Compression of Concentric Mosaic Scenery with Alignment and 3D Wavelet Transform

Lin Luo[a], Yunnan Wu[a], Jin Li[b], Ya-Qin Zhang[b],

[a] University of Science & Technology of China
Hefei, Anhui Province, 230026, P.R.China

[b] Microsoft Research China
5F Research, Sigma Ctr, 49 Zhichun Road, Haidian, Beijing, 100080, P.R.China
Email: {jinl, yzhang}@microsoft.com

ABSTRACT

As a new scene representation scheme, the concentric mosaic offers a quick way to capture and model a realistic 3D environment. This is achieved by shooting a lot of photos of the scene. Novel views can be rendered by patching vertical slits of the captured shots. The data amount in the concentric mosaic is huge. In this work, we compress the concentric mosaic image array with a 3D wavelet scheme. The proposed scheme first aligns the mosaic images, and then applies a 3D wavelet transform on the aligned mosaic image array. After that, the wavelet coefficients in each subband are split into cubes, where each of the cubes is encoded independently with an embedded block coder. Various cube bitstreams are then assembled to form the final compressed bitstream. Experimental result shows that the proposed 3D wavelet coder achieves a good compression performance.

Keywords: Image based rendering, concentric mosaic, compression, 3D wavelet, embedded coding, rate-distortion optimization.

1. INTRODUCTION

Image based rendering distinguishes itself as a promising tool in the constant pursuit of computer generated photo-realism. Using plenoptic function[1], image based rendering models a 3D scene by recording the light ray at every space location and pointing toward every possible direction. The plenoptic function of a full 3D scene is of 5 dimensions, with 3 dimensions for the viewing position and another 2 dimensions for the pointing direction. Lightfield[2] and Lumigraph[3] are two examples of the 4D plenoptic functions, which capture the complete appearance of a 3D scene/object if the viewer/object can be bounded in a box. Shum and He[4] propose COncentric Mosaic (COM), a 3D plenoptic function restricting viewer movement on a plane. The COM has the ability to easily construct a realistic 3D walkthrough by rotating a single camera at the end of a beam, with the camera pointing outward and shooting images as the beam rotates. When a novel view is rendered, we just split the view into vertical ray slits and search for each slit its counterpart in existing shot images. The technology is termed concentric mosaic because the data structure is a stack of mosaic images along different radiuses.

Though easy to create 3D walkthrough, the amount of data associated with the concentric mosaic is huge. As an example, a concentric mosaic of 240 pixels in height, 1350 pixels in circular (horizontal resolution), and 320 pixels in radius (depth resolution, which provides the function of 3D view) occupies a total of 297 mega bytes (MB). In [4], a vector quantization was used to compress the COM scene with a compression ratio of 12:1. The compressed data is 25MB, which is still far too large considering that it will take one hour to download a COM scene over a 56kbps modem connection. We may compress each individual shot of COM with a high performance still image coder, such as JPEG or JPEG 2000[8]. Because a COM scene consists of multiple highly correlated shots, this may not be very efficient. An alternative approach is to treat a COM scene as a video and compress it with a video coder, such as MPEG[11]. The approach does not take the random access requirement into consideration, and thus is not practical for COM rendering. Moreover, though the PSNR

performance of the MPEG coder may be satisfactory, the result COM scene may not be of high quality in the rendering stage, because video sequence where MPEG is optimized is played continuously, while the COM scene is viewed statically. Consequently, a highly efficient compression is indispensable for the application of concentric mosaic.

In this work, we compress the image array of the concentric mosaic with 3D wavelet transform. The algorithm is separated into four functional blocks: the alignment, 3D wavelet transform, scalar quantizer & embedded block coder, and bitstream assembler. First, the image array is aligned so that the mosaic images looks as similar as possible. The aligned images are then decomposed by a 3D lifting operation, which reduces the required memory for the forward and inverse transform of the huge COM image array. Various 3D wavelet packets are also investigated in this stage. After that, the wavelet coefficients are cut into cubes, and each cube is then compressed independently into an embedded bitstream. Finally, a global rate-distortion optimizer is used to assemble the bitstream. There do exist several 3D wavelet coding algorithms for video[6][7]. The proposed algorithm differs from prior schemes by providing a more memory efficient lifting implementation and a block coding structure so that the COM scene can be easily accessed locally and displayed with different resolution in the rendering stage.

The paper is organized as follows. The data structure of the concentric mosaic and its rendering scheme are introduced in Section 2. In section 3-6, we explain in details the functions of the alignment, 3D wavelet, quantization & entropy coding, and bitstream assembler. Experimental results are shown in Section 7. A conclusion is given in Section 8.

2. THE CONCENTRIC MOSIAC SCENE

A concentric mosaic (COM) scene is obtained by mounting a single camera at the end of a beam, and shooting images at regular intervals as the beam rotates, as shown in Figure 1. Let the camera shots taken during the rotation be denoted as $c(n,w,h)$, where n indexes the camera shot, w indexes the horizontal position within a shot, and h indexes the vertical position. Let N be the total number of camera shots, W and H be the horizontal and vertical resolution of each camera shot, respectively. In the COM scene, the original image data are rearranged into mosaic images, where the mosaic image $\boldsymbol{F(w)}=\{f(w,n,h)|n,h\}$ consists of vertical slits at position w of all camera shots. Image $\boldsymbol{F(w)}$ can be considered as taken by a virtual slit camera rotating along a circle co-centered with the original beam with a radius $r=R\sin q$, where R is the radius of the rotating beam, r is the equivalent radius of the slit camera, and q is the angle between ray w and the camera normal. Since images $\boldsymbol{F(w)}$ are a set of concentric mosaic images with different radius, the scene representation is termed concentric mosaic.

