Compression of the Layered Depth Image

Jiangang Duan and Jin Li, *Senior Member, IEEE*

***Abstract*—A layered depth image (LDI) is a new popular representation and rendering method for objects with complex geometries. Similar to a two-dimensional (2-D) image, the LDI consists of an array of pixels. However, unlike the 2-D image, an LDI pixel has depth information, and there are multiple layers at a pixel location. In this paper, we develop a novel LDI compression algorithm that handles the multiple layers and the depth coding. The algorithm records the number of LDI layers at each pixel location, and compresses LDI color and depth components separately. For LDI layer with sparse pixels, the data is aggregated and then encoded. An empirical rate-distortion model is used to optimally allocate bits among different components. Compared with the benchmark compression tools such as JPEG-2000 and MPEG-4, our scheme improves the compression performance significantly.**

***Index Terms*—Data compression, layered depth image (LDI), virtual reality, wavelet.**

I. Introduction

A layered depth image (LDI) [1], [2] is a new representation and rendering method for objects with complex geometries. Instead of representing objects with triangular meshes like most models in the computer graphics, LDI represents the object with an array of pixels viewed from a single camera location. Each LDI pixel is represented by its color, depth that is the distance of the pixel to the camera, and a few other properties assisting LDI rendering. There may be multiple pixels along each line of sight, therefore, the LDI usually consists of multiple layers. LDI enables the rendering of virtual views of the object/scene at new camera position, and the rendering operation can be performed quickly with the warp-order algorithm [3]–[5] proposed by McMillian. The complexity of rendering an LDI view is only related to the number of pixels in the LDI, i.e., to the resolution of the view, and is not directly related to the scene complexity. Therefore, realistic object/scene with huge polygon counts can be rendered very quickly with the LDI. The LDI can also be relatively easily constructed. By using computer vision techniques to recover depth information from multiple photo shots, a still image is turned into an LDI, which can be used to render virtual views at novel camera positions.

A high resolution LDI can contain large amount of data. As an example, a feature rich LDI scene Cathedral [Fig. 7(c)] occupies a total of 14.1 megabytes (MB). It is of resolution 1024×1024, and consists of 1 588 812 depth pixels. A realistic scene calls for even higher sampling density, and therefore, results in even more data.

Manuscript received September 21, 2001; revised October 28, 2002. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. A. Enis Cetin.

The authors are with the Signal Processing Group, Microsoft Research, Redmond, WA 98052 USA (jinl@microsoft.com).

Digital Object Identifier 10.1109/TIP.2003.809010

Generic lossless compression tools, such as WinZip, reduce the data amount of the LDI without sacrificing the quality. However, the lossless compression ratio is usually too small (around 2–3:1). Although there are a variety of lossy two-dimensional (2-D) image compression algorithms and schemes to compress the depth map, such as the one proposed by Krishnamurthy *et al.* [7] with JPEG 2000 method, they cannot be directly applied to compress the LDI image. This is due to the fact that the LDI not only contains luminance and chrominance components, but also contains elements for 3-D rendering, such as the depth information. Moreover, there are multiple layers at each pixel location, and data in most back layers are not of rectangular region of support. The first few layers may be dense, but the data in the back layer is usually sparse. Although algorithms, such as MPEG-4 [8], have been developed to encode object of nonrectangular shape, they are not able to handle the sparse data of the LDI well.

In this work, we investigate the compression of the sparse and nonrectangular supported data of the LDI. We first record the number of layers (NOL) at each pixel location. The LDI data is then reorganized into a more suitable layout by dividing the LDI into layers, each of which contains a mask indicating the existence of pixel in the layer. Each LDI layer is then separated into individual components, such as Y, Cr, Cb, alpha, and depth. The component images of each layer are compressed separately. We aggregate the data on the same layer so that the data is more compactly distributed. An arbitrary shape wavelet transform and coding algorithm is used to compress the aggregated data. Finally, the compressed bitstreams of the different layers and components are concatenated to form the compressed LDI bitstream. An empirical rate-distortion model is used to optimally allocate bits among all the components.

The paper is organized as follows. The data structure of the LDI is briefly reviewed in Section II. We explain the details of our compression method in Section III. Experimental results are shown in Section IV and a conclusion is given in Section V.

II. Layered Depth Image

The layered depth image (LDI) [1] consists of an array of pixels viewed from a single camera position, with possible multiple pixels along each line of sight. Shown in Fig. 1, an array of light rays is shot from the camera position P. The rays intersect the object at multiple points, which are ordered from front to back. The first intersection points of all light rays constitute the first layer, the second intersection points constitute the second layer, and so on. We denote the number of intersection points along each light ray as the number of layers (NOL). As an example, there are two layers for the light rays *PA* and *PC*, and four layers for the light ray *PB* in Fig. 1. At the original camera position P, only the pixels in the first layer are visible.

