Smart Rebinning for the Compression of Concentric Mosaic[1]

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

[a]Department of Electrical Engineering, Princeton University, Princeton, NJ 08544

[b]Department of Electrical & Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213

[c]Microsoft Research, China, 3F, Sigma Ctr, 49 Zhichun Road, Haidian, Beijing 100080, China.

Contact Tel. +86 10 62617711 ext. 5793 Email: jinl@microsoft.com

ABSTRACT

The concentric mosaic offers a quick solution to construct a virtual copy of a real environment, and to navigate in that virtual environment. However, the huge amounts of data associated with the concentric mosaic are heavy burdens for its application. A 3D wavelet based compressor has been proposed in the previous work to compress the concentric mosaic. In this paper, we greatly improve the compression efficiency of the 3D wavelet coder by 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 effectively enhances the correlation in the 3D data volume, translating the data into a representation that is more suitable for 3D wavelet compression. Experimental results show that the performance of the 3D wavelet coder is improved by an average of 4.3dB with the use of the smart rebinning. The proposed coder outperforms MPEG-2 coding of the concentric mosaic by an average of 3.7dB.

Keywords: Image based rendering, concentric mosaic, compression, rebinning, multi-perspective panorama, 3D wavelet.

[1] At the time when this work was done, Yunnan Wu and Cha Zhang were interns at Microsoft Research China and students at the University of Science and Technology of China, Hefei, China and Tsinghua University, Beijing, China, respectively.

1. INTRODUCTION

Image-based rendering (IBR) techniques have received much attention in the computer graphics realm for realistic scenes/objects representation. Instead of referring to the complicated geometric and photometric properties as the conventional model-based rendering does, IBR requires only sampled images to generate high quality novel views. Furthermore, the rendering speed for an IBR scene is independent of the underlying spatial complexity of the scene, which makes IBR attractive for the modeling of highly complex real environments. The concentric mosaic[1] enables 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 level beam, which is pointing outward and shooting images as the beam rotates, a concentric mosaic scene can be quickly constructed. At the time of rendering, we just split the desired 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 mosaic 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 mosaic. 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 either storage or transmission. Since the captured concentric mosaic shots are highly correlated, much higher compression ratio should be expected.

Since the data structure of the concentric mosaic can be regarded as a video sequence with slowly panning camera motion, it is natural to apply existing still image/video compression technologies to the concentric mosaic. However, the concentric mosaic bears unique characteristics, which have led to new challenges in compression. First, the concentric mosaic is a 1D image array, with highly structured camera motion among images. The cross-frame correlation is stronger than that of a typical video sequence. Second, the distortion tolerance of the concentric mosaic is smaller, because each rendered image of the concentric mosaic is viewed statically, and the human visual system (HVS) is much more sensitive to static distortions than to time-variant distortions. Since a rendered view of the concentric mosaic is formed by the combination of the image rays, certain HVS properties such as spatial and temporal masking may not be used, because neighboring pixels in the concentric mosaic dataset may not be rendered as neighboring pixels in the final view. Most important, 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 concentric mosaic bitstream should not be fully decompressed and then rendered. In fact, the decompressed concentric mosaic is so large that most hardware today has difficulties handling it. It is therefore essential to maintain the concentric mosaic in the compressed form, and decode only the contents needed to render the current view.

We can classify existing concentric mosaic compression algorithms into two categories. The first category is the reference / prediction based coder. Such coders, as the reference block coder (RBC) proposed in [14], resemble existing video coding standards such as MPEG-2 and H.263, and use motion compensated prediction. The image shots in the concentric mosaic are classified into anchor and predicted frames. The anchor frame is independently encoded, just as an I frame in MPEG-2. The predicted frame is motion compensated with regard to one of its neighboring anchor frames, and only the prediction residue is encoded. The process is similar to P frame coding in MPEG-2, though the predicted frame in RBC only refers to the anchor frame so that the compressed RBC bitstream can be flexibly accessed. A two-level hierarchical index table is also embedded in the RBC bitstream for random data access. Magner *et. al.* [16] proposed a model-aided coder for the compression of the Lightfield, which is a 2D image array. Five images, one at the polar and four at the equator, are intra-coded. The other images are predictively encoded with reference to the intra-coded frame.

