Smart Rebinning for Compression of Concentric Mosaics

Yunnan Wu[a], Cha Zhang[b], Jin Li[c], Jizheng Xu[a]

[a]University of Science & Technology of China, Heifei 230026, China

[b]Electronic Engineering Department, Tsinghua University, Beijing 100084, China.

[c]Microsoft Research, China, 5F Research, Sigma Ctr, 49 Zhichun Road, Haidian, Beijing 100080, China.

Contact Email: jinl@microsoft.com

ABSTRACT

Concentric mosaics offer a quick solution to construct a virtual copy of a real environment, and navigate in the virtual environment. However, the huge amount of data associated with concentric mosaics is a heavy burden for its application. A 3D wavelet transform-based compressor has been proposed in previous work to compress the concentric mosaics. In this paper, we greatly improve the performance of the 3D wavelet coder with a data rearrangement mechanism called "smart rebinning". The proposed scheme first aligns the concentric mosaic image shots along the horizontal direction and then rebins the shots into multi-perspective panoramas. Smart rebinning greatly improves the cross shot correlation and enables the coder to better explore the redundancy among shots. Experimental results show that the performance of the 3D wavelet coder improves an average of 4.3dB with the use of smart rebinning. The proposed coder outperforms MPEG-2 coding of concentric mosaics by an average of 3.7dB.

Keywords

Image based rendering, concentric mosaics, compression, rebinning, multi-perspective panorama, 3D wavelet.

1. INTRODUCTION

Image-based rendering (IBR) techniques have received much attention in the computer graphic realm for realistic scene/object representation. Instead of referring to complicated geometric and photometric properties as the conventional model-based rendering does, IBR requires only sampled images to generate high quality novel virtual views. Furthermore, the rendering time for an IBR dataset is independent of the underlying spatial complexity of the scene, which makes IBR attractive for the modeling of highly complex real environments. Concentric mosaics[1] enable quick construction of a virtual copy of a real environment, and navigation in the virtual environment. By rotating a single camera mounted at the end of a leveled beam, with the camera pointing outward and shooting images as the beam rotates, a concentric mosaic scene can quickly be constructed. At the time of the rendering, we just split the rendered view into vertical ray slits, and reconstruct each slit through similar slits captured during the rotation of the camera.

Though it is easy to create a 3D walkthrough, the amount of data associated with the concentric mosaics is tremendous. As an example, a concentric mosaic scene from [1] includes 1350 RGB images with resolution 320x240 and occupies a total of 297MB. Efficient compression is thus essential for the application of the concentric mosaics. In [1], a vector quantization approach was employed to compress the concentric mosaic scene with a compression ratio of 12:1. However, the size of the compressed bitstream is still 25MB, far too large for storage and transmission. Since the captured concentric mosaic shots are highly correlated, much higher compression ratio should be achievable.

Since the data structure of the concentric mosaics can be regarded as a video sequence with slowly panning camera motion, video compression techniques may be used to compress the concentric mosaics. We consider two major categories of video compression techniques. Existing standards, such as MPEGx and H.26x, adopt a prediction-based framework, where the temporal redundancy across frames is reduced through motion compensation and block residue coding. A reference block coder (RBC) has been proposed with a similar concept to compress the concentric mosaic scene [13].

Along a separate direction, three-dimensional (3D) wavelet video coders[2][3][4][5] present another category of video coding approaches that explore the temporal redundancy via wavelet filtering in the temporal direction. One attractive property of the 3D wavelet video coder is its spatial, temporal and quality scalability. Here the term scalability means that a 3D wavelet coder can compress a video into a single bitstream, where multiple subsets of the bitstream can be decoded to generate complete videos of different spatial resolution/temporal resolution/quality commensurate with the proportion of the bitstream decoded [6]. This is extremely useful in the Internet streaming environment where heterogeneous decoder/network settings prevail. Furthermore, since 3D wavelet based coders avoid the recursive loop in predictive coders, they perform better in an error prone environment, such as a wireless network. In previous work, we developed a 3D wavelet transform coding system to compress concentric mosaics[7]. However, the performance of that coder is inferior to that of RBC and MPEG-2. In this work, we investigate the performance bottleneck and further improve the compression performance of the 3D wavelet concentric mosaic coder.

In a 3D wavelet coder, wavelet transforms are applied separately along the horizontal, vertical and temporal directions to concentrate the signal energy into relatively few large coefficients. However, one common problem with the 3D wavelet compression schemes is that the temporal wavelet filtering does not achieve efficient energy compaction. In a prediction-based video / concentric mosaic coder, local motion can be specified on a per block basis, thus inter-frame correlation due to the moving object/camera can be explored which is very beneficial to the coding performance. However, local motion cannot be easily incorporated into the framework of 3D wavelet compression. Because of the nature of temporal filtering, each pixel can be engaged in one and only one transform.

Taubman and Zakhor [2] proposed a pan compensation module that aligned the image frames prior to the wavelet transform. In our former 3D wavelet concentric mosaic codec[7], a panorama alignment module was used to eliminate global translation. Wang *et al.* [3] proposed to register and warp all image frames into a common coordinate system and then apply a 3D wavelet transform with an arbitrary region of support to the warped volume. To make use of the local block motion, Ohm [4] incorporated block matching and carefully handled the covered/uncovered, connected/unconnected regions. By trading off the invertibility requirement, Tham *et al.* [5] employed a block-based motion registration for the low motion sequences without filling the holes caused by individual block motion. However, both Ohm and Tham's approaches are complex.

In this paper, a smart rebinning operation is proposed as a novel preprocessing technique for the 3D wavelet compression of the concentric mosaics. Rather than adapting the compression algorithm or the filter structure to the mosaic image array, we modify the data structure for easy compression by 3D wavelet. The proposed scheme begins with pair-wise alignment of the image shots. Then the original concentric mosaic scene is rebinned to form multi-perspective panoramas. The rearranged data have much stronger correlation across frames; and thus can be compressed more efficiently by the 3D wavelet coder.

