ON THE COMPRESSION OF IMAGE BASED RENDERING SCENE

Jin Li, Harry Shum and Ya-Qin Zhang
Microsoft Research China
5F Research Beijing Sigma Center, 49 Zhichun Road, Haidian, Beijing 100080, P. R. China
Tel. + 86 10 6261 7711 – 5793, Email: jinl@microsoft.com

ABSTRACT

In image based rendering (IBR), a 3D scene is recorded through a set of photos, and a novel view is rendered by assembling data from the photo set. Compression is essential to reduce the huge data amount of IBR. In this paper, we examine three categories of IBR compression algorithms: the block coder, the reference coder and the high dimensional transform (wavelet) coder. It is observed that the block coder consumes the least computation resource, however, its compression ratio is low. The reference coder achieves good compression ratio with reasonable computation complexity. The high dimensional wavelet coder achieves the best compression ratio, however, it is also the most complex.

Keywords: image based rendering (IBR), compression, block coding, vector quantization, reference coding, high dimensional transform, wavelet.

1. INTRODUCTION

Image-based rendering (IBR) has attracted much attention recently in realistic scene/object representation. Proposed by Adelson and Bergen [1] as a 7D plenoptic function, IBR modeled a 3D dynamic scene by recording the light rays at every space location, towards every possible direction, over any range of wavelengths and at any time. By ignoring time and wavelength and discretizing the data set, McMillan and Bishop [2] defined plenoptic modeling as generating a continuous 5D plenoptic function from a set of discrete samples. The proposals of Lumigraph [3] and Lightfield [4] made IBR more popular, as they provided a clever 4D parameterization of the plenoptic scene under the constraint that the object or the viewer is within some 3D bounding boxes. Even though most IBR scenes are synthetic, it is possible to capture Lumigraph/Lightfield of a realistic scene/object. There are technical challenges, e.g., careful motion control of the camera array so that the pictures can be taken from regular grids parallel to the image plane. Shum and He [5] proposed concentric mosaics, a 3D plenoptic function restricting viewer movement inside a planar circle and looking outside. A concentric mosaic scene can be captured very easily by rotating a single camera at the end of a round-swinging beam, with the camera pointing outward and shooting images as the beam rotates. In either the Lumigraph/Lightfield/concentric mosaics, the 3D object/scene is recorded by a set of photos. New view is rendered by splitting the rendered view into a set of rays, where each ray is reconstructed through data in the photo set.

The IBR is an attractive tool for quick modeling and rendering of a complex 3D object/environment without recovering the object geometry. However, huge data amount is involved in IBR. As an example, the Lumigraph scene *Fruit* (shown in Figure 3) consists of a 32x32 array of images with resolution 256x256. The total raw data amount is *196MB.* The concentric mosaic scene *Lobby* (shown in Figure 4) consists of 1350 images with resolution 320x240. The total raw data amount is *297MB*. The data amount can be even larger if the IBR scene is of higher resolution.

The importance of compression has been realized from the birth of IBR. Since IBR consists of a set of photos, it is natural to apply existing still image/video compression technologies to IBR. However, the IBR scene bears unique characteristics, which have lead to new challenges in compression. On the one hand, IBR is a 1D (for concentric mosaics) or 2D (for Lumigraph/Lightfield) image array, with regular camera motion between images. There is better cross-frame correlation in the IBR than in the video sequences. On the other hand, the distortion tolerance of IBR is small, because each view of the IBR is static, and the human visual system (HVS) is much more sensitive to static distortions than time-variant distortions. Since a rendered view of IBR is formed by a combination of the image rays, certain HVS properties such as spatial and temporal masking may not be used in IBR compression. Most importantly, a compressed image bitstream is usually decompressed to get back the original image, a compressed video bitstream is played frame by frame, however, a compressed IBR bitstream should not be fully decompressed and then rendered. In fact, the decompressed IBR data is so large that most hardware today has difficulties to handle it. It is therefore essential to maintain the IBR data in the compressed form, and decode only the contents needed to render the current view. We call such concept the just-in-time (JIT) rendering. JIT rendering is a key to design IBR compression and rendering algorithm.