The second category is the high dimensional wavelet coders, which explore the cross frame redundancy via wavelet filtering. Luo *et. al.* [7] proposed a 3D wavelet coder for the compression of the concentric mosaic. The mosaic images are aligned and wavelet filtered both within the mosaic image and across the mosaic images. After that, the wavelet coefficients are split into fixed size blocks, which are then quantized, entropy encoded and assembled into the compressed bitstream. Magner *et. al.* also propose a 4D Haar wavelet coder with SPIHT coefficient coding in [17] to compress the Lightfield. One attractive property of the high dimensional wavelet coder is its spatial, temporal and quality scalability. Here the term scalability means that a high dimensional wavelet coder can compress the scene into a single bitstream, where multiple subsets of the bitstream can be decoded to generate complete scenes of different spatial resolution/temporal resolution/quality commensurate with the proportion of the bitstream decoded. This is extremely useful in the Internet streaming environment where heterogeneous decoder/network settings prevail. Furthermore, since wavelet coders avoid the recursive loop in the predictive coders, they perform better in an error prone environment, such as a wireless network.

One common challenge with the high dimensional wavelet compression schemes is that the cross image wavelet filtering does not achieve efficient energy compaction. In other words, the coherence in the cross-image direction is not strong, resulting in many large high frequency coefficients in that direction. As a result, the 3D wavelet coder of the concentric mosaic [7] and the 4D wavelet coder of the Lightfield [17] lag behind in compression performance compared to the predictive concentric mosaic coder [14] and the predictive Lightfield coder [16]. The same is the case if the compression performance of the 3D wavelet video coder [2][3][4][5] is compared with that of a state-of-the art video compression standard such as MPEG-4 or H.263++. In a prediction-based video / concentric mosaic coder, local motion can be specified on a per block basis, thus interframe correlation due to the moving object / camera can be efficiently explored which is very beneficial to the coding performance. However, local motion cannot be easily incorporated into the framework of 3D wavelet compression. Due to the nature of temporal filtering, each pixel has to engage in one and only one transform. Therefore, a pixel without matching correspondence (e.g., a pixel which is newly covered by a foreground object) still has to be filtered with certain other pixels. And tricks such as half pixel motion compensation cannot be used in a 3D wavelet coder.

There was work to improve the cross frame correlation (coherence) before the wavelet filtering. In the 3D wavelet concentric mosaic codec[7], a panorama alignment module was used to rotationally shift the mosaic panorama. This module is similar to the pan motion compensation incorporated in [2]. The aligned concentric mosaic is then transformed and encoded. The 4D wavelet coder of the Lightfield [17] morphed individual images onto a common texture plane, which were then wavelet filtered. For video compression, 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 low motion sequences without filling the holes caused by individual block motion.

In this paper, a smart rebinning operation is proposed as a novel preprocessing technique for the 3D wavelet compression of the concentric mosaic. Rather than adapting the compression algorithm or the filter structure to the mosaic image array, we modify the data structure for easy compression by the 3D wavelet. Conceptually speaking, the proposed scheme improves the inter-frame coherence by explicitly clustering similar contents together. 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 maintain the high correlations inside the image shots; meanwhile greatly improve the correlation across the shots. Therefore they 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 mosaic is provided in Section 2. The smart-rebinning operation and its rationale and potential benefits for the 3D wavelet codec are detailed in Section 3. The 3D wavelet coding and rendering of the rebinned concentric mosaic volume, which is no longer of rectangular region of support, is discussed in Section 4. Experimental results are presented in Section 5. Finally, we conclude the paper in Section 6.

2. BACKGROUND: THE CONCENTRIC MOSAIC

A concentric mosaic scene is captured by mounting a camera at the end of a level 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 $F_n=\{f(n,w,h)|w,h\}$, 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 F_n, or alternatively be interpreted as a series of rebinned mosaics $P_w=\{f(n,w,h)|n,h\}$ where each individual mosaic consists of vertical slits at position w of all camera shots. Three rebinned mosaics at different radii are shown in Figure 2. Mosaic 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 $d=Rsinq$, where R is the radius of the rotation beam, d is the equivalent radius of the slit camera, and q is the angle between the tangent of the mosaic P_w and the camera norm. Since the entire data volume $\boldsymbol{P_w,\ w=0,\cdots,W-1}$ can be considered as a stack of co-centered mosaics with different radii, it is called the concentric mosaic[1].