However, pixels in the back layers can be exposed as the view moves away from the original camera position. Unlike an ordinary image which consists just the luminance and chrominance components, each LDI pixel contains 63 bit information, which are distributed as follows: 8 bits each for the R, G and B components, 8 bits for the alpha channel, 20 bits for the depth of the object (the distance of the pixel to the camera), and 11 bits for the index into a splat table, which in turn is divided into 5 bits for the depth of the object, 3 bits for the x norm, and 3 bits for the y norm. The splat table is used to calculate the rendered pixel size in LDI rendering.

LDI rendering is rather straightforward. Shown in Fig. 2, the LDI is divided into four quadrants above, below, to the left and right of the epipolar point. Each quadrant is traversed separately in reverse scan line order. For each LDI pixel, its projection in the new view is incrementally calculated using the depth information, and the pixel with its luminance and chrominance component is rendered (splat) on to the output image with a size determined by the splat table. The operation can be performed very fast.

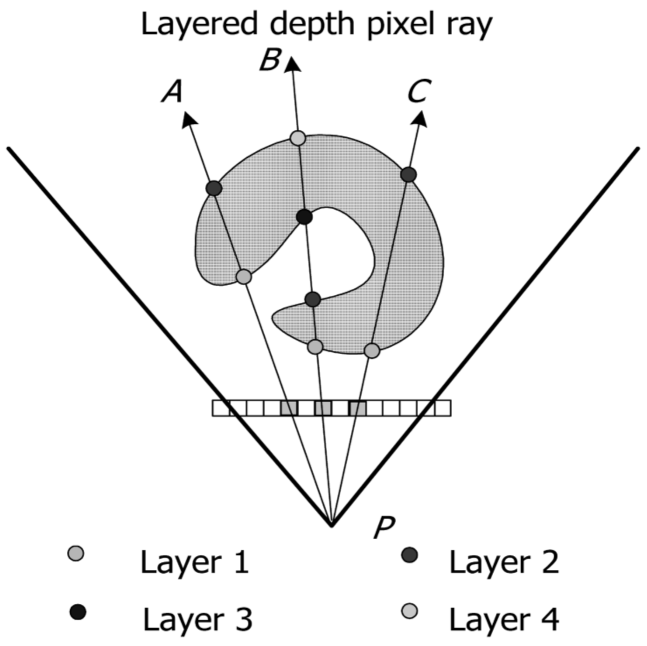


Fig. 1. Layered depth image.

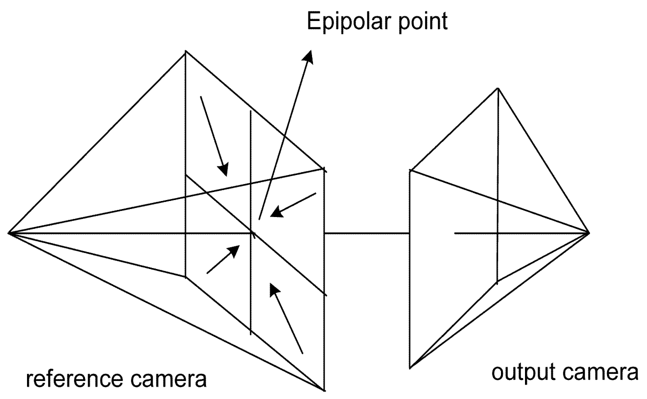


Fig. 2. McMillan's ordering algorithm.

III. Compression of the Layered Depth Image

Because of the special data structure of the LDI, existing still image compression methods, such as JPEG, cannot be applied directly or are not very efficient. There are three key characteristics of the LDI data. It consists of multiple layers; the content in the back layer is sparse; and each pixel consists of multiple property values, including the color, depth and splat table index. The key to develop an efficient LDI compression algorithm is thus to deal with these three issues.

A. Coding of the LDI Structure

We observe that the structure of the LDI is described by the number of layers (NOL) at each spatial location, which must be encoded before the component coding can be performed. Let $n(i,j)$ be the NOL at each location point (i,j). All $n(i,j)$ in the image space form a NOL image $\boldsymbol{N} = \{n(i,j)\}$, which must be encoded losslessly to preserve the LDI structure. We use the JPEG-LS [6], a lossless image coding standard, to encode the NOL image. Compared with a generic compression algorithm such as WinZip, JPEG-LS is faster and offers better compression performance.

With the NOL image, image mask for each layer can be easily recovered. The mask $m_l(i,j)$ for layer l LDI can be derived as

$$m_l(i,j) = \begin{cases} 1 & n(i,j) > l \\ 0 & n(i,j) \le l \end{cases}. \tag{1}$$

There is an LDI pixel at position (i,j) at layer l if and only if $m_l(i,j) = 1$. Through the mask, the location of pixel can be identified.

B. Separation of the Component Image

In the LDI, each existing pixel has multiple properties. An individual component image is formed for each property, and encoded separately. Experience in color image compression shows that multiple color components, such as the Y, Cr, Cb component can be separately compressed without loss of the compression efficiency. We expect that the same holds true for the LDI. Eight component images are formed: the R, G, B, alpha, depth, distance, x and y norm. Note that the distance, x and y norm components combined in the splat are separated into individual component images. The R, G and B components are further converted to the Y, Cr, Cb components. An example of eight separated component images (layer 1 of the scene Sunflowers) is shown in Fig. 3. The compressed LDI bitstream is formed by concatenating the NOL bitstream and the individual bitstream of each component image of each layer.