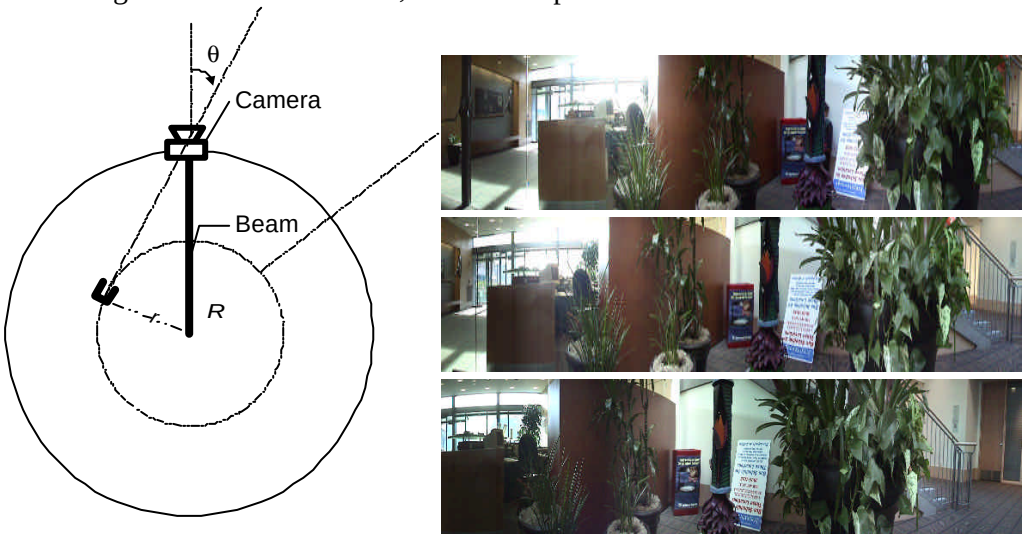

Figure 1 Illustration of the concentric mosaic.

Let the horizontal field of view of the camera be *FOV,* the COM representation can render any view pointing to any direction within the circle $R \times sin(FOV/2)$, as shown in Figure 2. Let (p,b) be the location of a new view point in polar coordinates, let the view be separated into a set of vertical slits, with one slit pointing to the direction of a. With elementary geometry, it would not be difficult to show that the viewing slit is equivalent to a slit on the concentric mosaic circle with radius $p \times sin(b-a)$ and angular coordinate $(p/2+a)$. The process is shown in Figure 2. By collecting multiple ray slits of the new view in the above way, the rendered image can be shown.

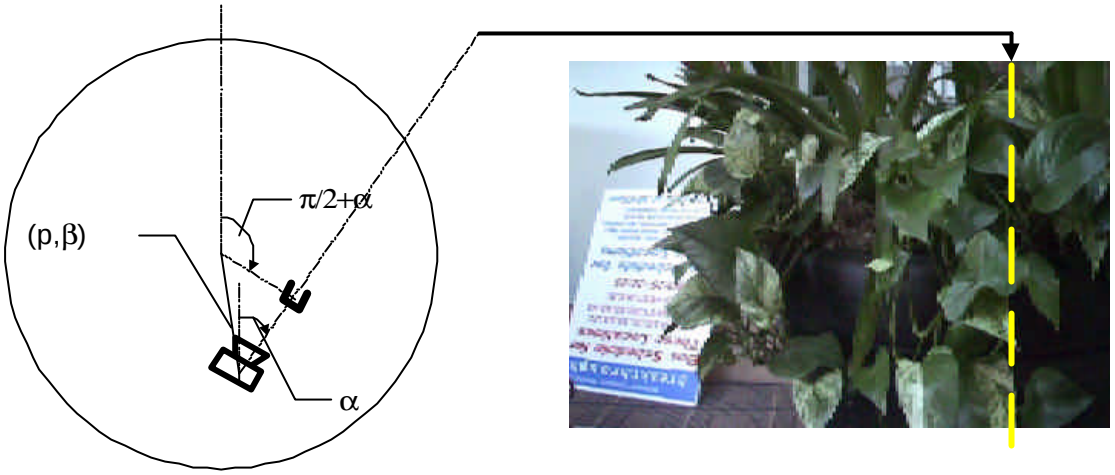

Figure 2 Rendering of the concentric mosaic.

Thus, by recording all the mosaic images ***F(w)***, we equivalently capture the dense 3D walkthrough view (the COM scene) within the circle $R \times sin(FOV/2)$. A single mosaic image ***F(W/2)*** provides the center panorama of the scene which enables the viewer to rotate at the center of camera track, and it is the rest of the mosaic images that supply the additional information required by the 3D walkthrough. The original data of the COM scene is three-dimensional, with strong correlation among different mosaic images. We therefore developed a 3D wavelet scheme for the compression of the COM scene. The four individual function blocks of the COM scene compression - the alignment, 3D wavelet transform, scalar quantizer & embedded block coder, and the bitstream assembler will be described in details in the following sections.

3. ALIGNMENT

The first step in our proposed compression framework is to align the mosaic images so that correlation among different mosaic images is maximized. The alignment can be performed implicitly, considering only the capturing process; alternatively, it may be performed explicitly, by rotationally matching two adjacent mosaic images. The implicit match takes into consideration that parallel ray slits of a faraway object appears similar. Thus the mosaic image is aligned: $g(w,n,h)=f(w,n-D(w),h)$ where all rays $g(*,n,h)$ point to the same direction. The alignment factor $D(n)$ can be calculated as:

$$\Delta(w) = \frac{N}{2p}\arctan\left[\frac{2(w-W/2)}{W}\tan\frac{FOV}{2}\right] \tag{1}$$

Alternatively, we may explicitly match two consecutive mosaic images and reduce the mean absolute error (MAE) or the mean square error (MSE) between the two rotationally shifted mosaic images:

$$MAE(w) = \frac{1}{N \cdot H}\sum_{n=1}^{N}\sum_{h=1}^{H}\left| f(w, n-\Delta(w), h) - f(w-1, n-\Delta(w-1), h)\right| \tag{2}$$

$$MSE(w) = \frac{1}{N \cdot H}\sum_{n=1}^{N}\sum_{h=1}^{H}\left[f(w, n-\Delta(w), h) - f(w-1, n-\Delta(w-1), h)\right]^2 \tag{3}$$

The consecutive displacement $D(w)-D(w-1)$ that minimizes the matching MAE or MSE is the relative alignment factor between mosaic images. By setting the alignment of a specific mosaic image to zero, e.g., that of the 0^{th}

mosaic image *D(0)=0*, we may derive the absolute alignment factors of other images. Experiments have been performed and results show that the compression performance of explicit alignment outperforms that of implicit alignment. Therefore, we use the explicit alignment in the following. The alignment factor *D(n)* is recorded in the compressed bitstream.