The concentric mosaic is able to capture a realistic environment and render arbitrary views within an inner circle of radius $r=Rsin(FOV/2)$, where FOV is the horizontal field of view of the capturing camera. Rendering the concentric mosaic 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 equivalently in the rebinned mosaic 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 the shot P_1, and P_2V_{21} and P_2V_{22} are closest to P_2V in the 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*. Such a scheme is called point sampling interpolation. 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 search for 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

The framework of the smartly rebinned 3D wavelet concentric mosaic coder can be shown in Figure 4. Our proposed smart rebinning technique serves as a preprocessing stage which rearranges the original data volume. The rearranged data volume is then 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.

This framework is similar to the 3D wavelet concentric mosaic compression scheme we have proposed in [7]. A key contribution of this paper is an efficient data reorganization strategy that greatly improves the efficiency of cross-shot filtering (equivalent to the temporal direction in video, as there is no time domain in the concentric mosaic), while maintaining about the same filtering efficiency along the other directions.

In recognition of the significant role of motion compensation in the 3D wavelet compression, we look for an efficient de-correlation scheme along the cross-shot direction. In the previous work [7], the concentric mosaic is aligned through global panning of the mosaic $\boldsymbol{P_w}$. However, this approach doesn't improve the compression efficiency much because the major discontinuity of the dataset happens inside each mosaic. Thus the compression performance of the 3D wavelet codec suffers. In this work, we begin by aligning the image shots and forming a new data volume of non-rectangular region of support. Since the concentric mosaic assumes 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 none 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 $\boldsymbol{F_0}$ is right next to shot $\boldsymbol{F_{N-1}}$. 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 5. 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 black region on the horizontal line in Figure 5 corresponds to one captured image. 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. The gray area contains no data, thus the aligned image shot forms a dataset of non-rectangular region of support.

After the horizontal shot alignment, the correlation across image shots is expected to improve, however, since the resultant data volume is highly sparse and not rectangular, the compression efficiency may be compromised. Our proposed **smart rebinning** goes beyond the 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 5, and form smartly rebinned panoramas. The details of the smart rebinning operation are shown in Figure 5. The 7 image shots are rebinned into 5 panoramas. The aligned frame boundaries are shown with dashed lines. We divide the original shots into groups of aligned slits according to the horizontal displacement vectors, which are called stripes. 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 (fast moving), and more stripes for that with the faraway background. We then downward stack the stripes and form the rebinned panorama set. We 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, or by stripes $s_{0,0}, s_{1,0}, \cdots, s_{6,0}$. The second panorama $\boldsymbol{P_1}$ 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. In general, a smartly rebinned panorama $\boldsymbol{P_i}$ consists of the ith stripes of all frames cut and pasted 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. 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. In this specific example, every stripe is at least partially filled. However, for an actual concentric mosaic scene with a larger variation of scene depth, several end stripes may be completely empty. The smartly rebinned panoramas are thus not of rectangular region of support. Special handling for coding those empty regions will be addressed in the next section. Note that in order to reverse such a rearrangement, only the horizontal displacement between each pair of neighboring shots needs to be recorded, adding minor overhead to the compressed bitstream.

Examining the smartly rebinned panorama, the unfilled regions of the skewed dataset are largely reduced, which makes the compression much more efficient and the implementation much more convenient. 3D wavelet filtering of the rebinned panorama is still very efficient. Filtering across the panorama is exactly equivalent to filtering across the image shots in the horizontal shot alignment approach shown in Figure 5. However, filtering within the panorama 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 very efficient.

A degenerated approach of the smart rebinning 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 mosaic $\boldsymbol{P}_w=\{f(n,w,h)|n,h\}$ 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 mosaic in Figure 7. The rebinned concentric mosaic forms 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 a 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 mosaic, the smartly rebinned panorama appears to be smoother and more 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 graphics 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 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 [14] can also be applied, 3D wavelet coding is ideal because good alignment across multiple image shots can be more efficiently explored by the 3D wavelet coder. For the simple rebinning, straightforward 3D wavelet encoding may be adopted. We use the 3D wavelet codec with block arithmetic coding as proposed in our previous paper [7]. The data volume of the concentric mosaic is decomposed by multi-resolution 3D wavelet transform. The wavelet coefficients are then cut into fixed size blocks, embeddedly encoded, and assembled with a rate-distortion optimization criterion. For details of the 3D wavelet coding algorithm, we refer the reader to [7].

For the 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]. There are two categories of approaches: padding the data to a rectangular volume and then using a rectangular codec, or using an arbitrary shape wavelet transform and coder. Both approaches are investigated.