C. Preprocessing by Data Aggregation

The LDI component image is sparse and not of rectangular support. The more back the layer is, the fewer number of pixels there is. The component image in the LDI is much sparser than an image/video object in MPEG-4 [8], which is usually solid with a continuous boundary. We denote the percentage of the available pixels with regard to the rectangle image area for layer l as the covering ratio of layer l. Shown in Fig. 4, the covering

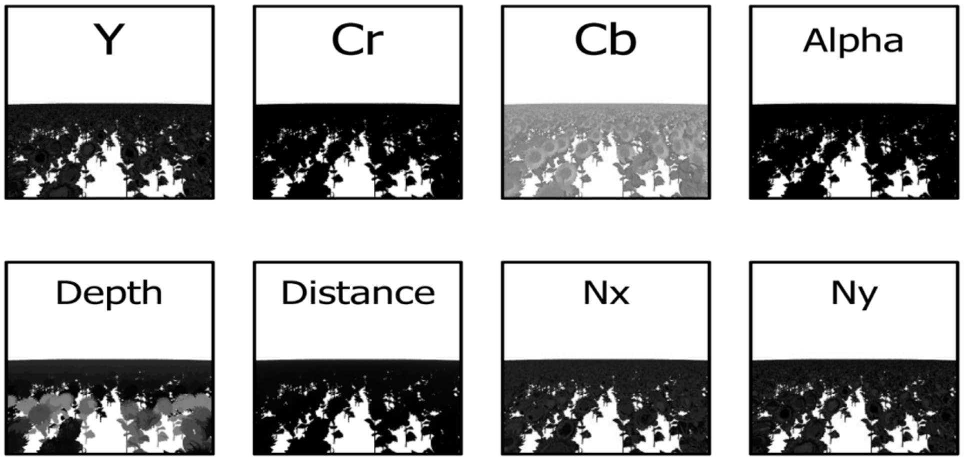


Fig. 3. Layered depth image: components separation.

ratios of the LDI scenes are compared with those of the MPEG-4 video object Akiyo and Bream. We show the covering ratios of different layers for the LDI, and show the covering ratios of different frames for the video object. It can be observed that the covering ratios of the video objects are nearly invariable through the frames, and are usually around 30–40%. On the contrary, the covering ratios decrease significantly as the layer increases in the LDI scenes, and can reach less than 3% for the LDI scene beyond layer five.

Another measure of the sparseness is the percentage of long pixel segments (POLS). In this work, the long segment is defined as a segment with more than four pixels. We compare POLS between the LDI scenes and the video objects, again for multiple layers and frames. The result is shown in Fig. 5. It is observed that for the video object, the POLS is close to 100%, which means the video object is pretty solid. The first layer of LDI is solid too, as close to 100% of object segments are longer than four pixels. However, starting from the second layer, the layer mask quickly disintegrates as the POLS decreases rapidly. Most LDI pixels in the back layer are usually isolated with no neighbors. Direct transform coding on such data set is not very efficient.

A preprocessing step is thus proposed to aggregate the data in each layer. The aggregation used here is simple horizontal aggregation: all the pixels in the same line are pushed to the left of the line. An example aggregation operation is shown in Fig. 6. We then transform and encode the aggregated data. Since the layer mask derived from the NOL is available at the decoder, the aggregation can be easily inversed. The aggregation algorithm lengthens the horizontal object segments, and improves the transform efficiency. Though it affects the alignment along vertical line, it is shown from the experiment that the advantage of the data aggregation outweighs its disadvantage. The aggregation operation is similar to the data push operation used in MPEG-4 Shape Adaptive DCT (SA-DCT) [8]. However, in MPEG-4, data is only aggregated within each block, while in the proposed algorithm, the data are aggregated over the entire image space. We have also tried to further aggregate the data along the vertical direction, i.e., to push the data again along the vertical axis. However, experiments show that such further aggregation does not provide additional performance improvement.

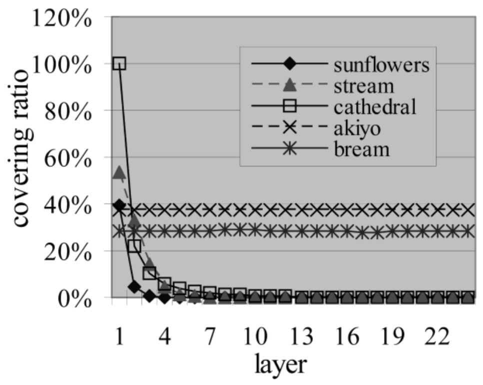


Fig. 4. Covering ratio of layers.

