International Journal of Image and Graphics, Vol. 1, No. 1 (2001) 1–17

ON THE COMPRESSION OF IMAGE BASED RENDERING SCENE: A COMPARISON AMONG BLOCK, REFERENCE AND WAVELET CODERS

JIN LI*, HEUNG-YEUNG SHUM† and YA-QIN ZHANG‡
Microsoft Research China, 3F, Beijing Sigma Center,
49 Zhichun Road, Haidian Beijing 100080, P. R. China

In image based rendering (IBR), a 3D scene is recorded through a set of photos which is then rendered to form novel views. 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 wavelet coder. We examine both the compression efficiency and the capability to render the compressed IBR bitstream in real-time. 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 wavelet coder achieves the best compression ratio.

Keywords: Image Based Rendering (IBR); Compression; Block Coding; Reference Coding; Wavelet

1. Introduction

Image-based rendering (IBR) has attracted much attention recently in realistic scene/object representation. A major thread of IBR work has been based on the Plenoptic function proposed by Adelson and Bergen,[1] in which a 3D dynamic scene is modeled using 7D plenoptic function or 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

*E-mail: jinl@microsoft.com
†E-mail: hshum@microsoft.com
‡E-mail: yzhang@microsoft.com

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 set is involved in IBR. As an example, the Lumigraph scene *Fruit* (shown in Fig. 1) consists of a 32×32 array of images with resolution 256×256. The total raw data amount is 196 MB. The concentric mosaic scene *Lobby* (shown in Fig. 2) consists

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

Fig. 2. The running scene of the reference block coder (RBC) concentric mosaic render.[15] The scene is Lobby, compressed at ratio 100:1.

of 1350 images with resolution 320×240. The total raw data amount is 297 MB. 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, IBR scene bears unique characteristics which leads 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 IBR than in 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 image rays, certain HVS properties such as spatial and temporal masking may not be used in IBR compression because neighboring pixel in IBR dataset may not be rendered as neighboring pixel in the final view. 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 them only when the contents are 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 algorithms, the block coder the reference coder and the 3D 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 the concentric mosaics in Sec. 2. The compression systems are then examined in Sec. 3. Performance comparison of representative systems is shown in Sec. 4. A conclusion is given in Sec. 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

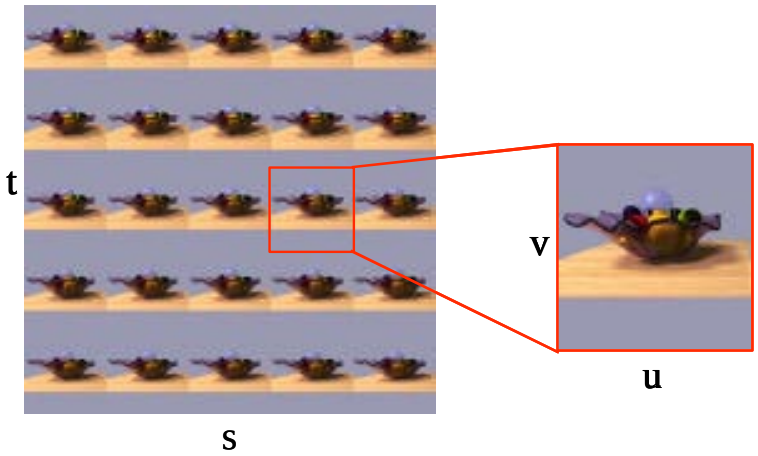

Fig. 3. 2D image array of Lumigraph/Lightfield.

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 co-ordinate (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 Fig. 3. 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. As shown in Fig. 4, 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 $r = R_{\sin}\theta_{\mathrm{FOV}}$. 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