This paper is organized as follows: The background for the acquisition and display of the concentric mosaics is provided in Section 2. The smart-rebinning operation and its rationale and potential benefits to the 3D wavelet codec are detailed in Section 3. Since the smart-rebinned concentric mosaics may no longer be of rectangular region of support, the associated 3D wavelet coding technique is discussed in Section 4. Experimental results are presented in Section 5. Finally, we conclude the paper in Section 6.

2. BACKGROUND: THE CONCENTRIC MOSAICS

A concentric mosaic scene is captured by mounting a camera at the end of a rotating beam, and shooting images at regular intervals as the beam rotates. We show the capturing device in Figure 1. Let the camera shots taken during the rotation of the beam be denoted as $\boldsymbol{F_n}=\{f(n,w,h)|w,h\}$, 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. The entire concentric mosaic database can be treated as a series of camera shots $\boldsymbol{F_n}$, or alternatively be interpreted as a series of rebinned panoramas $\boldsymbol{P_w}=\{f(n,w,h)|n,h\}$ where each individual panorama consists of vertical slits at position w of all camera shots. Three rebinned panoramas at different radii are shown in Figure 2. Panorama $\boldsymbol{P_w}$ can be considered as taken by a virtual slit camera rotating along a circle co-centered with the original beam with a radius $d=R\sin\theta$, where R is the radius of the rotation beam, d is the equivalent radius of the slit camera, and θ is the angle between ray w and the camera normal, which can be calculated as:

$$\theta=\arctan\frac{2w-W}{W}.$$

Since the entire data volume $\boldsymbol{P_w}$, $\boldsymbol{w=0,\cdots,W-1}$ can be considered as a stack of co-centered mosaic panoramas with different radius, it is called the concentric mosaic[1].

Concentric mosaics are able to capture a realistic environment and render arbitrary views within an inner circle of radius $r=R\sin(FOV/2)$, where FOV is the horizontal field of view of the capturing camera. Rendering concentric mosaics involves reassembling slits from the captured dataset. Shown in Figure 3, 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. A basic hypothesis behind the concentric mosaic rendering is that the intensity of any ray does not change along a straight line unless blocked. Thus, when a slit PV is rendered, we simply search for the slit $P'V$ in the captured dataset, i.e., either in the captured image set $\boldsymbol{F_n}$ or the rebinned panorama set $\boldsymbol{P_w}$, where P' is the intersection point between the direction of the ray and the camera track. Because of the discrete sampling, the exact slit $P'V$ might not be found in the captured dataset. The four sampled slits closest to $P'V$ may be P_1V_{11}, P_1V_{12}, P_2V_{21} and P_2V_{22}, where P_1 and P_2 are the two nearest captured shots, P_1V_{11} and P_1V_{12} are the slits closest to P_1V in direction in shot P_1, and P_2V_{21} and P_2V_{22} are closest to P_2V in shot P_2. We may choose only the slit that is closest to $P'V$ from the above four to approximate the intensity of PV. However, a better approach is to use bilinear interpolation, where all four slits are employed to interpolate the rendered slit PV. The environmental depth information may assist the finding of the best approximating slits and alleviate the vertical distortion. More detailed description of the concentric mosaic rendering may be found in [1].

3. SMART REBINNING: A CROSS SHOT DECORRELATION APPROACH

In the previous paper[7], we compress the 3D data volume of the concentric mosaics through the global alignment of the panorama and 3D wavelet coding. However, 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. In recognition of the significant role of motion compensation in the 3D wavelet compression, we look for an efficient decorrelation scheme along the cross-shot direction. 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 non vertical motion. We can easily calculate the horizontal translation vector between each pair of consecutive shots. Let x_n denote the calculated horizontal displacement between shot $\boldsymbol{F_n}$ and $\boldsymbol{F_{n+1}}$. Since the shots are circularly captured, shot x_0 is right next to shot x_{N-1}. We thus denote x_{N-1} as the displacement vector between frame F_0 and F_{N-1}. Note that the horizontal displacement vectors may not be equal for all frames. They are inverse proportional to the distance of the object, i.e., larger for shots with a close-by object, and smaller for shots with the far away background. We can maximize the correlation between neighboring shots by horizontally aligning them according to the calculated displacement vector, as shown in Figure 4. We term this approach horizontal shot alignment. We use 7 concentric mosaic image shots $\boldsymbol{F_0}$, $\boldsymbol{F_1}$, $\cdots$, $\boldsymbol{F_6}$ as an example. Each shaded horizontal line in Figure 4 corresponds to one captured image. The vertical direction of the image is not shown since we are only concerned with horizontal translation. An additional virtual image $\boldsymbol{F_0}$ is drawn right after the last image $\boldsymbol{F_6}$ to show the circular capturing activity of the camera.

After the horizontal shot alignment, the concentric mosaics form a skewed data volume, which may be encoded by a 3D wavelet codec with horizontal, vertical and cross-shot filtering with a non-rectangular (arbitrary) region of support. The correlation across image shots is expected to improve, however, since the resultant data volume is highly sparse and is not rectangular, the compression efficiency may be compromised.