4. 3D WAVELET TRANSFORM

In the next step, a 3D separable wavelet transform is applied on the concentric mosaic (COM) image array to decorrelate the images in all three dimensions, and to compact the energy of the image array into a few large coefficients. In addition to energy compaction, the multi-resolution structure provided by the 3D wavelet transform may also be used to access a reduced resolution mosaic image array in the rendering, in case that there is not enough bandwidth or computation power to access the full resolution COM scene, or the display resolution of the client device is low.

The entire COM scene is too large to be loaded into memory simultaneously to perform the 3D wavelet transform. For the sake of memory saving and computational simplicity, a 3D lifting scheme with frame/line buffer has been implemented. A sample one-dimension bi-orthogonal 9-7 lifting wavelet is illustrated in Figure 3. Corresponding to the three dimensions, there are three different lifting units – the frame lifting, the line lifting and the horizontal lifting. The original data are fed into the lifting unit one element at a time, where a single element is one mosaic image for the frame lifting, one line for the line lifting, or a single pixel for the horizontal lifting. The 4 stage forward lifting, shown in the left of Figure 3, can be formulated below:

$$\begin{cases} y_1(2i+1) = x(2i+1) + a\left[x(2i) + x(2i+2)\right] \\ y_2(2i) = x(2i) + b\left[y_1(2i-1) + y_1(2i+1)\right] \\ h(i) = y_3(2i+1) = y_1(2i+1) + c\left[y_2(2i) + y_2(2i+2)\right] \\ l(i) = y_4(2i) = y_2(2i) + d\left[y_3(2i-1) + y_3(2i+1)\right] \end{cases}, \tag{4}$$

where *x(i)* is the original data, $y_s(i)$ is the *s*th stage lifting, *h(i)* and *l(i)* are the high and low pass coefficients, respectively. The coefficients of the 9-7 biorthogonal lifting are *a=-1.586, b=-0.052, c=0.883* and *d=0.444*. The elementary operation of lifting is *Y=(L+R)*d+X*, which is depicted in the right of Figure 3. Since the elementary operation can be easily inversed as *X=Y-(L+R)*d*, the inverse of the lifting can be easily derived, as shown in the middle Figure 3.

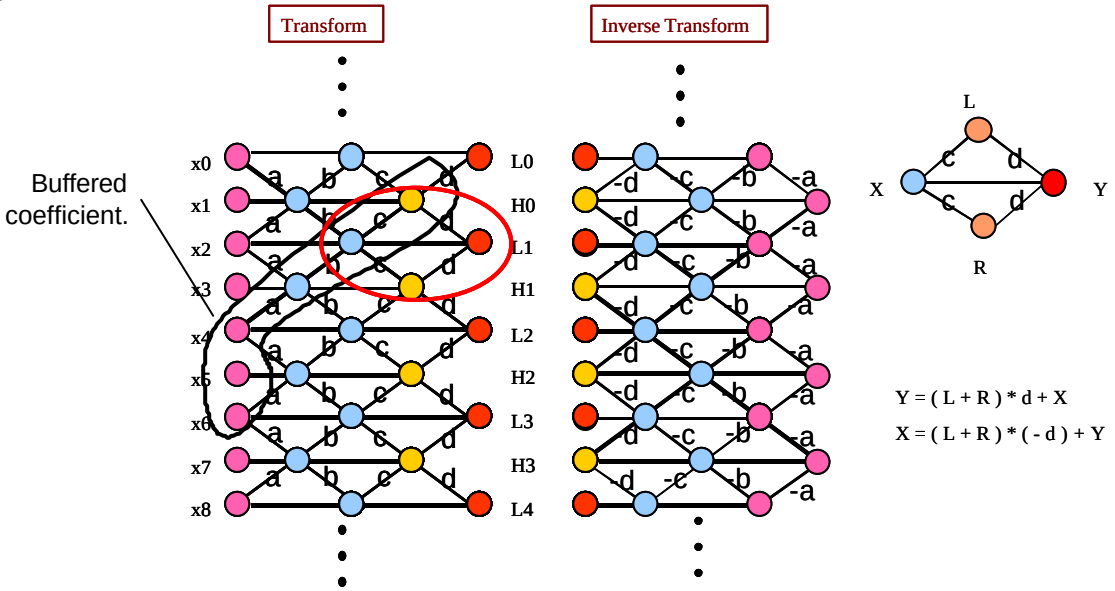

Figure 3 One dimensional forward and inverse lifting operation.

The elementary operation of the lifting consists of 2 additions and 1 multiplication operation. The average computation load is thus 4 additions and 2 multiplications per coefficient. In contrast, the average computation of the traditional convolution implementation of the same 9-7 biorthogonal wavelet is 8 additions and 4.5 multiplications, which more than doubles the computation load. The original data X is no longer needed once the coefficient Y is calculated; therefore, the result Y may be stored at the same memory unit that holds X. Such in-place calculation may be used to reduce the memory required for the lifting operation. In fact, in the 9-7 biorthogonal lifting, only 6 elements need to be buffered. Shown in the circle of Figure 3, three elements are intermediate results of the lifting, and the other three are original coefficients. For every two input data points, four elementary lifting operations are performed and one low pass and one high pass coefficients are output. The required memory buffer to perform the 3D lifting is thus 6 frames for the frame lifting, and 6 lines for the line lifting. A single scale 3D lifting is illustrated in Figure 4, where frame lifting is performed first, then line lifting, and finally horizontal lifting.