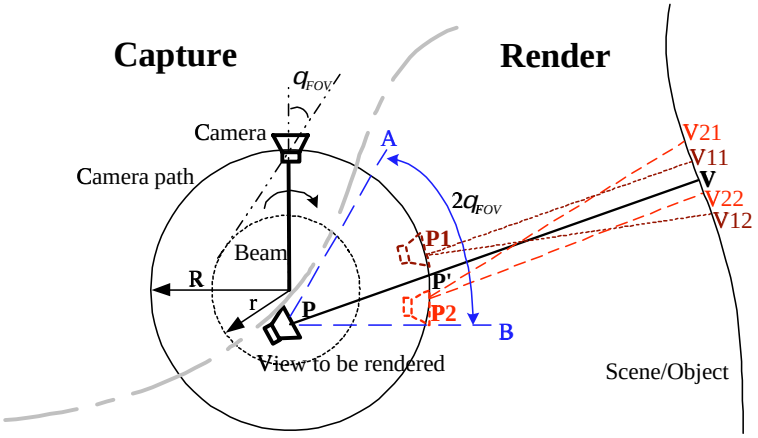

Fig. 4. Concentric mosaics capturing and rendering.

from that is observed in $P'V$.[a] 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. Alternatively, a point sampling mode may be used where the closest slit in the photo set is used to reconstruct $P'V$.

3. IBR Compression Approaches

In this section, we investigate three categories of IBR compression approaches and comment on their pros and cons.

3.1. *The block coder*

We first examine the block coder, a category of coders that segment the IBR dataset into blocks (2D or 3D) and encode each block into fixed length bitstream. We restrict each block to be encoded into fixed length so that the output bitstream can be easily indexed. A common technology used in block coder is the spatial vector quantization (SVQ) which was used in Ref. 4 to compress blocks of the Lightfield and in Ref. 5 to compress blocks of the concentric mosaics. SVQ chops the IBR photo set into small blocks either 2D blocks within frames,[4] or 3D blocks across frames,[5] and then encodes each block with a spatial domain vector quantizer. The operation of SVQ can be illustrated in Fig. 5. Each subblock is matched with an entry in the lookup table and the index of the closest entry is recorded as the coding result. At the time of the decoding, the index is used to lookup the representative entry from the lookup table and to reconstruct the original sub-block. SVQ is simple

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

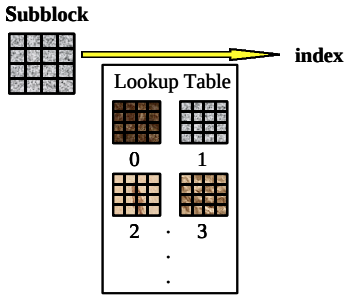

Fig. 5. Spatial vector quantization.

to decode and the just-in-time (JIT) accessing of SVQ compressed bitstream is easy as SVQ index is encoded with equal length. The main weakness of the SVQ is that the compression ratio is limited. Compression ratio of 6:1 to 23:1 is achieved in Ref. 4 and 12:1 is achieved in Ref. 5 for reasonable quality IBR scene. In Ref. 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. *The 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

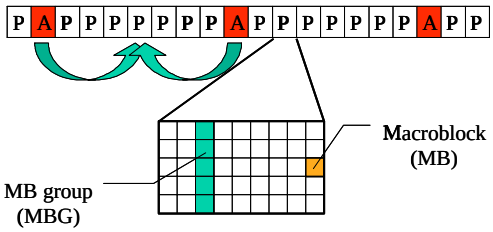

Fig. 6. Framework of the reference block coder (RBC).

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 Light-field/Lumigraph. Similar work termed the reference block coder (RBC) is proposed in Ref. 14 to compress the concentric mosaics. A model-aided coding approach is proposed by Magnor, Eisert and Girod,[18] where five images (one polar, and four on the equator) are independently coded, and the rest images are encoded with model-aided prediction from the reference image.

We use the RBC as an example to explain the reference coder. The framework of the RBC can be shown in Fig. 6. Each block marked A or P is an image frame in the concentric mosaic. RBC classifies 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 two nearby A frames.

Both A and P frame are segmented into square blocks of size 16×16. We call it a macroblock (MB) for its similarity with the macroblock used in JPEG and MPEG. All MBs at the same vertical position of a shot are grouped together and form a macroblock group (MBG) which is the smallest unit of accessing and decoding.

The MBs in A frame are independently encoded. Each MB is split into six 8×8 subblocks with four of which luminance subblocks and the other two chrominance subblocks which are sub-sampled by a factor of 2 in both the horizontal and vertical direction. The sub-blocks are transformed by a basis-8 discrete cosine transform (DCT), quantized by an intra Q-table with a quantization scale Q_A, and then entropy encoded by a run-level Huffman coder with an A frame Huffman table. The compressed bitstreams of all MBs belong to the same A frame MBG are then grouped together.