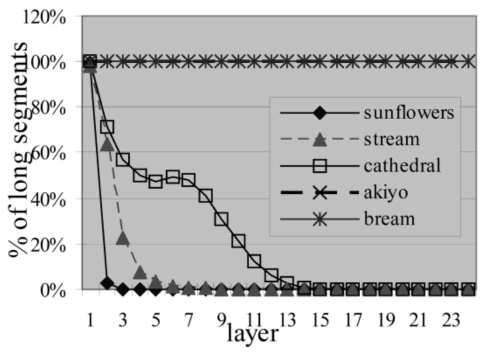


Fig. 5. Percentage of long segments (note: Akiyo and Bream curves happen to be the same).

D. Compression of the Component Image

We may use existing coding tools to compress the component images. One possible approach is the MPEG-4 SA-DCT mode [8], where all component images are treated as a video sequence and compressed by MPEG-4. An alternative approach is first to pad the component image to a rectangular image, and then to compress it with a rectangular still image coder such as the JPEG-2000 [12]. A third approach is to use the

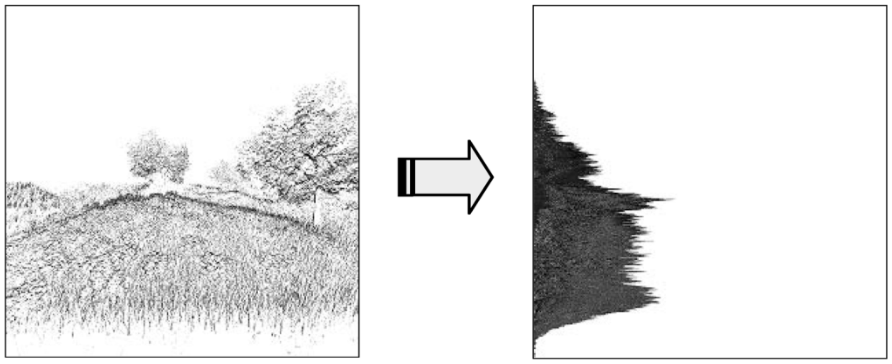


Fig. 6. Data aggregation.

video object wavelet (VOW) codec [9], which is an enhancement of the arbitrary shape wavelet codec in MPEG-4 [8]. In VOW, a shape adaptive wavelet transform [10], which is also called arbitrary shape wavelet transform with phase alignment (ASWP) [11], is used to decompose the aggregated component image into an exact number of wavelet coefficients as that of the original object. The wavelet coefficients are then quantized and entropy encoded with a partial bitplane coder. VOW outperforms the PEZW and ZTE used in MPEG-4 still wavelet object coding mode, because the partial bitplane coder does not use the wavelet domain zerotree assumption which is frequently broken, especially in object boundaries and for the prediction residue. VOW encoded bitstream has the embedding property, i.e., it may be truncated at any point and still decode a fair quality image. For details of the VOW coder, we refer to [9]. We have modified VOW so that it can handle a large range of bit depth (3 bits–20 bits).

E. Bit Allocation

For an LDI compressed bitstream, the available bit budget needs to be allocated among different components, i.e., among the luminance (Y), chrominance (C), depth, and splat table components. To calculate the actual amount of bits allocated to each component, an empirical rate-distortion optimization is performed. Let i index the component. Let n_l be the number of pixels at layer l, (R_i, D_i) be the average coding rate and distortion for component i. The bits required to encode the entire LDI can be formulated as

$$B = \sum_i \sum_l n_l \cdot R_i. \tag{2}$$

Note that for a certain component, the total number of bits $B_{l,i} = n_l \times R_i$ allocated to a certain layer is proportional to the number of pixels n_l in that layer. Thus, layers with more pixels are proportionally allocated with more bits. We measure the quality of the compressed LDI data set based on the rendered LDI images. A total of 16 LDI images are rendered, where one LDI image is rendered at the central location, and five LDI images per direction are rendered as we move away from the central location along the x, y and z axes. We use the average rendered PSNR of the 16 LDI images as a quality measure of the compressed LDI. Rather than deriving an analytical relationship between the LDI quality and the coding distortion of the individual component, which is very complicated and may not be useful to guide bit allocation, we model the overall quality of the compressed LDI as a linear weighted average of distortions of different component

$$D = \sum_i w_i \sum_l n_l D_i \tag{3}$$

where w_i is a weighting constant reflecting the importance of the component in the rendered LDI. Equation (3) may be considered as the first order approximation of the relationship between the distortion of the individual component and the overall LDI distortion. The optimal bit allocation is thus to minimize the overall LDI distortion D given a fixed bit budget B.