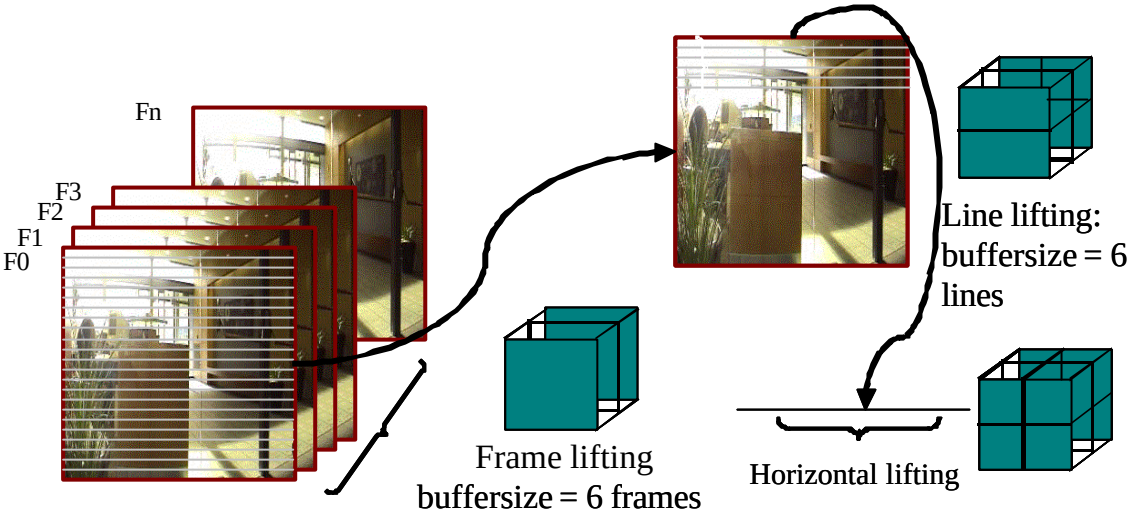

Figure 4 Single scale three-dimension lifting.

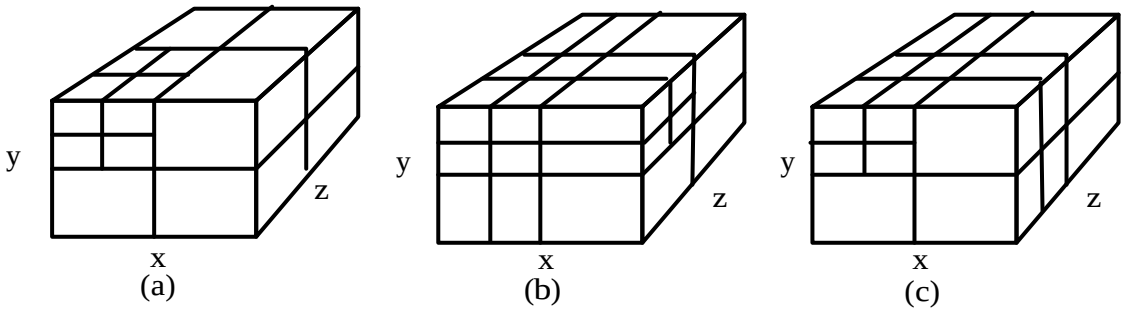

Figure 5 Multiple scale 3D lifting, (a) two-level mallat in all directions, (b) two-level x decomposition + two-level (y,z) mallat, (c) two-level z decomposition + two-level (x,y) mallat.

In the COM scene coding, more than one single scale lifting operation is usually performed, thus some of the result wavelet subbands may be further decomposed. We may choose to decompose only the low pass band of the frame, line and horizontal lifting, such decomposition being called the mallat decomposition. A two-level full mallat decomposition is depicted in Figure 5(a), where the lifting along the x, y and z axes is the frame, line and horizontal lifting, respectively. Alternatively, we may first perform wavelet decomposition along one axis, and then decompose the plane spanned by the other two axes. For example, we may first perform a two-level decomposition along the x axis, and then a two-level full mallat decomposition on the (y, z) plane; the result is

shown in Figure 5(b). Another lifting configuration shown in Figure 5(c) first performs a two-level z axis decomposition, and then a two-level full mallat decomposition on the (x, y) plane. All such decompositions are forms of wavelet packet, and the adopted wavelet packet structure will be recorded in the compressed COM bitstream.

5. SCALAR QUANTIZATION AND EMBEDDED BLOCK CODING

The wavelet transformed coefficients are chopped into cubes, which are compressed by a scalar quantizer and embedded block coder. The compressed block bitstreams are first buffered, and then assembled by a rate-distortion optimized assembler after all blocks have been encoded. Even though the quantization and entropy coding are performed on a block-by-block basis, the wavelet transform operates on the entire concentric mosaic (COM) data, therefore, no explicit blocking artifact is visible in the decoded COM scene. The block coding structure selected for the COM scene compression has the following advantages:

1. Benefit of local statistical variations.

The statistical property may not be homogenous across the entire COM data set. Since each block of coefficients is processed and encoded independently, the encoder may tune to local statistical property, and thus improve coding performance. The variation of the statistics across the COM scene may also be used in the bitstream assembler, and bits may be distributed in a rate-distortion optimized fashion across the COM scene.

2. Easy random access.

With the block quantization and entropy coding, we may randomly access a portion of the COM scene without decoding the entire data set. From the accessed region required by the rendering unit, we may derive the related blocks using the wavelet basis and wavelet decomposition scheme. Only the bitstreams of the accessed blocks are decoded.