MBs in the P frames are predictively encoded with reference to a nearby A frame. The P frame may refer to two nearby A frames, however, for a single MB in the P frame, it only refers to one of the two. In fact, we restrict all MBs in a single MBG to refer to the same A frame. This restriction reduces the amount of accessed data when a slit in the P frame MBG is accessed. Since the concentric mosaic frames are shot by swinging a single camera mounting on a beam, the motion between

two concentric mosaic images is predominantly horizontal translation with little to no vertical motions. A two-stage motion estimation including a global translation motion and a local refinement motion is applied to calculate the motion vector of each MB. The camera motion between shots is modeled by a global horizontal translation motion. We do not use more complex model such as affine or perspective model because the camera moves only in a very small interval between shots with dominant translation motion and more complex motion model is not justified. The dominant global horizontal translation vectors mv_1 and mv_2 of the P frame with regard to the two referring A frames are calculated and recorded. The vector mv will be used to reduce the search range and the entropy of the P frame MB motion vector. The individual refinement motion vector of the P frame MB is restricted to $+/-5$ pixels of the global translation vector mv with half pixel accuracy because we know that most MBs just move along the underlying P frame. In fact, around half of the local refinement motion, vectors of the P frame MBs are zeros. To encode a P frame MBG, we encode the MBG against both reference A frames. The MBG is split into a number of MBs. For each MB, its best match is searched in the two reference A frames. Since the search is restricted, it can be performed very fast. The prediction residue of the MB is then split into six sub-blocks with each subblock transformed by a basis-8 DCT, quantized by scale Q_P, and then run-level Huffman coded with a P frame Huffman table. After all the MBs in the MBG are encoded, the rate and distortion of MB codings with reference to the two nearby A frames are compared. The one that offers a better rate-distortion trade off is selected as the reference frame for the MBG. One bit is encoded for each MBG to identify the reference A frame. After that, the motion vector and the prediction residue of the MBs are grouped together and encoded.

The RBC coding scheme bears a strong resemblance to MPEG. In fact, the MB of an A frame and the MB prediction residue of a P frame are encoded exactly the same way as those in MPEG. However, RBC has a very different frame structure, motion model, and bitstream syntax from that of MPEG. In contrast to MPEG which is a general-purpose video codec, RBC is tuned specifically for the compression of the concentric mosaics. Unlike MPEG where a predicted P frame can refer to another predicted P frame, the P frame in RBC refers only to an A frame. MPEG allows strong motion for each MB while the motion model in RBC is predominantly global horizontal translation with only small local variation for individual MB due to parallax. The two-level hierarchical index table is also unique for RBC. The modifications enhance the compression performance of RBC and enables the RBC compressed bitstream to be randomly accessed at the rendering stage.

3.3. *The 3D wavelet coder*

Since IBR is a static photo set, high dimensional transform especially the high dimensional wavelet can be used to compress the data set. 3D wavelet coder has already been developed to compress the concentric mosaics.[13,15,16] 4D Haar wavelet

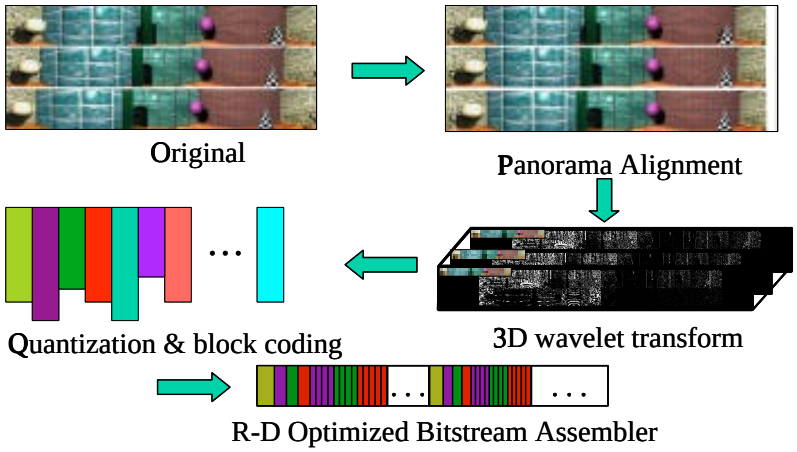

Fig. 7. Framework of smartly rebinned 3D wavelet coder.