The proposed smart rebinning goes beyond horizontal shot alignment one step further. The idea is to cut and paste (i.e., to rebin) the skewed dataset into panoramas by pushing the skewed data volume downward in Figure 4, and form smartly rebinned panoramas. The details of the smart rebinning operation are shown in Figure 5. Let the horizontal displacement vectors between frames be $x_0, x_1, \cdots, x_{N-1}$. The original shots are divided into groups of vertical slits according to the horizontal displacement vectors, which are called stripes. As shown in Figure 5, 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 integral unit in the smart rebinning. Let the stripe be denoted as $s_{n,j}$, where n indexes the image shot $\boldsymbol{F_n}$ that the stripe belongs to, and j indexes the stripe within $\boldsymbol{F_n}$. The length of the first stripe $s_{n,0}$ is x_n, the horizontal displacement vector between frame $\boldsymbol{F_n}$ and $\boldsymbol{F_{n+1}}$. 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. Therefore, there are few stripes for the frame with a close-by object, and more stripes for that with the faraway background. We then downward stack the stripes and form the rebinned panorama set. We also warp the right part of the data volume 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 $x_0+x_1+\ldots+x_{N-1}$. The first rebinned panorama $\boldsymbol{P_0}$ is constructed by concatenating the first stripes of all frames, i.e., the bottom of the downward-stacked data volume, which is shown in Figure 5 as the trace of the dotted circles. In general, a smartly rebinned panorama $\boldsymbol{P_i}$ consists of the ith stripes of all frames cut and paste sequentially, with the ith stripe of frame F_0 at the ith slot:

$$\boldsymbol{P_i}=\{ s_{(-i) \bmod N,\, i}\ ,\ s_{(-i+1) \bmod N,\, i}\ ,\cdots,\ s_{(-i+N-1) \bmod N,\, i} \},\ i=0, 1,\cdots, S-1.$$

An illustration of the resultant rebinned panorama is shown in Figure 6. As shown in Figure 5, the sample concentric mosaic image array has a total of 7 frames with 12 slits each frame. The 7 horizontal displacement vectors for the frames are 2, 3, 3, 3, 2, 3 and 3 respectively. There are at most 5 stripes in any frame. As a result, the mosaic image array is rebinned into 5 panoramas with width 2+3+3+3+2+3+3=19. The first panorama is consisted of the first stripes from all shots. The second panorama is 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 $\boldsymbol{F_{N-1}}$ is at the head. Some portions of the stripes in panorama $\boldsymbol{P_4}$ 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. Special handling for coding those empty regions will be addressed in the next section.

With the smart rebinning approach, the unfilled regions of the skewed dataset are largely reduced, which makes the compression much more efficient and the implementation much more convenient. Filtering across the panorama is exactly equivalent to filtering across the image shots in the horizontal shot alignment approach shown in Figure 4. However, the horizontal filtering is changed from filtering within an image shot to filtering within the rebinned panorama. The newly generated panorama $\boldsymbol{P_i}$ is highly correlated internally, because each stripe consists of successive slits in one original shot image, and two neighbor stripes are smoothly connected because they are from the matching stripes in neighboring concentric mosaic image shots. Consequently, horizontal filtering is still pretty efficient.

A degenerated approach is to restrict all horizontal translation vectors to be exactly the same:

$$x_0 = x_1 = \cdots = x_{S-1}= x.$$

We call this approach simple rebinning. All image shots now have the same number of stripes. If there are unfilled slits at the last stripe, we simply fill them by repeating the last slit. Rebinning the stripes into panoramas, a set of panoramas with a rectangular region of support is formed. The approach is similar to the formation of the concentric mosaics $\boldsymbol{P_w}=\{f(n,w,h)|n,h\}$ in [1]. The difference lies in that multiple slits are obtained from each shot to generate the rebinned panorama.

We show the volume of the original concentric mosaics in Figure 7. The rebinned concentric mosaics form a cube, with the front view showing a concentric mosaic panorama, the side view a camera shot, and the top view a cross-section slice at a certain height. We then show the smartly rebinned panorama volume in Figure 8 as a comparison. The smartly rebinned panorama forms volume of non-rectangular support, and the black region in Figure 8 identifies the unsupported region. We note that the area with a smaller region of support is closer to the capturing camera, because it has a larger horizontal displacement vector, and thus contains a smaller number of stripes. In comparison with the concentric mosaics, the smartly rebinned panorama appears to be more smooth and natural looking, as it adjusts its sampling density according to the distance of the shot to the object, and maintains a relative uniform object size as seen by the camera. The smartly rebinned panoramas have strong correlation across the panoramas. A set of rebinned panoramas at the same horizontal location is extracted and shown in Figure 9. We observe that most objects in the rebinned panoramas are well aligned. Only a few objects, such as the light bulb at the upper-left corner and the balloons behind the girl, show differences due to the gradual parallax transition among the rebinned panoramas. Such a well aligned data volume can be efficiently compressed by a 3D wavelet transform.

In fact, the smartly rebinned panorama belongs to a general category of *multi-perspective* panoramas that become popular recently in the computer graphic realm, such as manifold mosaics[8], multiple-center-of-projection image[9] and circular projection[10]. Multi-perspective panorama extends the conventional panorama by relaxing the requirement of having one common optical center and allows several camera viewpoints within a panorama. The idea of multi-perspective panorama construction via cutting and pasting stripes was first introduced in [8]. It has also been extended to enable stereo viewing in [10], where the stripes taken from the left side of each image shot generate the right eye panorama and those from the right generate the left eye view. However, in contrast to the work of [8][9] and [10], where only one or two panoramas are generated for their specific graphic application, we generate a whole set of rebinned panoramas to provide a dense representation of the environment, and to efficiently compress the concentric mosaic data set.

4. 3D WAVELET CODING OF REBINNED PANORAMAS

We further encode the rebinned panoramas with a 3D wavelet coder. Though other coders, such as the reference block coder (RBC) in [13] can also be applied, 3D wavelet coding is ideal because better alignment across image shots is more efficiently explored by the 3D wavelet coder. For the simple rebinning, straightforward 3D wavelet encoding may be adopted. In this work, we use a 3D wavelet codec with arithmetic block coding as proposed in our previous paper [7]. The data volume of the concentric mosaics is decomposed by multi-resolution 3D wavelet transform. The wavelet coefficients are then cut into fixed size blocks, embedded encoded, and assembled with a rate-distortion optimization criterion. For details of the 3D wavelet coding algorithm, we refer the reader to [7].