In this paper, three categories of IBR compression systems, the block coder, the reference coder and the high dimensional transform (wavelet) coder, are surveyed. The JIT rendering for each compression systems is discussed. The paper is organized as follows. We briefly review the data structure of Lumigraph/Lightfield and concentric mosaics in Section 2. The compression systems are then examined in Section 3. Performance comparison of representative systems is shown in Section 4. A conclusion is given in Section 5.

2. THE IMAGE BASED RENDERING

2.1 The Lumigraph/Lightfield

The Lumigraph [3] and Lightfield [4] can represent a complete 3D view of the object, as long as the object can be constrained in a bounding box. It can also represent environmental views if the user can be constrained in a bounding box. By placing the object in its bounding box which is surrounded by another larger box, the Lumigraph/Lightfield indexes all possible light rays with coordinate of the ray entering and exiting one of the six parallel planes of the double bounding boxes. The data is thus composed of six 4D functions. Let the plane of the inner box be indexed with coordinate *(u,v)* and that of the outer box with coordinate *(s,t)*. Usually, the discretization is denser for the inner bounding box closer to the object, and sparser for the outer bounding box. We can consider the Lumigraph/Lightfield as six two-dimensional image arrays, with all the light rays coming from a fixed *(s,t)* coordinate forming one image. This is equivalent to set a camera at each coordinate *(s,t)* and taking a picture of the object with the imaging plane be the *(u,v)* plane. An example of the Lumigraph/Lightfield image array is shown in Figure 2. Note that the neighboring Lumigraph/Lightfield images are very similar to one another. To create a new view of the object, we just split the view into its light rays, which are then calculated by interpolating existing nearby light rays in the image arrays. The novel view is generated by reassembling the split rays together.

2.2 The concentric mosaics

A concentric mosaic scene is captured by mounting a camera at the end of a round-swinging beam, and shooting images at regular intervals as the beam rotates [5]. The resultant concentric mosaics are a sequence of shots. For a typical realistic environment with large depth variation, 900 to 1500 shots have to be captured in a circle to render the scene properly without alias.

The concentric mosaics can render novel views of the environment without 3D or depth information. Shown in Figure 1, let R be the length of the round-swinging beam, θ_{FOV} be half of the horizontal field of view (FOV) of the camera, a concentric mosaic scene can render an arbitrary view with the same FOV looking at any directions within an inner circle of radius $r = R\sin\theta_{FOV}$. Let P be a novel viewpoint and AB be the field of view to be rendered. We split the view into multiple vertical slits, and render each slit independently. Let the slit PV be a rendered slit. We simply search for the slit $P'V$ in the captured dataset, where P' is the intersection between ray PV and the camera path. It is apparent that the rays $P'V$ and PV look at the same direction. Therefore, what is rendered at PV can be recovered from that is observed in $P'V$[1]. Since there may not be an exact slit $P'V$ in the dataset, we may bilinear interpolate the four slits closest to $P'V$, i.e., P_1V_{11}, P_1V_{12}, P_2V_{21} and P_2V_{22}, to reconstruct $P'V$. We call such rendering mode the bilinear interpolation mode. Alternative, a point sampling mode may be used, where the closest slit in the photo set is used to reconstruct $P'V$.