3. Low memory requirements.

The block structure also waives the need to buffer the entire volume of COM coefficients. Only K frames of coefficients need to be buffered for each subband, where K is the size of the block in frame direction. Once K frames of coefficients arrived, they are chopped into blocks, quantized and entropy encoded. We still buffer the compressed bitstreams of the coefficients, however, they are much smaller. The decoder side requires even less memory since the compressed bitstream is only partially accessed and decoding is only performed on those blocks related to render current view.

The quantizer is a simple scalar quantizer with step size Q and a dead zone $2Q$. The forward and inverse quantizer can be formularized as:

$$q = \begin{cases} \lfloor w/Q \rfloor & w > 0 \\ 0 & w = 0 \\ \lceil w/Q \rceil & w < 0 \end{cases} \qquad w' = \begin{cases} (q+0.5)Q & q > 0 \\ 0 & q = 0 \\ (q-0.5)Q & q < 0 \end{cases}, \tag{5}$$

where w and w' are the original and reconstructed coefficients, q is the quantizer output, $\lfloor x \rfloor$ and $\lceil x \rceil$ are the floor and ceiling functions, respectively. Since the bitstream assembler is used, the quantization step size Q no longer controls the final coding quality. We may simply choose a small constant quantization step size Q, such as $Q=1.0$ to ensure that the COM scene is compressed with sufficient quality before assembling.

Many implementations of the entropy encoder are feasible. Due to the use of the bitstream assembler, the entropy coder must have the embedding property, i.e., the compressed block bitstream can be truncated at a later stage with a good compression performance at such reduced bitrate. Three different embedded entropy coders are implemented with different complexity vs. performance tradeoff. They are all bitplane coders. Let the block under consideration be denoted by C, let $b(i,l)$ be the lth most significant bit of the absolute value of coefficient i. Let $\boldsymbol{B}_l$ be the lth bit plane, which consists of all bits at the same significance level l. Let L be the total number of

bitplanes, where it is satisfied that for all coefficients x in C, $|x|<2^L$, and there is at least one coefficient x so that $|x|>=2^{L-1}$. All three coders encode the block bitplane by bitplane, and all of them iterate from the most significant bitplane $L-1$ to the least significant bitplane 0. If the compressed bitstream is truncated later, at least the most significant bitplanes of all coefficients are accessible, and therefore, the block can still be decoded with good quality. For each coefficient x_i at a certain bitplane k, if all bits in prior bitplanes are 0, i.e., $b(i,l)=0$ for all $l>k$, the coefficient is called insignificant, and significant vise versa. For each bitplane, the bits of insignificant coefficients are encoded in significance identification mode, while the bits of significant coefficients are encoded in refinement mode. The bit in the refinement mode appears uniformly as 0 or 1, and hence leaves less room for compression. Due to the energy compaction property of wavelet transform, the bit in the significance identification mode skews highly toward 0, therefore, it is the task of significance identification to locate the coefficient which turns significant in current bitplane, i.e., those bits that satisfy $b(i,k)=1$ *and* $b(i,l)=0$ for all $l>k$. The proposed coders differ primarily in how the significance identification is performed.

During the embedded coding, the coding rate $R(l)$ and distortion $D(l)$ of the block are recorded at the end of each bitplane. The rate $R(l)$ can be easily derived from current encoding bitstream length. The distortion $D(l)$ may be calculated by measuring the difference between the original and reconstructed coefficients at current stage. A look-up table as the one presented in [8] may be used to speed up the distortion calculation. Alternatively, we may estimate the distortion $D(l)$ with technology presented in the rate-distortion optimized embedded coder (RDE)[10]. Calculating distortion $D(l)$ is of course more accurate and is beneficial for the coding performance, however, the estimation of the distortion $D(l)$ is much faster and introduces less overhead. The recorded rate-distortion performance of the block will be used in the bitstream assembler.

The details of the three implemented block entropy coder are described below:

5.1. TREE CODER

In the tree coder, the insignificant coefficients are grouped by oct-tree, while the significant coefficients are split into individual pixels. Tree coding is efficient because large area of insignificant bits are grouped together and represented with a single '0'. As the coder iterates from the most significant bitplane to the least significant bitplane, the oct-tree of insignificant coefficients gradually split and the locations of the significant coefficients are identified. The procedure of the tree coder is as follows:

Step 1. Initialization
Four lists are established, which are the list of insignificant sets (LIS), the list of candidate sets (LCS), the list of insignificant pixels (LIP), and the list of significant pixels (LSP). LIS is consisted of sets where all coefficients are insignificant; LCS is consisted of sets where all coefficients are insignificant but at least one coefficient will be significant in this bitplane; LIP is consisted of single insignificant coefficients; and LSP is consisted of single significant coefficients. Elements in the four lists may move according to Figure 6. Elements in LCS may split and move to LIS, LSP, LIP or form another element of LCS. However, LIS element can only move to LCS; LIP element can only move the LSP; and once an element moves to LSP, it stays there.

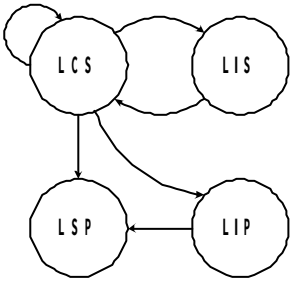

Figure 6 State transition of the tree coder.

