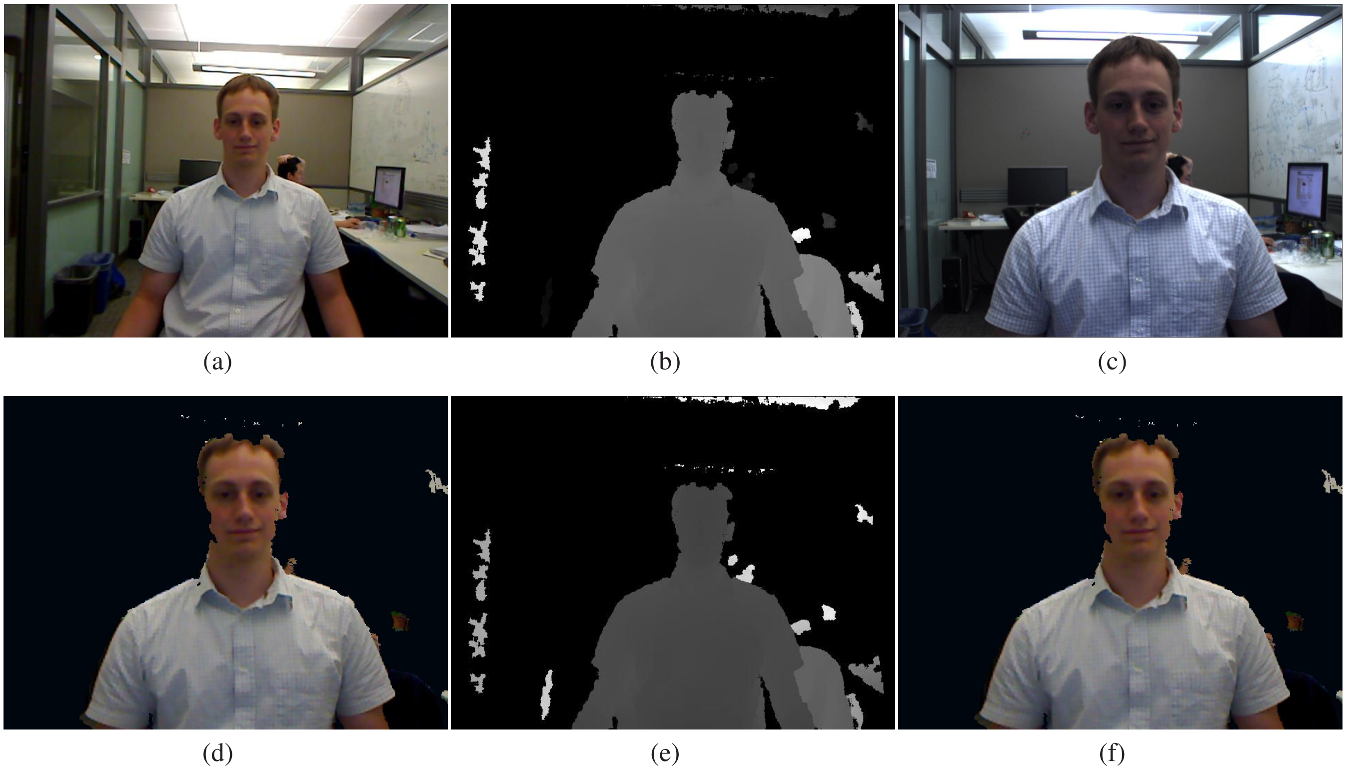


Fig. 5. Images from Set 2. (a) The original texture map from viewpoint A. (b) The original depth map from viewpoint A. (c) The original video captured from viewpoint B. (d) The reconstructed video from viewpoint B using texture map and uncoded (lossless) depth map from viewpoint A. (e) The coded depth map using $Z_0 = 750mm$, (f) The reconstructed video from viewpoint B using texture map and coded (using $Z_0 = 750mm$) depth map from viewpoint A.