3. IBR COMPRESSION

3.1 Block coder - spatial vector quantization (SVQ)

We first examine the block coder. A spatial domain vector quantization (SVQ) was used in [4] to compress blocks of the Lightfield, and in [5] to compress blocks of the concentric mosaics. The basic idea is to chop the IBR photo set into small blocks, either 2D blocks within frames [4], or 3D blocks across frames [5], and then to encode each block with a spatial domain vector quantizer. SVQ is simple to decode, and the just-in-time (JIT) rendering is easy, as SVQ index is of equal length. The main weakness of SVQ is that the compression ratio is limited. Compression ratio of 6:1 to 23:1 is achieved in [4] and 12:1 is achieved in [5] for reasonable quality IBR scene. In [4], gzip is used to further compress the index array by a factor of 5:1 to 8:1. Though achievable for the synthetic object and through low pass filtering of the photo set, such high compression of SVQ index may not be achievable for a realistic IBR object/scene. Moreover, the gzipped SVQ index cannot be randomly accessed. SVQ is also time-consuming at the encoding stage.

Alternative technologies can be used to encode the IBR blocks. With a scheme similar to JPEG, Miller *et al.* [8] encoded blocks of light field with DCT and Huffman coding. Though achieving better compression than SVQ with simpler encoding, the DCT and Huffman encoded block is of variable length and cannot be easily accessed. An index table can be built to randomly index the compressed block bitstream. However, this adds overhead. An alternative solution is to encode each block to equal length, just as SVQ. We may, for example, transform the block with DCT and then encode the block coefficients with an embedded bitplane coder, in which the bitstream can be truncated to meet certain length constraint exactly [12]. We may also compress the block with block truncation coding (BTC), or the DXTn technology used in DirectX [11]. The embedded DCT coding, DXTn and SVQ are all capable to compress a block to a fixed length bitstream. Among the algorithms, the SVQ is the most complex in encoding, followed by embedded DCT coding and then DXTn. However, in term of decoding complexity, the SVQ is the simplest, and then the DXTn and embedded DCT. In term of compression performance, the embedded DCT coding is a little better than SVQ and DXTn. Neither algorithm achieves high compression ratio, because the correlation between the blocks is not used in the block coder.

3.2 Reference frame coder

Since the IBR scene consists of photo sets, video coders such as the MPEG-x or H.26x can be applied. However, direct MPEG-x or H.26x compressed IBR bitstream is coupled tightly and has to be entirely decompressed for rendering. Besides, the specific characteristics of IBR, such as the regular camera motion between shots can be used to further improve the compression. Kiu et al.[6], Magnor and Girod [7][9][10], Zhang and Li [14] have adopted an MPEG like algorithm to compress the Lightfield/Lumigraph. Similar work called the reference block coder (RBC) is proposed in [14] to compress the concentric mosaics. Using RBC as an example, it classified the photo shots in the concentric mosaics into two categories, the anchor (A) frames and the predicted (P) frames. The A frames are independently encoded to provide anchor of access, and the P frames are predictively encoded with reference to the surrounding A frames. Both frames are segmented into macroblocks of size 16x16, which are encoded by DCT and a run-level Huffman coding exactly the same as MPEG-2. RBC differs from MPEG-2 in three aspects. First, a two-level hierarchical table is embedded in RBC for indexed bitstream access. Second, the P frame in RBC refers only to the A frame, so that random access of the data set is more readily available. Third, regular camera panning motion between neighboring frames of the concentric mosaics has been taken into consideration. By modifying the frame structure and motion compensation scheme, RBC slightly beats MPEG-2 in term of

[1] In fact, ray slit PV is a vertical scaled version of $P'V$, where the vertical scaling is necessary for depth correction [5].

compression performance. More significantly, the RBC compressed bitstream can be JIT rendered in real time. In a RBC render, only the data used in the current frame is decoded. Caches have been used extensively in RBC to avoid macroblocks to be decoded repeatedly and to speed up the rendering. A running scene of RBC rendered concentric mosaics is shown in Figure 4. The render can run smoothly on a desktop PC with Pentium II 300 CPU and 64MB RAM.