Step 2. Bitplane coding.
In each bitplane coding, we first examine sets in LIS one by one. If at least one coefficient becomes significant in this bitplane, a '1' is encoded and the set moves to LCS. Otherwise, a '0' is encoded and the set remains in LIS. Then the sets in LCS are examined. For a set c in LCS, since it is guaranteed that at least one coefficient will become significant in this bitplane, the set splits along the three axes into 8 sub-sets: $c_1, .., c_8$. If set c is of size 2x2x2, 8 child pixels are generated. For each subset/pixel c_i, a status bit s_i is established which is '0' if the subset/pixel is insignificant, and is '1' otherwise. The 8-bit status string $s=\{s_1,. . .,s_8\}$ is Huffman encoded. The subset/pixel c_i is placed into LCS/LSP if the status bit is '1', and into LIS/LIP if the status bit is '0'. If pixel c_i moves into LSP, its sign is encoded. After all sets in LCS have been processed, all coefficients in LIP in the beginning of current bitplane coding are examined. If the coefficient becomes significant, a '1' is encoded followed by the sign of the coefficient, and the coefficient moves to LSP. Otherwise, a '0' is encoded. Finally, all coefficients in LSP in the beginning of current bitplane coding are refined.

Although conceptually the tree coder encodes the block bitplane by bitplane, in actual implementation, a tree is built for the coefficient block and all subsequent examination is performed on that tree. The computational complexity of the tree coder is thus the lowest among the three.

5.2. GOLOMB-RICE (GR) CODER

The GR coder identifies the significant coefficient with the adaptive binary Golomb-Rice (ABGR) coder, which is a simplified run-length coder. More details of the ABGR coder can be referred to [9]. For better compression performance, we classify the insignificant coefficients further into two categories, i.e., the predicted significant bits where at least one of the 26 nearest neighbors are significant, and the predicted insignificant bits where all the 26 neighbors are insignificant. In each bitplane coding, we first encode the predicted significant bits, then the predicted insignificant bits, and finally the refinement bits. Such coding order is empirically determined by the rate-distortion property of different bit sets. We thus scan the coefficient block three times, each time with a serpentine scanning order that alternates between left-to-right and right-to-left visiting of a row of coefficients. The predicted significant and insignificant bits are encoded by independent ABGR coders with separate parameters. The refinement bits are not encoded and are just sent directly to the block bitstream. The ABGR coder is used to encode the run to next coefficient that will become significant in predicted significant or insignificant scan. If the run is greater than or equal to 2^m, a '*0*' is encoded to represent a zero run of 2^m, while the rest of the run is further examined. Otherwise, a bit '*1*' is encoded followed by m bits of the binary representation of the run length and the sign of the significant coefficient. In essence, the ABGR coder is a Huffman coder that assigns 1 bit for a run greater than or equal to 2^m, and $m+1$ bits for a run smaller than 2^m. m is an adaptive parameter in the ABGR coder, and is determined by the state transition table specified in [12].

5.3. CONTEXT-BASED ARITHMETIC CODER

The context-based arithmetic coder resembles that of the GR coder. The block is still encoded by bitplanes, and each bitplane is still scanned three times, as predicted significance, predicted insignificance and refinement. However, in the predicted significant and predicted insignificant scan, the insignificant bit is encoded with an arithmetic coder using a context derived from the 26 significant statuses of the neighbors of current coefficient. The 26 significant statuses are grouped into categories so that the number of context is reduced to avoid context dilution. The technique bears resemblance to the one used in JPEG 2000[8]. The coder yields the best compression performance, however, it is also computationally most expensive since context calculation is relatively time consuming.

6. RATE-DISTORTION OPTIMIZED BITSTREAM ASSEMBLER

After all the blocks of coefficients have been entropy encoded, a bitstream assembler is used to optimally allocate the bits among different blocks. We have obtained the rate-distortion curves of individual blocks during the embedded coding stage. The block distortion is further multiplied by the energy weight of the subband:

$$D_{Weighted} = w \cdot D_{Original} \tag{6}$$

where the subband weight w is the gain (energy) of the lifting synthesis function. For the bi-orthogonal 9-7 filter, the gain for the low pass filter is $w_L=1.299$, and the gain for the high pass filter is $w_H=0.787$. For a wavelet subband with n low pass lifting and m high pass lifting, the energy weight of the subband can be calculated as $w = w_L^n w_H^m$, no matter whether the lifting is performed along the frame, line or horizontal axes.

The rate-distortion theory indicates that optimal coding performance can be achieved if all blocks operate on the same rate-distortion curve. The functionality of the bitstream assembler is thus to find the common rate-distortion slope of all blocks, and calculate the included bits for each block. The two functionalities are performed below:

1. Find the truncation bit rate for each block

Given a rate-distortion slope, the coding rate of each block is determined by the portion of the bitstream with operating rate-distortion slope greater than the given slope. In essence, we find the tangent on the rate-distortion curve of the block that is equal to the given rate-distortion slope, and the operating bit rate at this tangent point is the coding bitrate for this block.

2. Finding the optimal common rate-distortion slope

The optimal common rate-distortion slope is found through a bi-section method. We first set two rate-distortion slope threshold l_{min} and l_{max}, where the optimal rate-distortion slope is enclosed in the interval (l_{min}, l_{max}). A current threshold $l=(l_{min}+l_{max})/2$ is calculated. Given current threshold l, each block is examined and the operating bitrate is calculated. Current coding rate is the sum of the coding rate of all blocks. Depending on whether current coding rate is larger or smaller than the desired rate, the lower or upper limit of the threshold is updated accordingly. The search stops after a number of iterations, or as current coding rate is close to the desired rate with a certain percentage. During the search of the optimal coding rate, only the rate-distortion function of each block is examined, no block needs to be re-encoded. Therefore, the search can be performed pretty fast.