To solve the optimal bit allocation, we need the relationship between the coding distortion and rate for each component. Extensive empirical rate-distortion experiments conducted during the development of H.263 TMN8 rate-control [13] show that the distortion and rate have the following relationship with the quantization step size:

$$\begin{aligned} R &= \begin{cases} \frac{1}{2}\log_2\left(\frac{2e^2\sigma^2}{Q^2}\right) & \text{high bitrate} \\ \frac{e\cdot\sigma^2}{(\ln 2)Q^2} & \text{low/medium bitrate,} \end{cases} \quad \text{and} \\ D &= \frac{Q^2}{12} \end{aligned} \tag{4}$$

where σ is the source variance. The LDI component image compression usually operates at the low bitrate. We thus derive the empirical rate-distortion relationship for each component coding as

$$D_i = \frac{K_i}{R_i} \tag{5}$$

where $K_i = e \cdot \sigma_i^2/(12 ln2)$ is a constant determining the rate-distortion relationship for the component i. Using Lagrange multiplier, and minimizing the distortion (3) under the constraint of a fixed bit budget (2), we can easily derive the optimal bitrate R_i allocated to a specific component i as

$$R_i \propto A_i = \sqrt{K_i w_i} \tag{6}$$

where A_i is a component bit allocation factor determining how much bits should be assigned to each component coding. Note that it is the bit allocation factor A_i that should be estimated, there is no need to separately estimate parameter K_i and w_i. In

Section IV, we will show how A_i is determined by evaluating the relationship between the overall LDI quality and each component coding. The depth component is found to be the most important, because noise in the pixel depth shifts the position of the pixel, and leads to annoying holes and oddly appearing pixels. It receives the largest portion of the bit budget. The next largest portion is allocated to the luminance (Y) component. The chrominance components Cr and Cb receive the third largest portion of bits. The components distance, x norm and y norm receive very few bits, as they only affect the size of the rendered pixels, and the distortion of which is much smaller.

IV. Experimental Results

The effectiveness of the LDI compression algorithm is demonstrated with extensive experiments. The test scenes are the LDI scenes Sunflowers [Fig. 7(a)], Stream [Fig. 7(b)], and Cathedral [Fig. 7(c)]. We list the resolution, the number of layers, the number of pixels and the original size of the test data in column 2–5 of Table I, respectively.

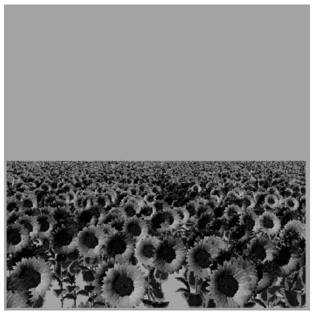
We show the peak signal-to-noise ratio (PSNR) of each component between the original LDI and the decompressed one. The PSNR of an individual component is calculated by

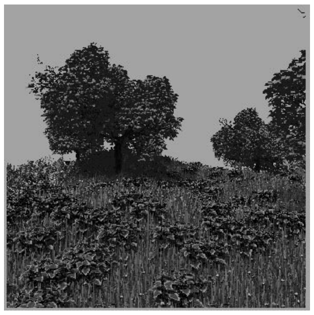
$$\mathrm{PSNR}_i = 10 \cdot \log_{10} \frac{(2^{bitdepth} - 1)^2}{\frac{1}{N} \sum_{l,x,y} [f(l,i,x,y) - f'(l,i,x,y)]^2} \quad (7)$$

where N is the number of pixels, $f(l,i,x,y)$ and $f'(l,i,x,y)$ are the original and decoded LDI pixel at layer l component i and position (x,y). The *bitdepth* is the number of bits in the original component data, which is 8 for component Y, Cr, Cb, alpha, 20 for the depth, 5 for the distance and 3 for the x and y norm. Though the individual component PSNR does not reflect the visual quality of the rendered LDI, it is a good tool to evaluate the effectiveness of component coding and data aggregation.

We first investigate the compression efficiency of the number of layer (NOL) image. We benchmark JPEG-LS against the Winzip version 7.0. Both algorithms encode the NOL image losslessly. Table II shows the compression ratios of the layer number image by Winzip and JPEG-LS, respectively. We observe that the JPEG-LS improves the compression ratio by a factor of 1.3 times over the more generic WinZip algorithm.

In the second experiment, we demonstrate the effectiveness of data aggregation presented in Section III-C. We encode the LDI component with the same coding tools, at the same bitrate. The component PSNRs for the three LDI scenes, with and without data aggregation, are shown in Table III. We show the coding results of video object wavelet coding (VOW) for components Y, C (combination of Cr and Cb components), depth, distance, and norm (combination of x and y norms). We also list the coding results of MPEG-4 for component Y, with and without data aggregation. It is observed that the data aggregation process significantly improves the coding performance. For the VOW coding algorithm, the performance gain obtained by aggregating the Y data component ranges from 0.58–0.86 dB, with an average of 0.71 dB. Data aggregation yields an especially large gain for the depth and distance components, which average to 14.18 dB and 8.18 dB, respectively. The data aggregation is effective for MPEG-4 SA-DCT coding as well, with an average performance boost of 1.56 dB for the Y component. Among the 18 comparison experiments, there is only one experiment (Y component MPEG-4 coding of Cathedral) in which data aggregation causes


(a)


(b)

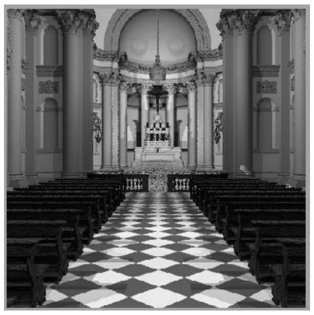

(c)

Fig. 7. Original LDI dataset: (a) Sunflowers, (b) Stream, and (c) Cathedral.