3.3 High dimensional transform (wavelet) coder

Since IBR is a static photo set, high dimensional transform, especially the high dimensional wavelet transform can be used to compress the data set. 3D wavelet compression algorithms have already been developed to compress the concentric mosaics [13][15][16]. 3D wavelet is computational intensive in both encoding and decoding, however, it achieves the best possible performance in compression. Moreover, the wavelet compressed concentric mosaics can be accessed with resolution, spatial and quality scalability, and are thus very suitable for the Internet application. Direct 3D wavelet transform and coding[13] offered only a comparable performance to that of MPEG-2. This was due to the fact that the concentric mosaics did not align well in the cross mosaics direction. Wu *et al*. [16] proposed to rebin and align the concentric mosaics across frames into a 3D region of arbitrary shape support, which is then 3D wavelet transformed and compressed. We call the algorithm smart rebinning wavelet coder. The algorithm more than doubles the compression ratio of MPEG-2, or outperforms it by an average of 3.7dB at the same bit rate.

Rendering of wavelet compressed IBR scene is tricky. Straightforward decompression of the entire compressed bitstream is obviously not a good choice, as it is too memory intensive and requires a long delay at the start. An alternative approach is to segment the IBR into spatially tiled high dimensional blocks, and compress each tile separately. The approach has visible blocking artifacts and does not achieve high compression efficiency since correlation across tiles cannot be easily exploited. A better approach, termed progressive inverse wavelet synthesis (PIWS) was proposed in [15]. In PIWS, wavelet transformed coefficients were segmented into blocks and entropy encoded. PIWS performed only the necessary inverse calculations to reconstruct the coefficients used in the current view, and decoded only the coefficient blocks used by the inverse wavelet operation. With extensive cache usage, PIWS was also able to render the wavelet compressed concentric mosaics in real time.

4. EXPERIMENTAL RESULTS

Representative block coder, reference coder and high dimensional transform coder are investigated with experimental results in this section. The IBR scene used in the experiment is the concentric mosaic scene *Lobby* and *Kids*. The scene *Lobby* has 1350 frames at resolution 320x240, with total 297MB data. The scene *Kids* has 1462 frames at resolution 352x288, with total 424MB data. We first compare the compression performance. The spatial vector quantizer (SVQ) in [5] is used to represent the block coder; the reference block coder (RBC) in [17] is used to represent the reference coder; and the smart rebinning wavelet coder (wavelet) in [16] is used to represent the high dimensional wavelet coder. The SVQ coder can only achieve a compression ratio of 12:1. In term of compression performance, it is no match with the RBC or the wavelet coder. The compression performance of the RBC and wavelet coder is further compared with a benchmark MPEG-2 video coder. We compress the *Lobby* scene at ratio 200:1 (0.12bpp, 1.48MB) and 120:1 (0.2bpp, 2.47MB), and the *Kids* scene at 100:1 (0.24bpp, 4.24MB) and 60:1 (0.4bpp, 7.07MB). The peak signal-to-noise-ratio (PSNR) between the original and decompressed scene of the MPEG-2, RBC and the wavelet coder is shown in Table 1. It is observed that RBC outperforms MPEG-2 for an average gain of 0.4dB. The smart rebinning wavelet coder outperforms MPEG-2 for 3.0 to 4.1dB, with an average of 3.7dB, and outperforms RBC for 2.3 to 4.5dB, with an average of 3.2dB. It shows that in term of compression performance, the high dimensional transform (wavelet) coder is the best.

Table 1 PSNR (dB) for the compressed concentric mosaics.

Dataset / Algorithm	Lobby 0.2bpp	Lobby 0.12bpp	Kids 0.4bpp	Kids 0.24bpp
MPEG-2	32.2	30.4	30.1	28.3
RBC	32.8	29.8	31.5	28.7
Wavelet	36.3	34.3	33.8	31.3

Next, we investigate the rendering speed of SVQ, RBC and the wavelet coder. The average rendered frames per second for the three coders are listed in Table 2. Both the point sampling and the bilinear interpolation rendering modes are investigated. SVQ achieves the fastest rendering speed. The speed of RBC is satisfactory, and only about 27-29% slower than that of SVQ. The wavelet coder with the trick of progressive inverse wavelet synthesis (PIWS) [15] is the slowest, and is 51-55% slower than SVQ, and 33-37% slower than RBC. Nevertheless, real time rendering of wavelet compressed concentric mosaic is still achievable.