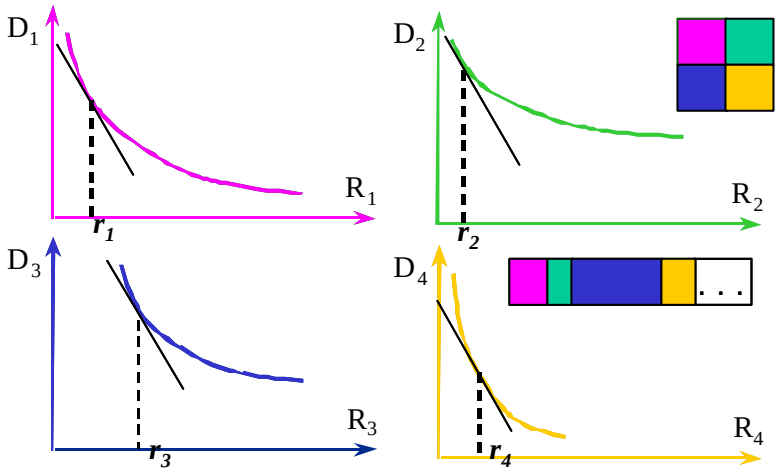

Figure 7 Illustration of the bitstream assembling

The process is illustrated in Figure 7, where the scene consists of 4 blocks. The weighted rate-distortion curve of each block is calculated during the embedded coding stage. We search for an optimal rate-distortion slope which is tangent with the rate-distortion curves of all blocks. The block bitstream is then truncated at the tangent point, and the truncated bitstream segments of different blocks are assembled together to form the compressed bitstream. The rate-distortion slope is adjusted so that output bitrate is equal or close to the desired bitrate.

7 EXPERIMENTAL RESULTS

The performance of the concentric mosaic (COM) coding with 3D wavelet is demonstrated with experimental results. We also investigate the efficiency of various 3D wavelet packet transforms and block entropy coding schemes. One test data set is the COM scene Lobby (Figure 8), which is shot with 1350 frames at resolution 320x240. Forming the mosaic images, the Lobby scene comprises of 320 mosaic images of resolution 1350x240. Another test data set is the scene Kids (Figure 9), which comprises of 352 mosaic images of resolution 1462x288. All scenes are in the YUV color space, with U and V components subsampled by a factor of 2 in both the horizontal and vertical direction. The Lobby scene is compressed at 0.2 bpp (bit per pixel) and 0.4bpp, respectively. The Kids scene has more details, and is thus compressed at 0.4bpp and 0.6bpp, respectively. The objective peak signal-to-noise ratio (PSNR) is measured between the original COM scene and the decompressed scene:

$$PSNR = 10\log_{10}\frac{255^2}{\frac{1}{N \cdot W \cdot H}\sum\left[f(w,n,h) - f'(w,n,h)\right]^2}, \qquad (7)$$

where $f(w,n,h)$ and $f'(w,n,h)$ are the original and reconstructed COM scene, respectively. The PSNR of the Y, U and V components are all presented in the experiment, though it is the PSNR of the Y component that matters most, as around *90%* of the bitstream is formed by the compressed bitstream of the *Y* component.

In the first experiment, different 3D wavelet packet decompositions are investigated. Let lifting along the *x*, *y* and *z* axes correspond to the frame, line and horizontal lifting, respectively. Five wavelet packet decomposition structures are evaluated:

Structure A: 4-level mallat in all 3 directions (frame, line and horizontal),
Structure B: 4-level decomposition along *y* axis, followed by 3-level *(x, z)* mallat decomposition,
Structure C: 4-level decomposition along *y* axis, followed by 4-level *(x, z)* mallat decomposition,
Structure D: 5-level decomposition along *z* axis, followed by 4-level *(x, y)* mallat decomposition,
Structure E: 5-level decomposition along *x* axis, followed by 4-level *(y, z)* mallat decomposition.

The wavelet coefficients within each subband are chopped into blocks and are encoded by the tree coder with block size set to 16x16x16. The test data set is the Kids scene compressed at 0.6 bpp. The results are listed in Table 1. We observe from Table 1 that a full mallat decomposition may not yield an optimal compression result. By further decomposing some bands along the *y* (line) axis with structure C, compression performance improves by about 0.5dB. Comparing wavelet packet structure C and D, it is observed that further decomposing existing wavelet subbands can improve compression performance slightly. However, we do observe that the gain achieved by further wavelet decomposing beyond 4-level is rather limited. We may choose to first decomposing along the *z* (frame) or *x* (horizontal) axis with structure D and E, however, the performance is inferior to the wavelet packet structure C. Therefore, in the following experiment, the wavelet packet structure C is used, i.e., first a 4-level line lifting is applied, followed by a 4-level mallat lifting in both the frame and horizontal direction.

Table 1 COM scene compression with different wavelet packet decomposition structures.

Wavelet Packet Structure	A	B	C	D	E
PSNR: Y (dB)	30.6	31.0	31.1	30.9	30.3
PSNR: U (dB)	36.4	37.1	37.3	38.4	37.6
PSNR: V (dB)	37.2	37.9	38.0	38.9	38.1

In the second experiment, we investigate the performance of various block entropy coders with block size 16x16x16. We also compare the performance of 3D COM scene coding with that of MPEG-2, which treats the entire COM scene as a video and compressed it with a MPEG-2 codec downloaded from www.mpeg.org. In MPEG-2, only the first image is encoded as I frame, the rest images are encoded as P frames. The MPEG-2 is used here as a benchmark, however we do note that random access using MPEG-2 is not so straightforward, making it not suitable for real time rendering. Moreover, though the PSNR performance of the MPEG coder is satisfactory, there is no guarantee of the quality of the rendered COM scene, as PSNR of the MPEG coded video usually fluctuates along the sequence. The results are shown in Table 2. It is observed that the performance of the tree coder is very close to that of the Golomb-Rice coder. Note that the computation complexity of the tree coder is lower than the Golomb-Rice coder, while the memory requirement of the tree coder is slightly higher, especially at high bitrate. Since the block encoder operates only on a 3D block of size 16x16x16, the memory consumed by the tree coder is limited. We thus favor the tree coder over the Golomb-Rice coder. The more complicated arithmetic coder improves the compression performance by 0.5dB. Comparing MPEG-2 with 3D wavelet COM scene compression by arithmetic coder, we observe that the 3D wavelet coder loses by 0.3dB on average. The performance is still comparable to that of MPEG-2. We may improve the performance of the 3D wavelet COM coder with further tuning. Additionally, the resolution scalability offered by the 3D wavelet and the quality scalability offered by the block embedded coder makes the 3D wavelet COM scene codec attractive in a number of environments, such as Internet streaming and browsing.