For smartly rebinned panoramas, a 3D wavelet coding algorithm that handles a data volume with an arbitrary region of support must be developed. Fortunately, there are wavelet algorithms designed to encode arbitrary shaped objects in the literature, most developed in the standardization process of MPEG-4[11]. A simple approach is to pad the unfilled arbitrary region of support to the tightest rectangular volume containing it and apply the rectangular 3D wavelet transform and coding algorithm to the padded data volume. In this work, we extend the low-pass extrapolation (LPE) adopted in MPEG4 for the padding work. The unsupported regions are first filled with the average pixel value of the boundary of the supported/unsupported region, and then a low-pass filter is applied in the unsupported region several times. Since in the unsupported region, all pixel values are initialized with the same average value, the effect of the low-pass filter is primarily at the boundary, where a gradual transition is built up. After the wavelet transform, coefficients in the unsupported regions will be mostly zeros, except at the boundary. The padded data volume is then compressed with the 3D wavelet codec described in [7]. Since the number of wavelet coefficients after padding is still more than the number of pixels in the supported region, the padding increases the coding rate, and therefore the compression performance is affected. The advantage is that the padding involves the least change in the 3D wavelet codec, and is very easy to implement. Moreover, although the padding operation adds complexity in the encoder, it does not affect the decoder, which decodes the entire data volume and simply ignores the decoded pixels in the unsupported region.

Another feasible solution is to use an arbitrary shape wavelet transform [12] 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 [12]. A block arithmetic coder with an arbitrary region of support in the wavelet domain is then used to compress the transformed coefficients. We call this codec the 3D arbitrary shape wavelet codec. It is observed that the arbitrary shape wavelet transform and coding is slightly superior in compression performance to padding the unsupported region. However, it is also more complex to implement, as we need to add support of the arbitrary shape region to both the transform and entropy coding module.

The smartly rebinned and 3D wavelet compressed concentric mosaic can be efficiently rendered as well. The rendering engine is very similar to the progressive inverse wavelet synthesis (PIWS) engine that we have proposed in [14]. According to the current viewing point and direction of the user, the rendering engine generates a set of slits which 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 computation to recover the accessed slits. Because smart rebinning can be considered as a preprocessing step of the 3D wavelet coder, the only extra step in rendering the smartly rebinned concentric mosaics is to locate the slits in the rebinned panorama, which can be easily performed with knowledge of the horizontal displacement vectors. The computational complexity of rendering the smartly rebinned concentric mosaics is thus similar to the rendering of 3D wavelet compressed concentric mosaics. With the PIWS engine, a rendering rate of 12 frames per second is achievable, which is fast enough for real time rendering applications.

5. EXPERIMENTAL RESULTS

The performance of the 3D wavelet concentric mosaic compression with smart rebinning is demonstrated with extensive experimental results. The test scenes are *Lobby* and *Kids*. The scene *Lobby* has 1350 frames at resolution 320x240, and the total data amount is 297MB. The scene *Kids* has 1462 frames at resolution 352x288, and the total data amount is 424MB. The *Kids* scene contains more details, and is thus more difficult to compress than the *Lobby* scene. The scenes are first converted from RGB to YUV color-space with 4:2:0 sub-sampling, and then compressed by different coders. 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 is shown as the objective measure of the compression quality. We report the PSNRs of all three color components (Y, U and V) in Table 1, however, it is the PSNR result of the Y component that matters most. Therefore, we comment only on the Y component PSNR in the discussion.

We compare the proposed smart rebinning 3D wavelet coder with three benchmark algorithms. The first algorithm (A) compresses the entire concentric mosaics as a video sequence using a MPEG-2 video codec. The MPEG-2 software is downloaded from www.mpeg.org. In the MPEG-2 codec, the first frame is independently encoded as I frame, and the rest frames are predictively encoded as P frames. The second algorithm (B) is a direct 3D wavelet codec as reported in [7], where we rebin the concentric mosaic image shots into mosaic panoramas, align the panoramas and encode them with the 3D wavelet and arithmetic block coding. The third benchmark algorithm (C) is the reference block coder (RBC) reported in [13]. It is a prediction-based codec tuned for compression of concentric mosaics. We observe that direct 3D wavelet coding of the concentric mosaic scene (algorithm B) is not very efficient; it is 0.3 to 1.0 dB inferior to MPEG-2 with an average of 0.6 dB, and is inferior to the RBC codec with an average of 1.1dB.

We tested three different configurations of the 3D wavelet codec with smart rebinning. In the first configuration (algorithm D), we restrict the horizontal displacement vector between frames to be constant, i.e., the simple rebinning is used. The actual displacement vector is 2 and 3 pixels for the *Lobby* and *Kids* scenes, respectively. The resultant rebinned concentric mosaics form a rectangular panorama volume and are compressed by the exact same 3D wavelet and arithmetic block coder as algorithm B. It is observed that by simply rebinning multiple slits into the panorama, a large compression gain is achieved. In fact, compared with the direct 3D wavelet codec, the PSNR improves between 3.2 to 3.6dB, with an average of 3.5dB. The 3D wavelet coder with simple rebinning outperforms the MPEG-2 concentric mosaic codec by 2.9dB, and outperforms the RBC codec by 2.4dB.

We then apply the full-fledged smart rebinning algorithm. The horizontal displacement vectors are calculated by matching neighborhood concentric mosaic image shots. They are then stored in the compressed bitstream. After the rebinning operation, the bounding volume for the rebinned panoramas is 2832x162x240 for the *Lobby* scene and 5390x149x288 for the *Kids* scene. In the *Lobby* scene, object is of relatively constant depth to the camera, and the unsupported regions occupy only 6% of the bounding volume. However, in the *Kids* scene, 36% of the bounding volume is unsupported. We compress the rebinned panoramas through padding the data volume and applying the same 3D wavelet codec as the one used in the algorithm B and D (denoted as algorithm E); also, we use an arbitrary shape wavelet transform and coefficient coding algorithm (denoted as algorithm F). According to the results shown for algorithm F, the smart rebinning further improves the compression performance over simple rebinning by 0.7 to 1.0 dB, with an average of 0.8dB. The average gain of the arbitrary shape wavelet transform (F) over the padding approach (E) is 0.3dB. Note that the system of algorithm E is very close in complexity to that of the simple rebinning (algorithm D), because both systems use rebinning, rectangular 3D wavelet transform, and arithmetic block coding. The only difference is that algorithm D rebins a fixed number of slits into the panorama, while algorithm E rebins a variable number of slits into the panorama, which is then padded before coding. In terms of PSNR performance, algorithm E outperforms algorithm D by 0.5dB on average. Therefore general smart rebinning with calculated horizontal translation vectors does have an advantage over simple rebinning, where a fixed translation vector is used for all image shots.