with SPIHT has also been used to compress the Lightfield.[19] High dimensional 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 thus very suitable for the Internet application. In the following, we use the smartly rebinned 3D wavelet concentric mosaic coder as an example to explain the 3D wavelet coder.[16]

The framework of a smartly rebinned 3D wavelet coder can be shown in Fig. 7. First, the mosaic image array is aligned to enhance its cross mosaic correlation. Then, the aligned mosaics are decomposed by 3D wavelet transform. After that, the wavelet coefficients are cut into cubes and each cube is quantized and compressed independently into an embedded bitstream. Finally, a global rate-distortion optimizer is used to assemble the bitstream.

The key of a successful 3D wavelet coder is the mosaic image alignment. Without smart rebinning alignment, a direct 3D wavelet transform and coding approach offers only a comparable performance to that of MPEG-2. This is due to the fact that filtering in the temporal direction (mentioned as cross shot filtering, as there is no time domain in the concentric mosaic) has not been very efficient and thus the compression performance of the 3D wavelet codec suffers. We will explain the smart rebinning process in the following. More details can be found in Ref. 16.

Since the concentric mosaics assume static scenery and the camera is slowly swinging within a planar circle, the motion between two successive images is predominantly horizontal translation with little or no vertical motion. We can easily calculate the horizontal translation vector between each pair of consecutive shots. The proposed smart rebinning then aligns the mosaic images according to their motion with two steps.

In the first step, we maximize the correlation between neighboring shots by horizontally aligning them according to the calculated displacement vector as shown in Fig. 8. We use seven concentric mosaic image shots $\mathbf{F}_0, \mathbf{F}_1, \ldots, \mathbf{F}_6$ as an example. Each shaded horizontal line in Fig. 8 corresponds to one captured image. The vertical direction of the image is not shown since we are concerned only with horizontal translation. An additional virtual image $\mathbf{F}_0$ is drawn right after the last image $\mathbf{F}_6$ to show the circular capturing activity of the camera.

In the second step, we further cut and paste (i.e., to rebin) the skewed dataset into panoramas by pushing the skewed data volume downward in Fig. 8 and form smartly rebinned panoramas. We illustrate this step in Fig. 9. Let the horizontal displacement vectors between frames be $x_0, x_1, \ldots, x_{N-1}$. The original shots are divided into groups of vertical slits according to the horizontal displacement vectors which are called stripes. As shown in Fig. 9, frames are aligned according to the horizontal displacement vectors. The frame boundaries are shown as dashed lines and stripes are the segments between the dashed lines. The stripe is the smallest

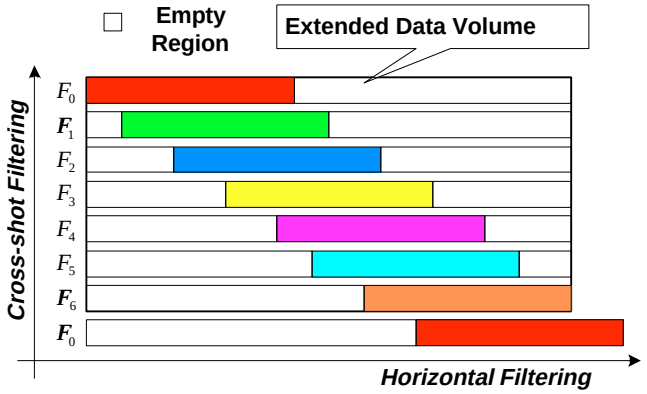

Fig. 8. Horizontal shot alignment of the concentric mosaic image shots.