Table 2. 3D concentric mosaic scene compression results.

	LOBBY (0.4bpp)	LOBBY (0.2bpp)	KIDS (0.4bpp)	KIDS (0.6bpp)
MPEG-2 (dB)	Y: 34.8 U: 39.9 V: 39.1	Y: 32.2 U: 38.7 V: 38.1	Y: 30.1 U: 36.6 V: 36.7	Y: 31.9 U: 38.0 V: 38.1
3D wavelet + tree coder (dB)	Y: 34.5 U: 41.6 V: 41.2	Y: 31.4 U: 40.1 V: 39.7	Y: 29.0 U: 35.8 V: 36.6	Y: 31.1 U: 37.3 V: 38.0
3D wavelet + Golomb-Rice coder (dB)	Y: 34.4 U: 41.5 V: 41.2	Y: 31.3 U: 40.0 V: 39.7	Y: 29.0 U: 35.7 V: 36.5	Y: 31.0 U: 37.0 V: 37.9
3D wavelet + arithmetic coder (dB)	Y: 35.0 U: 41.9 V: 41.5	Y: 31.9 U: 40.3 V: 39.9	Y: 29.4 U: 36.5 V: 37.2	Y: 31.5 U: 38.0 V: 38.6

8. CONCLUSION

A new approach for the compression of concentric mosaic (COM) scenery using 3D wavelet transform is presented in this paper. The proposed algorithm consists of 4 functional blocks: alignment, lifting wavelet decomposition, scalar quantizer & block entropy coder, and bitstream assembler. The compression performance of the 3D wavelet COM codec is comparable to that of MPEG-2. Moreover, the 3D wavelet codec offers the resolution and quality scalability which may be very useful in the Internet environment.

REFERENCES

[1] L. McMillan and G. Bishop, "Plenoptic modeling: an image-based rendering system", *Computer Graphics Proceedings, Annual Conference series (SIGGRAPH' 95)*, pp. 39-46, Aug. 1995.

[2] M. Levoy and P. Hanrahan, "Light field rendering", *Computer Graphics Proceedings, Annual Conference series (SIGGRAPH' 96)*, pp. 31-42, New Orleans, Aug. 1996.

[3] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph", *Computer Graphics Proceedings, Annual Conference series (SIGGRAPH' 96)*, pp. 43-54, New Orleans, Aug. 1996.

[4] H.-Y. Shum and L.-W. He. "Rendering with concentric mosaics", *Computer Graphics Proceedings, Annual Conference series (SIGGRAPH'99)*, pp. 299-306, Los Angeles, Aug. 1996.

[5] A. Said, "A new fast and efficient image codec based on set partitioning in hierarchical trees", *IEEE Trans. on Circuit and System for Video Technology*, Vol. 6, No. 3, pp. 243-250, Jun. 1996.

[6] A. Wang, Z. Xiong, P. A. Chou, and S. Mehrotra, ``3D wavelet coding of video with global motion compensation," *Data Compression Conference (DCC'99)*, Snowbird, UT, Mar. 1999.

[7] D. Taubman and A. Zakhor, "Multirate 3-D subband coding of video", *IEEE Trans. on Image Processing*, Vol. 3, No. 5, pp. 572-588, Sept. 1994

[8] Verification model ad-hoc, "JPEG 2000 verification model 5.0", *ISO/IEC JTC1/SC29/WG1/N1420*, Oct. 1999.

[9] E. Ordentlich, M. Weinberger, and G. Seroussi, "A low-complexity modeling approach for embedded coding of wavelet coefficients", *Data Compression Conference (DCC' 98)*, Salt Lake City, Utah, Mar. 1998.

[10] J. Li and S. Lei, "An embedded still image coder with rate-distortion optimization", *IEEE Trans. On Image Processing*, Vol. 8, No. 7, pp.913-924, Jul. 1999.

[11] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. LeGall, *MPEG video compression standard*. New York: Chapman and Hall, 1996.

[12] JPEG-LS AHG, "JPEG-LS part 2 CD version 1.0", *ISO/IEC JTC1/SC29/WG1N1557*, Maui, Hawaii, Dec. 1999.

Figure 8 Concentric mosaic scene Lobby.

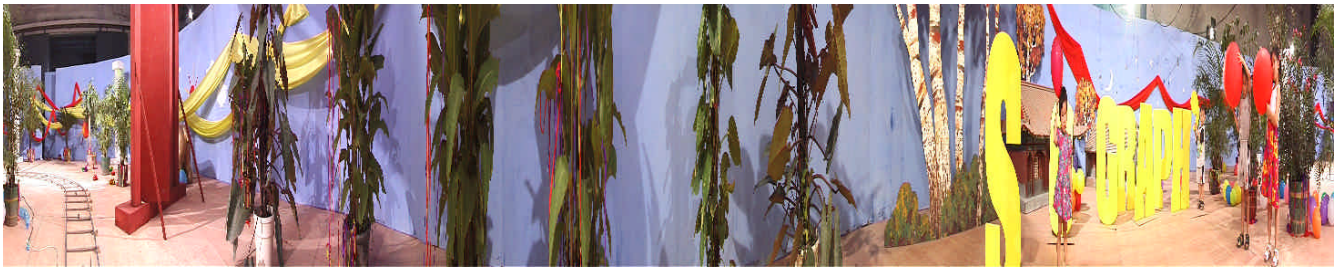

Figure 9 Concentric mosaic scene Kids.