Overall, smart rebinning with arbitrary shape wavelet transform and coding is the best performer of the proposed approaches. It outperforms the MPEG-2 concentric mosaic codec by an average of 3.7dB, outperforms the direct 3D wavelet video encoder by 4.3dB, and outperforms the reference block coder by 3.2dB. The PSNR of the smart rebinning compressed *Lobby* at 0.12bpp is even superior to prior concentric mosaic codecs operated at 0.2 bpp. It is 2.1dB superior to the MPEG-2, 2.4dB superior to the direct 3D wavelet, and 1.5dB superior to the RBC compressed scene at 0.2bpp. Since the PSNR of the *Lobby* scene compressed at 0.2bpp is on average 2.1dB higher than the PSNR of the same scene compressed at 0.12bpp, the smart rebinning almost quadruples the compression ratio for the *Lobby* scene. We also observe that the smart rebinning nearly doubles the compression ratio for the *Kids* scene over prior approaches. Considering the huge amount of data of concentric mosaics, and considering the relatively large bitstream even after a high ratio compression has been applied (1.48-7.07MB), smart rebinning is a very effective tool to greatly reduce the amount of data of concentric mosaics.

Algorithm \ Test Dataset		LOBBY (0.2 bpp)	LOBBY (0.12 bpp)	KIDS (0.4 bpp)	KIDS (0.24 bpp)
A	*MPEG-2*	Y: 32.2 U: 38.7 V: 38.1	Y: 30.4 U: 37.4 V: 36.9	Y: 30.1 U: 36.6 V: 36.7	Y: 28.3 U: 34.8 V: 34.9
B	*3D Wavelet*	Y: 31.9 U: 40.3 V: 39.9	Y: 30.0 U: 39.3 V: 38.9	Y: 29.4 U: 36.5 V: 37.2	Y: 27.3 U: 34.9 V: 35.7
C	*RBC*	Y: 32.8 U: 39.7 V: 40.5	Y: 29.8 U: 38.4 V: 39.0	Y: 31.5 U: 39.3 V: 38.9	Y: 28.7 U: 37.3 V: 36.6
D	*Simple rebinning*	Y: 35.5 U: 41.5 V: 40.9	Y: 33.6 U: 40.7 V: 40.2	Y: 32.8 U: 39.3 V: 40.1	Y: 30.5 U: 37.7 V: 38.5
E	*Smart rebinning + padding*	Y: 36.0 U: 41.6 V: 41.0	Y: 34.0 U: 40.9 V: 40.2	Y: 33.4 U: 39.9 V: 41.1	Y: 31.1 U: 38.4 V: 39.6
F	*Smart rebinning +arbitrary shape wavelet codec*	Y: 36.3 U: 43.9 V: 42.8	Y: 34.3 U: 42.9 V: 42.0	Y: 33.8 U: 41.1 V: 41.2	Y: 31.3 U: 39.5 V: 39.6

Table 1 Compression results for the concentric mosaic scenes

6. CONCLUSION AND EXTENSION

A technology termed smart rebinning is proposed in this paper to improve the 3D wavelet compression performance of the concentric mosaics. Through cutting and pasting stripes into a set of multi-perspective panoramas, smart rebinning greatly improves the performance of cross shot filtering, and thus improves the transform and coding efficiency of the 3D wavelet codec. The region of support after smart rebinning may cease to be rectangular, and a padding scheme and an arbitrary shape wavelet coding scheme have been used to encode the resultant data volume of smart rebinning. With the arbitrary shape wavelet codec, smart rebinning outperforms MPEG-2 by 3.7dB, outperforms a direct 3D wavelet coder by 4.3dB, and outperforms the reference block coder (RBC) by 3.2dB on the tested concentric mosaic image scenes. It nearly quadruples the compression ratio for the *Lobby* scene, and doubles the compression ratio for the *Kids* scene.

7. ACKNOWLEDGEMENT

The authors would like to acknowledge the following individuals: Harry Shum, Honghui Sun and Minsheng Wu for the raw concentric mosaic data; and Brandon Schwartz for proofreading the paper.

8. REFERENCES

[1] 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. 1999.

[2] 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.

[3] A. Wang, Z. Xiong, P. A. Chou, and S. Mehrotra, “3D wavelet coding of video with global motion compensation,” *Proc. DCC’99*, Snowbird, UT, Mar. 1999.

[4] J. R. Ohm, “Three-dimensional subband coding with motion compensation”, *IEEE Trans. on Image Processing*, vol. 3, no. 5, pp. 572-588, Sept. 1994.

[5] J. Y. Tham, S. Ranganath, and A. A. Kassim, “Highly scalable wavelet-based video codec for very low bit-rate environment”, *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 1, Jan. 1998.

[6] D. Taubman and A. Zakhor, “A common framework for rate and distortion based scaling of highly scalable compressed video”, *IEEE Trans. On Circuits and Systems for Video Technology*, Vol. 6, No. 4, Aug. 1996, pp. 329-354.

[7] L. Luo, Y. Wu, J. Li, and Y.-Q. Zhang, “Compression of concentric mosaic scenery with alignment and 3D wavelet transform”, *SPIE Image and Video Communications and Processing*, SPIE 3974-10, San Jose, CA, Jan. 2000.

[8] S. Peleg and J. Herman, “Panoramic mosaics by manifold projection”, *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 338-343, San Juan, Jun. 1997.