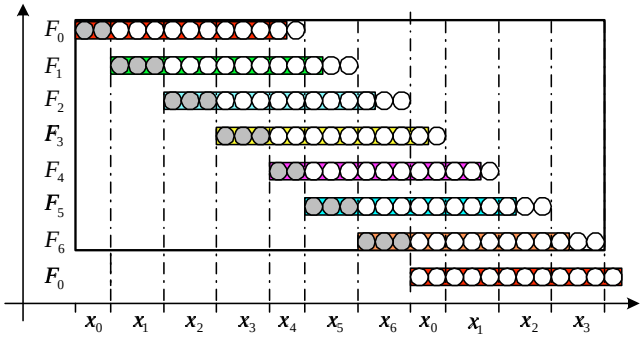

Fig. 9. The smart-rebinning process.

integral unit in the smart rebinning. Let the stripe be denoted as $s_{n,j}$ where n indexes the image shot $\mathbf{F}_n$ that the stripe belongs to, and j indexes the stripe within $\mathbf{F}_n$. The length of the first stripe $s_{n,0}$ is x_n, i.e., the horizontal displacement vector between frame $\mathbf{F}_n$ and $\mathbf{F}_{n+1}$. The length of the jth stripe $s_{n,j}$ is $x_{(n+j) \bmod N}$ correspondingly. The number of stripes is not constant for all frames; it is inversely proportional to the horizontal displacement vector. We then downward stack the stripes and form the rebinned panorama set. The right part of the data volume is also warped to the left due to the circular nature of the camera shots. Let the maximum number of stripes for all frames be S. A total of S panoramas are obtained with equal horizontal length $x_0 + x_1 + \cdots + x_{N-1}$. The first rebinned panorama P_0 is constructed by concatenating the first stripes of all frames, i.e., the bottom of the downward-stacked data volume which is shown in Fig. 9 as the trace of the dotted circles. In general, a smartly rebinned panorama $\mathbf{P}_i$ consists of the ith stripes of all frames cut and paste sequentially with the ith stripe of frame F_0 at the ith slot;

$$\mathbf{P}_i = \{s_{(-i) \bmod N,i}, s_{(-i+1) \bmod N,i}, \ldots, s_{(-i+N-1) \bmod N,i}\}, \quad i = 0, 1, \ldots, S-1\,.$$

An illustration of the resultant rebinned panorama is shown in Fig. 10. As shown in Fig. 9, the sample concentric mosaic image array has a total of seven frames with 12 slits each frame. The seven horizontal displacement vectors for the frames are 2, 3, 3, 3, 2, 3 and 3 respectively. There are at most five stripes in any frame. As a result, the mosaic image array is rebinned into five panoramas with width $2+3+3+3+2+3+3 = 19$. The first panorama consisted of the first stripes from all shots. The second panorama consisted of the second stripes from all shots. To align the first and the second panoramas in the cross panorama direction, the second panorama is rotationally shifted so that the stripe from frame $\mathbf{F}_{N-1}$ is at the head. Some portions of the stripes in panorama $\mathbf{P}_4$ contain no data as the corresponding image shot do not have a full 5th stripe. The smart-rebinned panoramas are thus not of rectangular region of support.

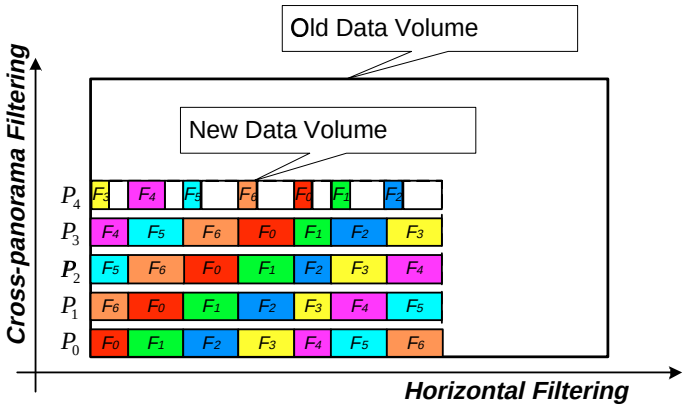

Fig. 10. The smart-rebinned data volume.