TABLE I
ORIGINAL LDI SCENE

LDI	Resolution	Layers	Pixels	Size
Sunflowers	1024x1024	6	463,821	5.55MB
Stream	1024x1024	12	1,125,690	10.86MB
Cathedral	1024x1024	24	1,588,812	14.1MB

TABLE II
COMPRESSION RATIOS OF THE NUMBER OF LAYER (NOL) IMAGE

	WinZip	JPEG-LS	Compression ratio gain
Sunflowers	18.3:1	25.7:1	28.8%
Stream	5.5:1	7.1:1	22.5%
Cathedral	7.7:1	10.0:1	23.0%

TABLE III
LDI CODING WITH AND WITHOUT DATA AGGREGATION (dB)

Algorithm	Component	Sunflowers		Stream		Cathedral		Average gain
		w/o Agg	w/ Agg	w/o Agg	w/ Agg	w/o Agg	w/ Agg	
VOW	Y	30.17	30.75	27.21	28.07	25.44	26.12	0.71
	C	34.92	35.52	26.87	26.87	23.52	24.58	0.55
	Depth	47.96	57.84	38.70	51.53	34.25	54.07	14.18
	Distance	16.42	20.33	20.27	27.34	16.72	30.27	8.18
	Norm	14.42	14.93	9.81	9.88	9.84	10.23	0.32
MPEG	Y	26.91	29.56	26.26	29.31	31.27	30.24	1.56

TABLE IV
COMPARISON OF MPEG-4, JPEG 2000 AND VOW CODING (dB)

	Sunflowers			Stream			Cathedral		
	MPEG	J2K	VOW	MPEG	J2K	VOW	MPEG	J2K	VOW
Y	30.25	28.74	**30.95**	29.52	27.73	**30.24**	31.81	30.24	**32.67**
C	28.06	28.00	**28.36**	28.22	27.97	**28.29**	37.14	36.23	**37.85**
Depth	N/A	52.68	**62.71**	N/A	47.46	**53.75**	N/A	50.34	**59.83**
Distance	N/A	17.76	**34.61**	N/A	26.29	**28.41**	N/A	18.68	**26.43**
Norm	N/A	11.05	**11.30**	N/A	10.35	**10.52**	N/A	15.46	**16.19**

a negative effect.[1] In general, the data aggregation proves itself as an effective tool to significantly improve the transform and coding performance in the LDI compression.

In the third experiment, we evaluate the three different component coding tools—the MPEG-4, JPEG 2000 and VOW coder. We again encode the three LDI scenes at the same bitrate. The PSNR results for the components Y, C, depth, distance, and norm are shown in Table IV, for different schemes and different LDI scenes. With the existing MPEG-4 software, we have difficulties to encode the component depth due to its large bit-depth, and to encode the distance and norm components as the allocated bit rate is less than 0.05 bpp. Therefore, no test results are reported for the aforementioned components in MPEG-4. The best coding results among the three algorithms are highlighted with boldface. It is observed that the video object wavelet coder (VOW) achieves the best result among the three schemes, for all component coding and for all LDI scenes. It outperforms the MPEG-4 for an average of 0.76 and 0.36 dB for the Y and C components, respectively. Moreover, it outperforms the JPEG 2000 for an average of 2.38, 0.77, 8.60, 8.90, and 0.38 dB for the Y, C, depth, distance and norm components. VOW demonstrates itself as a superior coding tool for the LDI component coding.

[1]As mentioned in Section III-C, we note that MPEG-4 SA-DCT coding mode aggregates the data within each block before the following arbitrary shape DCT transform and coding operation. Therefore, with an additional frame data aggregation operation, the data are aggregated twice in MPEG-4 SA-DCT, which may be the reason of PSNR degradation in scene Cathedral. Nevertheless, even for MPEG-4 SA-DCT, data aggregation improves the compression performance in two of the three scenes.

In the fourth experiment, we investigate bit allocation among different components. We vary the bit rate of one component while fixing the rate of all others. The LDI quality is then measured as the average MSE of the 16 rendered LDI images at each bit rate point. We show the curve of the component bit rate versus the LDI quality for the Y and depth component in Figs. 8 and 9, respectively, where the horizontal axis is the bit rate of the investigating component, and the vertical axis is the MSE of the LDI images. The solid curve is the actual coding result and the dashed curve is the theoretical value predicted by (3) and (5). It is observed that the two curves match nicely. Our two key assumptions in the bit allocation thus hold pretty well: the average rendered distortion can be approximated by the linear combination of the component distortion (3); and the distortion is inversely proportional to the coding rate (5). We may calculate the component bit allocation factors of the Y and depth component from Figs. 8 and 9 as: $A_y^2 = 98.43$ and $A_d^2 = 343.96$. The optimal bitrate allocation at a certain compression ratio can thus be calculated. For example, at 10:1 compression of the Sunflowers scene, we roughly have $R_y = 2.86$ bpp and $R_d = 5.34$ bpp. The bit allocation factors of other components can be calculated in a similar fashion.