[9] P. Rademacher and G. Bishop, “Multiple-center-of-projection images”, *Computer Graphics Proceedings, Annual Conference series (SIGGRAPH’98)*, pp. 199-206, Orlando, Jul. 1998.

[10] S. Peleg and M. Ben-Ezra, “Stereo panorama with a single camera”, *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 395-401, Fort Collins, Jun. 1999.

[11] MPEG-4 Video Verification Model 14.2. *ISO/IEC JTC1/SC29/WG11 5477*, Maui, Dec. 1999.

[12] J. Li and S. Lei, “Arbitrary shape wavelet transform with phase alignment”, *Proc. Int’l. Conf. of Image Processing*, Chicago, IL, Oct. 1998.

[13] C. Zhang and J. Li, “Compression and rendering of concentric mosaic scenery with reference block codec (RBC)”, *accepted by SPIE Visual Communication and Image Processing (VCIP 2000)*, Perth, Australia, Jun. 2000.

[14] Y. Wu, L. Luo, J. Li and Y. –Q Zhang, “Rendering of 3D wavelet compressed concentric mosaic scenery with progressive inverse wavelet synthesis (PIWS)”, SPIE Visual Communication and Image Processing (VCIP 2000), Perth, Australia, Jun. 2000.

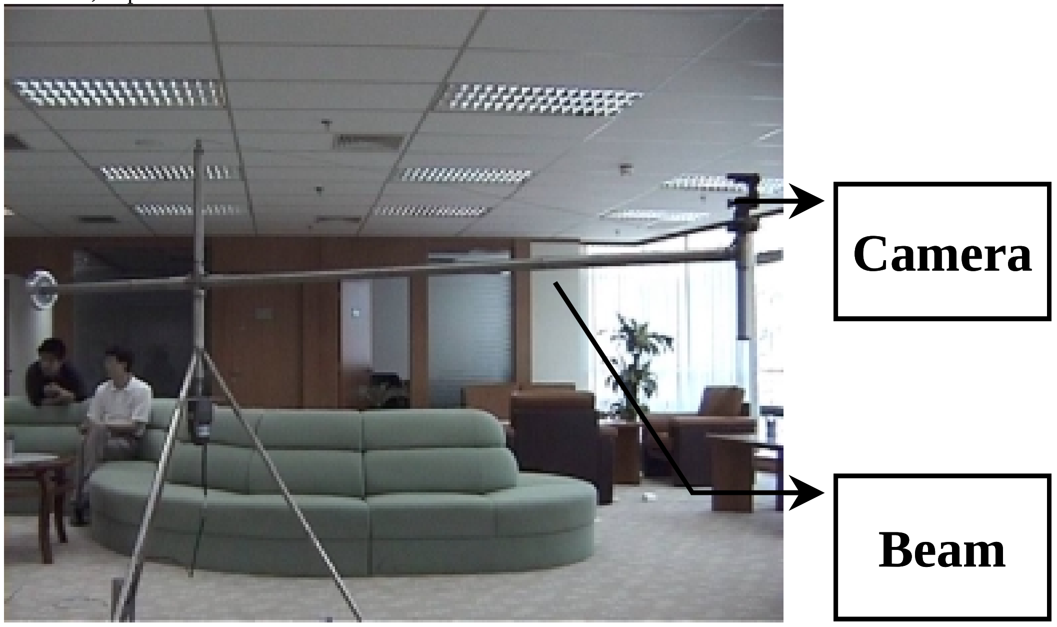


Figure 1: Capturing device of concentric mosaics.