With the smart rebinning alignment, the 3D wavelet filtering becomes highly efficient. Filtering across the panorama is efficient because it is exactly equivalent to filtering across the image shots shown in Fig. 8 where image shots are aligned according to their motion. The horizontal filtering is within the rebinned panorama which is highly correlated internally because each stripe consists of successive slits in one original shot image and the two neighbor stripes are smoothly connected because they are from the matching stripes in neighboring concentric mosaic image shots. Consequently, horizontal filtering is also efficient. Compared with direct horizontal alignment shown in Fig. 8, the unfilled regions of the skewed dataset are largely reduced which makes the compression much more efficient and the implementation much more convenient.

For smartly rebinned panoramas, a 3D wavelet coding algorithm that handles a data volume with an arbitrary region of support must be used. Fortunately, there are wavelet algorithms designed to encode arbitrary shaped objects in the literature, mostly developed in the standardization process of MPEG-4.[20] We use an arbitrary shape wavelet coder[21] directly on the irregular region of support. For each directional wavelet transform, a set of straight lines parallel to the axis intersects the supported region and creates several segments. Each segment is then decomposed separately using a bi-orthogonal symmetric filter with symmetric boundary extension into the exact number of wavelet coefficients. We then store the coefficients in the wavelet domain and record the region of support for the wavelet coefficients. The process can be recursively applied for multi-resolution decomposition and can transform the arbitrarily supported concentric mosaic volume into an exact number of wavelet coefficients as that of the original data. For details of the scheme, we refer the reader to Ref. 21. A block arithmetic coder with an arbitrary region of support in the wavelet domain is then used to compress the transformed coefficients.

Rendering of wavelet compressed concentric mosaic 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 concentric mosaic into spatially tiled 3D 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 Ref. 15. According to the current viewing point and direction of the user, the rendering engine generates a set of slits that need to be accessed from the concentric mosaic data set. It then figures out the position of the accessed slits in the rebinned panorama set. After that, the PIWS engine is used to locate the wavelet coefficients in the rebinned panorama set and perform just enough lifting computation to recover the slits used in the current view. A special cache is developed in PIWS to hold the wavelet coefficient, the intermediate lifting result and the recovered pixel all in one memory place, with a state indicating the current progress of the lifting. For more details, we refer to Ref. 15.

4. Experimental Results

Representative block coder, reference coder and 3D wavelet 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 320×240, with a total of 297 MB data. The scene Kids has 1462 frames at resolution 352×288, with a total of 424 MB data. We first compare the compression performance of the three coders. The spatial vector quantizer (SVQ) in Ref. 5 is used to represent the block coder; the reference block coder (RBC) in Ref. 17 is used to represent the reference coder; and the smart rebinning wavelet coder (wavelet) in Ref. 16 is used to represent the high dimensional wavelet coder. The SVQ can only achieve a compression ratio of 12:1 where the compression ratio of the other two coders is at least 60:1. In term of compression performance, it is no match with the RBC or the wavelet coder. The compression performance of RBC and wavelet coder is further compared with a benchmark MPEG-2 video coder and 3D wavelet coder without the smart rebinning process to demonstrate the importance of cross-mosaic alignment. We compress the *Lobby* scene at ratio 200:1 (0.12 bpp, 1.48 MB) and 120:1 (0.2 bpp, 2.47 MB) and the *Kids* scene at 100:1 (0.24 bpp, 4.24 MB) and 60:1 (0.4 bpp, 7.07 MB). The peak signal-to-noise-ratio (PSNR) between the original and decompressed scene of MPEG-2, RBC and wavelet coder without and with smart rebinning is shown in Table 1. We show the PSNR of Y, U and V component. However, it is the PSNR result of Y component that matters most, therefore, we comment only on Y component PSNR in the discussion. It is observed that RBC out-performs MPEG-2 for an average gain of 0.4 dB. With