The average rendered PSNRs for the compressed LDI scenes at ratio 10:1 and 17:1 are shown in Table V. Since the three LDI scenes are all synthetic sceneries with high details, it is difficult to raise the compression ratio further. Nevertheless, the subjective quality of the rendered LDI view is good even though the PSNR number is not high. The original and the decompressed LDI scene Sunflowers under compression ratio

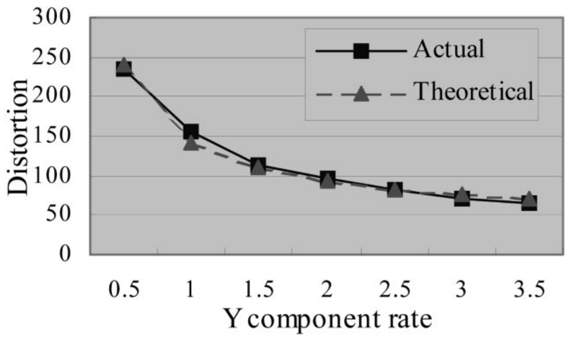


Fig. 8. R-D for Y Component.

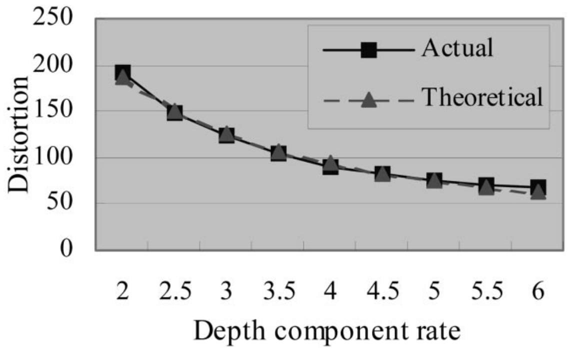
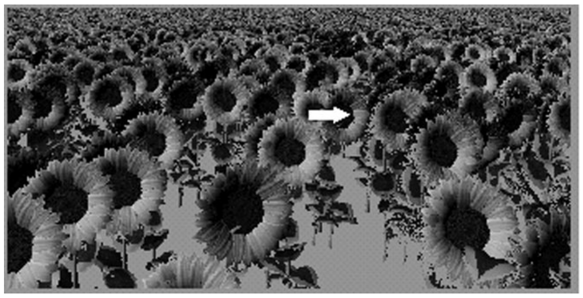


Fig. 9. R-D for Depth Component.

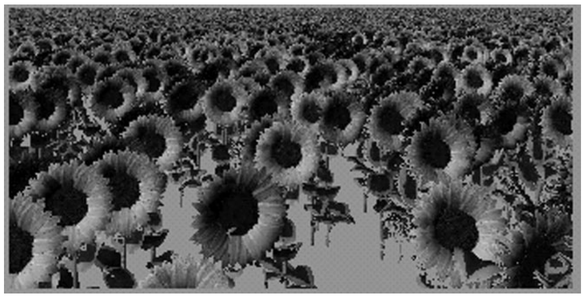
10:1 and 17:1 rendered at the same virtual position are shown in Fig. 10(a)–(c), respectively. Little distortion is observable from the compressed LDI at ratio 10:1. The quality of the LDI view compressed at 17:1 is still good, though distortion starts to be observable. Apart from the normal image coding distortions such as the blur and ringing artifacts, the object in a highly compressed LDI starts to disintegrate, especially around the boundaries (pointed by the arrows). This is a unique distortion in LDI coding caused by the coding distortion of the depth component, which makes the rendered pixel to shift to a wrong location. It explains the reason that we assign the largest portion of bit budget to the depth component.

V. Conclusions

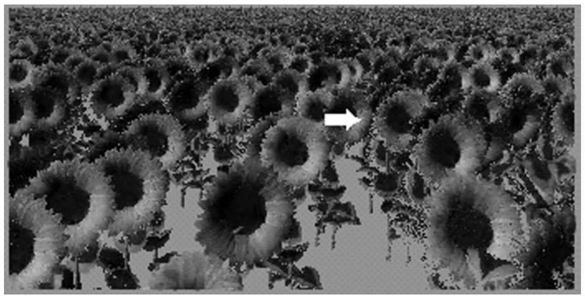
In this paper, an efficient algorithm is developed to compress the layered depth image (LDI). We first record the number of layers (NOL) at each pixel location. The LDI is then divided into layers, each of which contains a mask indicating the existence of pixel in this layer. After that, the LDI data in each layer is divided into individual components, which are compressed separately. We aggregate the data in the same layer so that the data is more compactly distributed. An arbitrary shape wavelet transform and coding algorithm is then used to compress the aggregated data. The compressed bitstream of different layers and components are then concatenated to form the compressed bitstream of LDI. An empirical rate-distortion model is developed to optimally allocate bits among the components. The proposal scheme can compress an LDI up to 17:1 with minimal visual distortion.