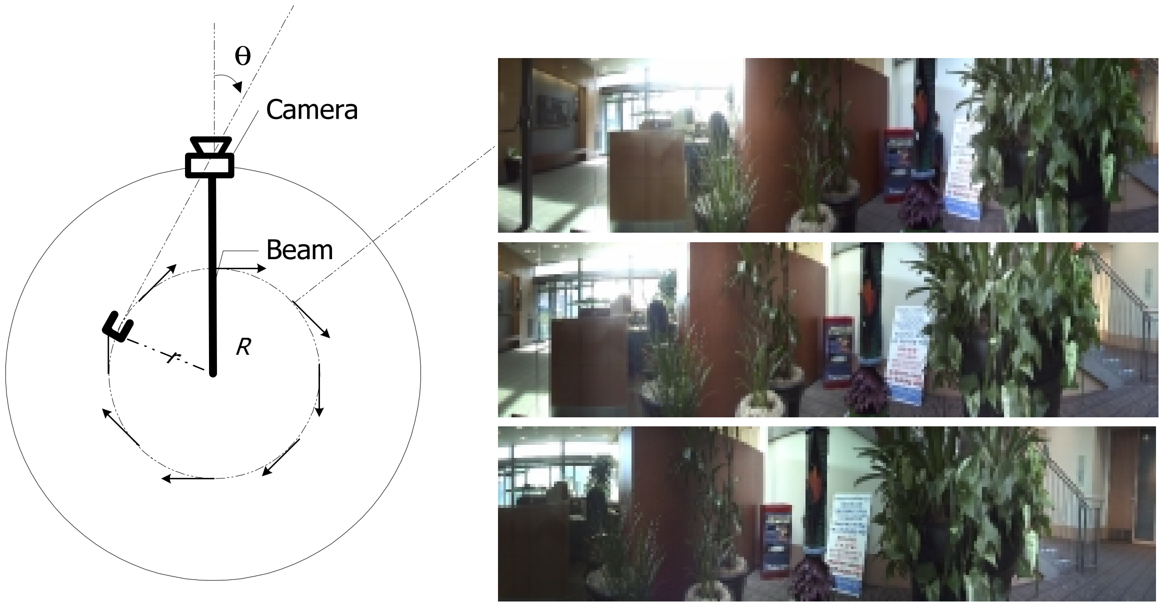


Figure 2 The concentric mosaic imaging geometry

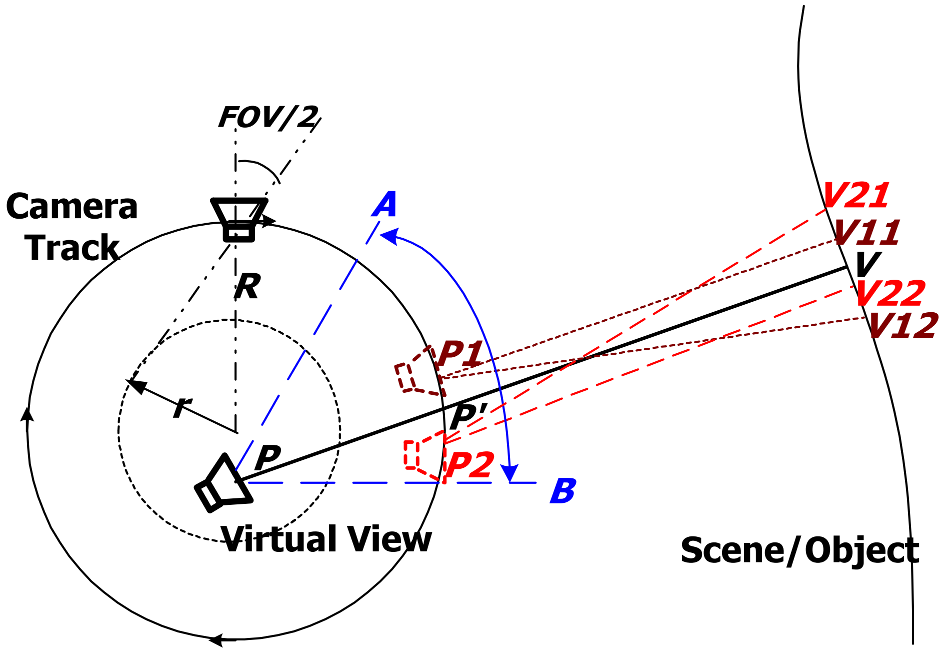


Figure 3 Rendering with concentric mosaics

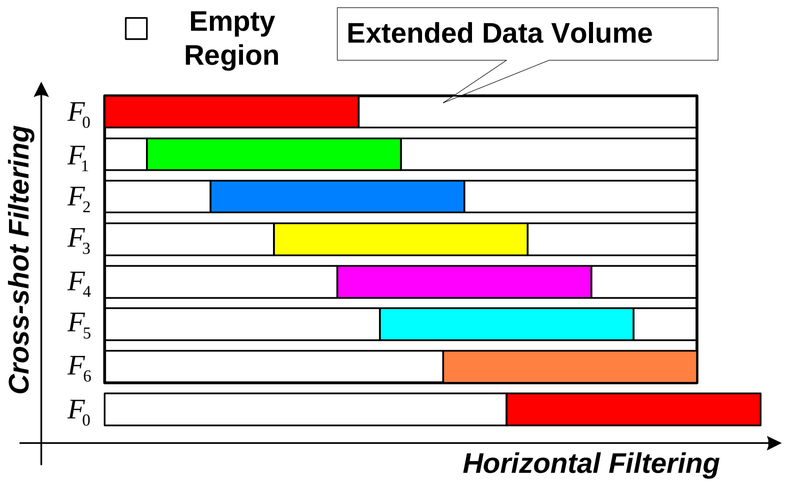
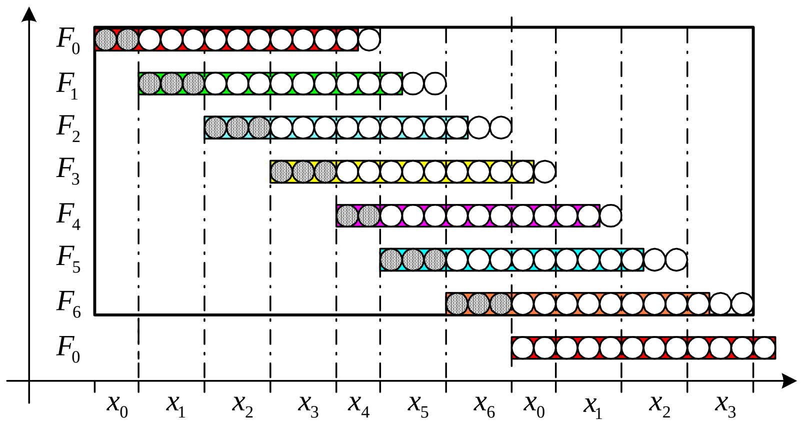
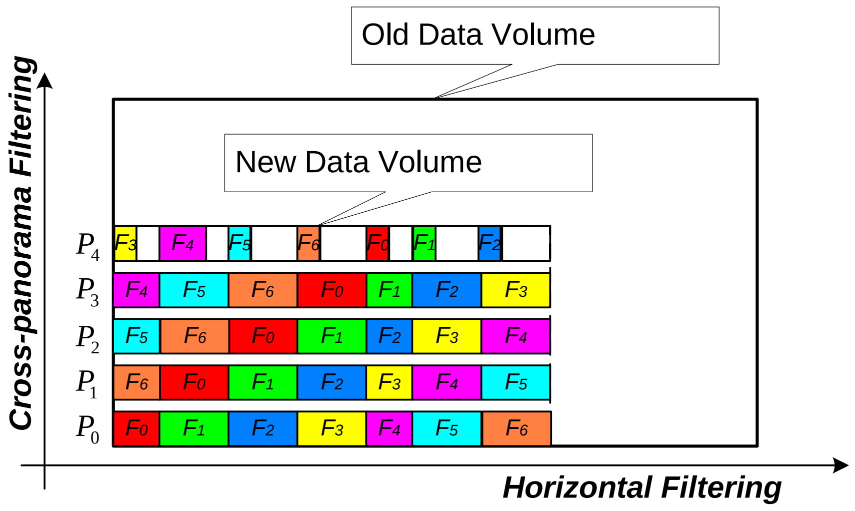


Figure 4 Horizontal shot alignment of concentric mosaic image shots.