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

Dataset	Lobby	Lobby	Kids	Kids
Algorithm	0.2 bpp	0.12 bpp	0.4 bpp	0.24 bpp
MPEG-2	Y: 32.2	Y: 30.4	Y: 30.1	Y: 28.3
	U: 38.7	U: 37.4	U: 36.6	U: 34.8
	V: 38.1	V: 36.9	V: 36.7	V: 34.9
RBC	Y: 32.8	Y: 29.8	Y: 31.5	Y: 28.7
	U: 39.7	U: 38.4	U: 39.3	U: 37.3
	V: 40.5	V: 39.0	V: 38.9	V: 36.6
Wavelet without Smart rebinning	Y: 31.9	Y: 30.0	Y: 29.4	Y: 27.3
	U: 40.3	U: 39.3	U: 36.5	U: 34.9
	V: 39.9	V: 38.9	V: 37.2	V: 35.7
Wavelet with Smart rebinning	Y: 36.3	Y: 34.3	Y: 33.8	Y: 31.3
	U: 43.9	U: 42.9	U: 41.1	U: 39.5
	V: 42.8	V: 42.0	V: 41.2	V: 39.6

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

specially designed motion compensation, RBC outperforms a standard video coder such as MPEG. Moreover, the RBC compressed bitstream can be just-in-time (JIT) accessed and rendered, yet, the MPEG compressed bitstream cannot. Without the smart rebinning, the 3D wavelet coder under-performs MPEG-2 for an average of 0.6 dB and under-performs RBC for an average of 1.0 dB. However, with the smart rebinning, the 3D wavelet coder outperforms MPEG-2 for 3.0 to 4.1 dB with an average of 3.7 dB and outperforms RBC for 2.3 to 4.5 dB with an average of 3.2 dB. It shows that in term of compression performance, the 3D wavelet coder is the best and the smart rebinning greatly improves the compression performance of the 3D wavelet coder.

Next, we investigate the rendering speed of SVQ, RBC and 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.

5. Conclusions

In this paper, three categories of image based rendering (IBR) compression algorithms are investigated. The block coders such as SVQ are the least complex in decoding and can easily achieve just-in-time (JIT) rendering. The reference coders such as the reference block coder (RBC) achieve a good comprise between the compression efficiency and the real-time rendering capability. The high dimensional wavelet coders such as the smartly rebinned 3D wavelet coder achieve the best compression performance. They are the most complex coders. However with effort, real-time JIT rendering is achievable.

References

1. E. H. Adelson and J. R. Bergen, "The plenoptic function and the elements of early vision," in *Computational Models of Visual Processing*, eds. M. Landy and J. A. Movshon (The MIT Press, Cambridge, Mass., 1991), Chapter 1.

2. L. McMillan and G. Bishop, "Plenoptic modeling: An image-based rendering system," *Comput. Graphics (SIGGRAPH'95)*, pp. 39–46 (August 1995).
3. S. J. Gortler, R. Grzeszczuk, R. Szeliski and M. F. Cohen, "The Lumigraph," *Computer Graphics (SIGGRAPH'96)*, pp. 43–54 (August 1996).
4. M. Levoy and P. Hanrahan, "Light field rendering," *Computer Graphics (SIGGRAPH'96)*, p. 31 (August 1996).
5. H. Shum and L. He. "Rendering with concentric mosaics," *Computer Graphics (SIGGRAPH'99)*, pp. 299–306 (August 1999).
6. M. Kiu, X. Du, R. Moorhead, D. Banks and R. Machiraju, "Two-dimentional sequence compression using MPEG," *SPIE: Visual Communication and Image Processing (VCIP'98)*, pp. 914–921 (January 1998).
7. M. Magnor and B. Girod, "Adaptive block-based light field coding," *Proc. 3rd Int. Workshop on Synthetic and Natural Hybrid Coding and Three-Dimensional Imaging IWSNHC3DI'99*, Santorini, Greece, pp. 140–143 (September 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, pp. 281–292 (June 1998).
9. M. Magnor and B. Girod, "Model-aided light field coding," in *SPIE: Visual Communication and Image Processing (VCIP'2000)* **4067**(2), Perth, Australia (June 2000).
10. M. Magnor and B. Girod, "Hierarchical coding of light fields with displarity maps," *Proc. Int. Conf. Image Proc. (ICIP-99)*, Kobe, Japan, pp. 334–338 (October 1999).
11. Microsoft Document, "Creating compressed textures," http://msdn.microsoft.com/library/psdk/directx/ddover_2jef.htm.
12. J. Li and J. Kuo, *IEEE Trans. Circuit and System for Video Technology* **7**(2), 440 (April 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 Comm. and Processing* **3974**(10), San Jose CA, pp. 89–100 (January 2000).
14. C. Zhang and J. Li, "Compression of Lumigraph with multiple reference frame (MRF) prediction and just-in-time rendering," in *Proc. IEEE Data Compression Conference (DCC'2000)*, Snowbird, Utah, pp. 254–263 (March 2000).
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)* **4067**(4), Perth, Australia (June 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 (October 2000).
17. C. Zhang and J. Li, "Compression and rendering of concentric mosaics with reference block codec (RBC)", in *SPIE: Visual Comm. and Image Processing (VCIP'2000)* **4067**(5), Perth, Australia (June 2000).
18. M. Magnor, P. Eisert and B. Girod, "Model-aided coding of multi-viewpoint image data," *Proc. Int. Conf. Image Processing (ICIP-2000)*, Vancouver, Canada (September 2000).
19. M. Magnor and B. Girod, "Model-based coding of multi-viewpoint imagery," *Proc. Visual Comm. and Image Processing 2000 (VCIP'2000)*, Perth, Australia (June 2000).
20. MPEG-4 Video Verification Model 14.2, *ISO/IEC JTC1/SC29/WG11 5477*, Maui (December 1999).
21. J. Li and S. Lei, "Arbitrary shape wavelet transform with phase alignment," *Proc. Int. Conf. Image Processing*, Chicago, IL (October 1998).