Table 2 Rendering speed (frames per second) for the compressed concentric mosaics (on a 500 Mhz Pentium III PC).

Rendering setting / Algorithm	Point sampling mode	Bilinear interpolation
SVQ	17.7	14.9
RBC	12.5	10.9
Wavelet	7.9	7.3

5. CONCLUSIONS

In this paper, three categories of image based rendering (IBR) compression algorithm is investigated. The block coders, such as SVQ, are the least complex in decoding and can easily achieve just-in-time (JIT) rendering. However, their compression ratio is poor. The reference block coder (RBC) achieves good compression ratio and real-time JIT rendering capability. The high dimensional wavelet coder achieves the best compression performance, and with effort, can also achieve JIT rendering. However, it is also the most complex.

6. BIBLIOGRAPHY

[1] E. H. Adelson, and J. R. Bergen, "The plenoptic function and the elements of early vision", *Computational Models of Visual Processing, Chapter 1*, Edited by Michael Landy and J. Anthony

Movshon. The MIT Press, Cambridge, Mass. 1991.
[2] L. McMillan and G. Bishop, “Plenoptic modeling: an image-based rendering system”, Computer Graphics (SIGGRAPH’95), pp. 39-46, Aug. 1995.
[3] S. J. Gortler, R. Grzeszczuk, R. Szeliski and M. F. Cohen, “The Lumigraph”, Computer Graphics (SIGGRAPH’96), pp. 43-54, Aug.1996.
[4] M. Levoy and P. Hanrahan, “Light field rendering”, Computer Graphics (*SIGGRAPH’96)*, pp. 31, Aug. 1996.
[5] H. Shum and L. He. “Rendering with concentric mosaics”, *Computer Graphics (SIGGRAPH'99)*, pp. 299-306, Aug. 1999.
[6] M. Kiu, X. Du, R. Moorhead, D. Banks and R. Machiraju, “Two-dimentional sequence compression using MPEG”, *in SPIE: Visual Communication and Image Processing (VCIP’98)*, pp. 914-921, Jan. 1998.
[7] M. Magnor and B. Girod, "Adaptive block-based light field coding," Proc. 3rd International Workshop on Synthetic and Natural Hybrid Coding and Three-Dimensional Imaging IWSNHC3DI'99, Santorini, Greece, pp. 140-143, Sep. 1999.
[8] G. Miller, S. Rubin and D. Ponceleon, “Lazy decompression of surface light fields for precomputed global illumination”, *Eurographics Rendering Workshop 1998*, Vienna, Austria, Jun. 1998, pp. 281-292.
[9] M. Magnor and B. Girod, “Model-aided light field coding”, *in SPIE: Visual Communication and Image Processing (VCIP’2000)*, Vol. 4067, No. 2, Perth, Australia, Jun. 2000.
[10] M. Magnor and B. Girod, “Hierarchical coding of light fields with displarity maps”, *Proc. International Conference on Image Processing (ICIP-99)*, Kobe, Japan, pp. 334-338, Oct. 1999, pp.334-338.
[11] Microsoft Document, “Creating compressed textures”, http://msdn.microsoft.com/library/psdk/directx/ddover_2jef.htm
[12] J. Li, J. Li, and J. Kuo, “Layered DCT still image compression,” *IEEE Trans. On Circuit and System for Video Technology*, Vol. 7, No.2, pp.440-443, Apr. 1997.
[13] L. Luo, Y. Wu, J. Li and Y. Zhang, “Compression of concentric mosaic scenery with alignment and 3D wavelet transform”, *in SPIE: Image and Video Communication and Processing,* Vol. 3974, No. 10, San Jose CA, Jan. 2000, pp. 89-100.
[14] C. Zhang and J. Li, “Compression of Lumigraph with multiple reference frame (MRF) prediction and just-in-time rendering”, *in Proc. Of IEEE Data Compression Conference (DCC’2000),* Snowbird, Utah, Mar. 2000, pp. 254-263.
[15] Y. Wu, L. Luo, J. Li and Y. Zhang, “Rendering of 3D wavelet compressed concentric mosaic scenery with progressive inverse wavelet synthesis (PIWS)”, *in SPIE: Visual Communication and Image Processing (VCIP’2000),* Vol. 4067, No. 4, Perth, Australia, Jun. 2000.
[16] Y. Wu, C. Zhang, J. Li and J. Xu, “Smart-rebinning for compression of the concentric mosaics”, *submitted to ACM Multimedia,* Los Angeles, CA, Oct. 2000.
[17] C. Zhang and J. Li, “Compression and rendering of concentric mosaics with reference block codec (RBC)”, *in SPIE: Visual Communication and Image Processing (VCIP’2000),* Vol. 4067, No. 5, Perth, Australia, Jun. 2000.