(a)



(b)



(c)

Fig. 10. Rendered Sunflowers scene (bottom portion) for the (a) original LDI, (b) decompressed LDI at 10:1, and (c) decompressed LDI at 17:1.

TABLE V
LDI Quality: In Term of the Average PSNR of Rendered LDI Images (dB)

	Compression ratio	Y	Cr	Cb
Sunflowers	10:1	31.12	29.10	35.11
	17:1	26.95	25.97	34.84
Stream	10:1	24.86	32.02	26.61
	17:1	21.95	30.30	26.19
Cathedral	10:1	28.94	34.78	38.97
	17:1	23.98	32.61	37.96

Acknowledgment

The authors would like to acknowledge J. Shade and M. Cohen for providing the LDI render source code and unpack the LDI data set; O. Deussen for the original LDI scene data Sunflowers, Stream, Cathedral; and G. Xing for his contribution in the development of the video object wavelet coder.

REFERENCES

[1] J. Shade, S. Gotler, L. He, and R. Szeliski, "Layered depth image," in *Proc. ACM SIGGRAPH'98*, Orlando, FL, July 1998, pp. 231–242.

[2] C. Chang, G. Bishop, and A. Lastra, "LDI tree: A hierarchical representation for image-based rendering," in *Proc. ACM SIGGRAPH'99*, Los Angeles, CA, July 1999, pp. 291–298.

[3] L. McMillan, "Computing Visibility Without Depth," University of North Carolina, 95-047, 1995.

[4] ——, "A List-Priority Rendering Algorithm for Redisplaying Projected Surfaces," University of North Carolina, 95-005, 1995.

[5] L. McMillan and G. Bishop, "Plenoptic modeling: An image-based rendering system," in *ACM SIGGRAPH'95*, Aug. 1995, pp. 39–46.

[6] M. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," *IEEE Trans. Image Processing*, vol. 9, pp. 1309–1324, Aug. 2000.

[7] R. Krishnamurthy, B. Chai, H. Tao, and S. Sethuraman, "Compression and transmission of depth maps for image-based rendering," in *2001 Int. Conf. Image Processing*, Vancouver, BC, Canada, pp. 828–831.

[8] "MPEG-4 Video, MPEG-4 Video Verification Model Version 13.0,", ISO/IEC JTC1/SC29/WG11 N2687, 1999.

[9] G. Xing, J. Li, S. Li, and Y. Zhang, "Arbitrarily shaped video object coding by wavelet," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 1135–1139, Oct. 2001.

[10] S. Li and W. Li, "Shape adaptive discrete wavelet transform for arbitrarily shaped visual object coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 725–743, Aug. 2000.

[11] J. Li and S. Lei, "Arbitrary shape wavelet transform with phase alignment," in *Proc. 1998 IEEE International Conf. Image Processing*, vol. 3, Chicago, IL, Oct. 1998, pp. 683–687.

[12] "JPEG VM ad-hoc, JPEG 2000 Verification Model 5.2 (Technical Description),", ISO/IEC JTC1/SC29/WG1N1422, 1999.

[13] J. Ribas and S. Lei, "Rate control in DCT video coding for low-delay communications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 172–185, Feb. 1999.

Jiangang Duan received the B.S. and M.S. degrees from the Department of Electrical Engineering, Tsinghua University, Bejing, China, in 1999 and 2001, respectively.

He is currently with Intel China Software Lab, Shanghai, China. From November 1999 to October 2000, he was an Intern and held part-time position at Microsoft Research, Beijing, China. His research interests includes image-based rendering, multimedia processing and transmission, high-availability operating systems, etc.

Jin Li (S'94–A'95–M'96–SM'99) received the B.S., M.S., and Ph.D. degrees in electrical engineering, all from Tsinghua University, Beijing, China, in 1990, 1991, and 1994, respectively.

From 1994 to 1996, he served as a Research Associate at the University of Southern California (USC). From 1996 to 1999, he was a Member of Technical Staff at the Sharp Laboratories of America (SLA), and represented the interests of SLA in the JPEG2000 and MPEG4 standardization efforts. He was a Researcher/Project Leader at Microsoft Research China from 1999 to 2000. He is currently a Researcher at Microsoft Research Redmond. Since 2000, he has also served as an Adjunct Professor in the Electrical Engineering Department, Tsinghua University. is an active contributor of the ISO JPEG 2000/MPEG4 project. He has published 15 journal papers and 40+ conference papers in a diversified research field, with interests cover image/video compression and communication, audio compression, virtual environments, and graphic compression. He holds five U.S. patents, with many more pending. He is an Area Editor for the *Journal of Visual Communication and Image Representation*.

Dr. Li is the recipient of the 1994 Ph.D. thesis award from Tsinghua University and the 1998 Young Investigator Award from SPIE Visual Communication and Image Processing.