Photos and Bibliography

Jin Li (M'94-SM'99) received his M.S. and Ph.D. in Electrical Engineering from Tsinghua University, Beijing, China in 1991 and 1994 respectively.

From 1994 to 1996, he served as a Research Associate at the University of Southern California. Later on, from 1996–1999, he was a member of the technical staff at the Sharp Laboratories of America (SLA). He joined Microsoft Research, China on 1999 where he is currently a Researcher/Project Leader. From 2000, Dr. Li has also served as a Guest Professor at the Electrical Engineering Department, Tsinghua University. He is an active contributor for the ISO JPEG 2000/MPEG4 project and has published over 50 refereed papers in leading international conferences and journals relating to image/video compression and multimedia communication. Dr. Li is an Associate Editor for the *Journal of Visual Communication and Image Representation*. He is a senior member of IEEE. He has won 1994 Ph.D. thesis award from Tsinghua University and in 1998, Young Investigator Award from SPIE *Visual Communication and Image Processing*.

Heung-Yeung Shum received his Ph.D. in Robotics from the School of Computer Science, Carnegie Mellon University in 1996. He worked as a Researcher for three years in the vision technology group at Microsoft Research Redmond. In 1999, he moved to Microsoft Research China where he is currently a Senior Researcher and the Assistant Managing director. His research interests include computer vision, computer graphics, pattern recognition, and robotics.

Ya-Qin Zhang is the Managing Director of Microsoft Research China, leaving his post as the Director of Multimedia Technology Laboratory at Sarnoff Corporation in Princeton, NJ (formerly David Sarnoff Research Center, and RCA Laboratories). He has been engaged in research and commercialization of MPEG2/DTV, MPEG4/VLBR, and multimedia information technologies. He was with GTE Laboratories Inc. in Waltham, MA and Contel Technology Center in Virginia from 1989 to 1994. He has authored and co-authored over 200 refereed papers in leading international conferences and journals. He has been granted dozens of US patents in digital video, Internet, multimedia, wireless and satellite communications. Many of the technologies he and his team developed have become the basis for start-up ventures, commercial products, and international standards. He serves on the Board of Directors of five high-tech IT companies.

Zhang served as the Editor-in-Chief for the IEEE Transactions on Circuits and Systems for Video Technology from July 1997 to July 1999. He was the Chairman of Visual Signal Processing and Communications Technical Committee of IEEE Circuits and Systems. He serves on the Editorial Boards of seven other professional journals and over a dozen conference committees. He has been a key contributor to the ISO/MPEG and ITU standardization efforts in digital video and multimedia.

Zhang is a Fellow of IEEE. He received his B.S. and M.S. in Electrical Engineering from the University of Science and Technology of China (USTC) in 1983 and 1985. He received his Ph.D. in Electrical Engineering from George Washington University, Washington D.C. in 1989. He completed the executive business program from Harvard University.