F_0
F_1
F_2
F_3
F_4
F_5
F_6
F_0
x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_0 x_1 x_2 x_3

Figure 5 The smart-rebinning process



Old Data Volume
New Data Volume
Cross-panorama Filtering
P_4 F3 F4 F5 F6 F0 F1 F2
P_3 F4 F5 F6 F0 F1 F2 F3
P_2 F5 F6 F0 F1 F2 F3 F4
P_1 F6 F0 F1 F2 F3 F4 F5
P_0 F0 F1 F2 F3 F4 F5 F6
Horizontal Filtering

Figure 6 Smart-rebinned data volume

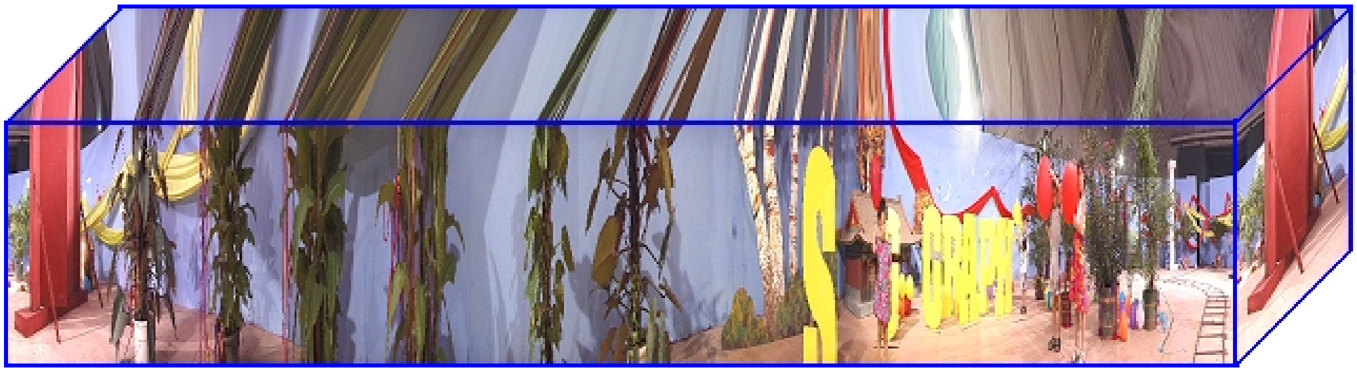


Figure 7 The volume of the concentric mosaics

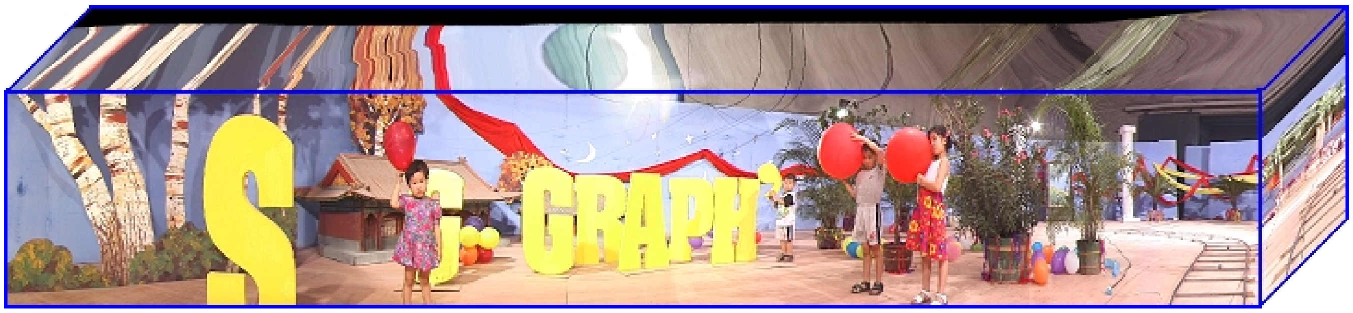


Figure 8 Part of the volume of the rebinned multi-perspective panorama set

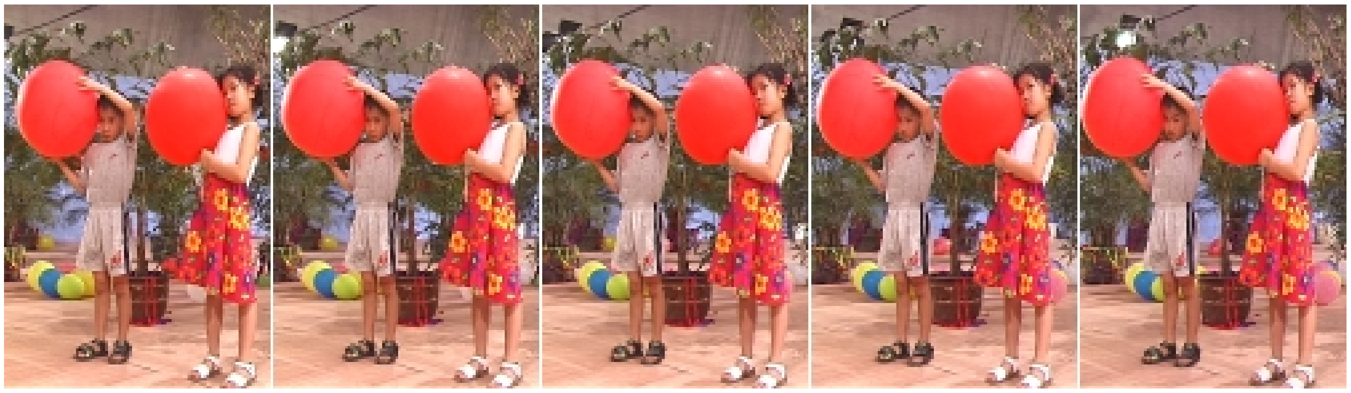


Figure 9 A set of smart-rebinned panoramas at the same horizontal location (note the parallex shown by the lightbulb and the balloon behind the girl).