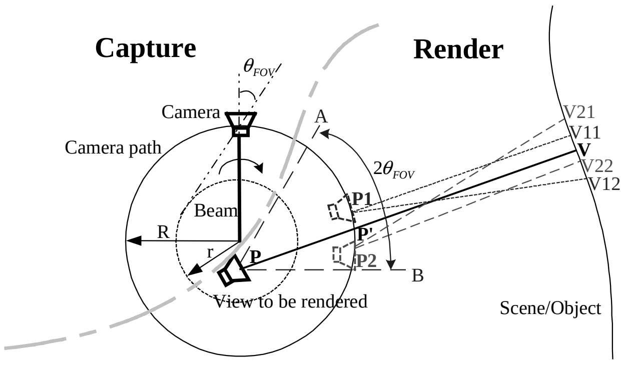

Figure 1: Concentric mosaics capturing and rendering.

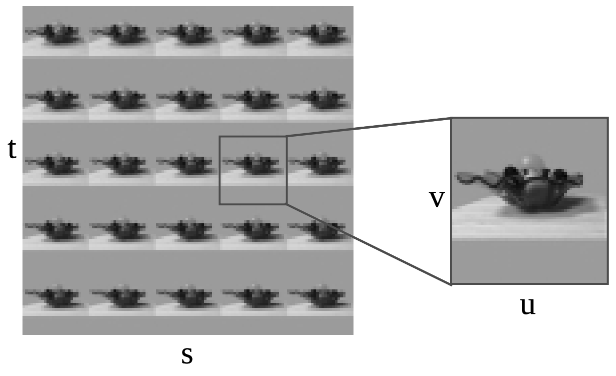

Figure 2 2D image array of Lumigraph/Lightfield.

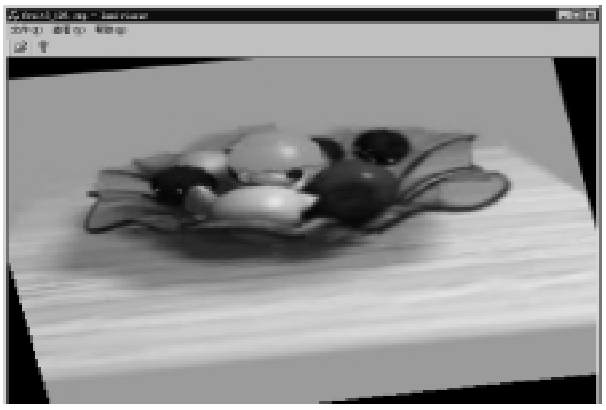

Figure 3 Running scene of the multiple reference frame (MRF) Lumigraph render [14]. The scene is Fruit, compressed at 296:1.

Figure 4 The running scene of the reference block coder (RBC) concentric mosaic render [15]. The scene is Lobby, Compressed at ratio 100:1.