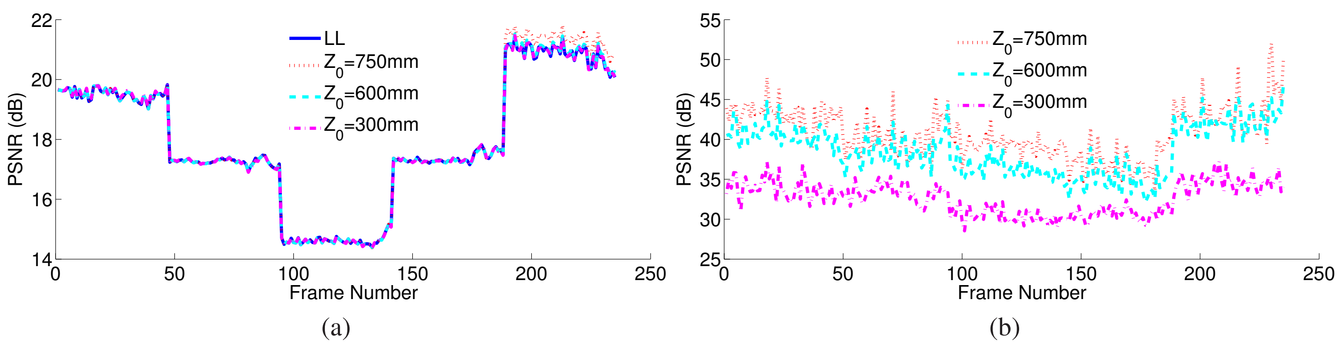


Fig. 6. PSNR Results. (a) PSNR between captured video and video reconstructed using texture map plus depth map. The depth map is coded using four different ways, "LL" stands for numerically lossless (without inverse coding), $Z_0 = 750mm$ is lossless encoding of sensor capabilities, $Z_0 = 600mm$, and $Z_0 = 300mm$ are lossy encodings. (b) PSNR between video reconstruction using original (uncoded) depth map and video reconstruction using coded depth map for various Z_0 values. The five viewpoints are shown back to back for the 47 frame sequence (frames 1-47 represent the first viewpoint, frames 48-94 the second viewpoint, and so on).

lossy compression algorithms, or can be used on its own, allowing multiple high-definition depth camera arrays to be connected on a single USB interface.

References

[1] K. Müller, P. Merkle, and T. Wiegand, "3-d video representation using depth maps," *Proceedings of the IEEE*, vol. 99, no. 4, pp. 643–656, Apr. 2011.

[2] B.-B. Chai, S. Sethuraman, and H. Sawhney, "A depth map representation for real-time transmission and view-based rendering of a dynamic 3d scene," in *Proc. First International Symposium on 3D Data Processing Visualization and Transmission*, 2002, pp. 107–114.

[3] A. Smolic, K. Mueller, P. Merkle, P. Kauff, and T. Wiegand, "An overview of available and emerging 3d video formats and depth enhanced stereo as efficient generic solution," in *Proc. Picture Coding Symposium*, May 2009, pp. 1–4.

[4] O. Faugeras, *Three Dimensional Computer Vision*. MIT Press, 1993.

[5] *Overview of 3D Video Coding*, Apr. 2008, ISO/IEC JTC1/SC29/WG11, Doc. N9784.

[6] S.-Y. Kim and Y.-S. Ho, "Mesh-based depth coding for 3d video using hierarchical decomposition of depth maps," in *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, vol. 5, Oct. 2007.

[7] K.-J. Oh, A. Vetro, and Y.-S. Ho, "Depth coding using a boundary reconstruction filter for 3-d video systems," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 21, Mar. 2011.

[8] B.-B. Chai, S. Sethuraman, H. S. Sawhney, and P. Hatrack, "Depth map compression for real-time view-based rendering," *Pattern Recognition Letters, Elsevier*, pp. 755–766, May 2004.

[9] H. S. Malvar, "Adaptive Run-Length / Golomb-Rice encoding of quantized generalized Gaussian sources with unknown statistics," in *Data Compression Conference*, 2006, pp. 23–32.

[10] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The jpeg 2000 still image compression standard," *Signal Processing Magazine, IEEE*, vol. 18, no. 5, pp. 36 –58, Sep. 2001.