A simple approach is to pad the arbitrary shaped 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. The purpose of padding is to generate as many zero coefficients as possible in the unsupported region, because zero coefficients consume fewer bits in the subsequent entropy coder. In this work, we extend the low-pass extrapolation (LPE) adopted in MPEG-4 for the padding. 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 to 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 arbitrary shaped 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 shaped 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] and [13]. A block arithmetic coder with an arbitrary shaped region of support in the wavelet domain is then used to code 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 slows down the decompressing speed. 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 modules.

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 [15]. Instead of decompressing the entire compressed concentric mosaic and then rendering it, the entire concentric mosaic can be kept in the compressed form, and only the part of the bitstream necessary to render the current view is decoded and rendered. This not only reduces the memory requirement of the render, but also avoids a long decoding delay at the beginning. The working flow of the selective rendering engine can be shown in Figure 10. 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. Note that the position of a slit in the rebinned panorama is only related to the horizontal translation vectors between image shots, and can be easily calculated. After that, the PIWS engine is used to locate the positions of the slits in the wavelet domain. Only the wavelet coefficient blocks containing accessed slits are decoded from the compressed bitstream. With a mixed cache transition among the wavelet coefficients, the intermediate results, and the reconstructed pixels, PIWS ensures minimal computation is performed 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 mosaic is to locate the slits in the rebinned panorama. The computational complexity of rendering the smartly rebinned concentric mosaic is thus similar to the rendering of 3D wavelet compressed concentric mosaic. With the PIWS engine, a rendering rate of 12 frames per second is achievable, which is fast enough for the 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 used 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 smartly rebinned 3D wavelet coder with three benchmark algorithms. The first algorithm (A) compresses the entire concentric mosaic 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 the direct 3D wavelet codec 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 block arithmetic coding. The third benchmark algorithm (C) is the reference block coder (RBC) reported in [14]. It is a prediction-based codec tuned for the compression of concentric mosaic. 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 test 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 mosaic forms a rectangular panorama volume and is compressed by exactly the same 3D wavelet and block arithmetic coder as algorithm B. It is observed that by simply rebinning multiple slits into the panorama, a large compression gain can be achieved. In fact, compared with the direct 3D wavelet codec, the PSNR improves between 3.2 and 3.6dB, with an average gain 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 which has a larger depth variation, 36% of the bounding volume is unsupported. The smartly rebinned panoramas are compressed through two approaches. In the first approach, we compress the rebinned panoramas through padding the data volume and applying a rectangular 3D wavelet codec used in the algorithm B and D (denoted as algorithm E). Alternatively, we use an arbitrary shape wavelet transform and coefficient coding algorithm developed in [13](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 codec (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 the rebinning, rectangular 3D wavelet transform, and block arithmetic 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 (RBC) by 3.2dB. The PSNR of the smart rebinning compressed *Lobby* at 0.12bpp is even superior to prior concentric mosaic coders operated at 0.2 bpp. More specifically, 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. In the same way, we observe that the smart rebinning nearly doubles the compression ratio for the *Kids* scene over prior approaches. Considering the huge amount of data of the concentric mosaic, and 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 the concentric mosaic.

6. CONCLUSION AND EXTENSION

A technology termed smart rebinning is proposed in this paper to improve the 3D wavelet compression of the concentric mosaic. 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 coder. 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 scenes. It nearly quadruples the compression ratio for the *Lobby* scene, and doubles the compression ratio for the *Kids* scene.

It will be interesting to extend the smart rebinning technology to general 3D wavelet coding of video sequence, and see if similar performance gain can be achieved. However, such extension is far from straightforward. The motion is more complicated in a typical video sequence. Moreover, video needs to be decoded frame by frame, which does not match well for a data rearrangement scheme such as rebinning. We are investigating along the direction.

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 the concentric mosaic renderer; Jizheng Xu *et. al.* for providing the arbitrary shape wavelet transform and coder in [13].

8. REFERENCES

[1] H.-Y. Shum and L.-W. He. "Rendering with concentric mosaic", *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] J. Xu, S. Li and Y. Zhang, "Three-dimensional shape-adaptive discrete wavelet transforms for efficient object-based video coding", *IEEE/SPIE Visual Communications and Image Processing (VCIP) 2000*, Perth, June 2000.

[14] 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.

[15] 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.

[16] M. Magnor, P. Eisert and B. Girod, "Model-aided coding of multi-viewpoint image data", *Proc. International Conference on Image Processing (ICIP-2000)*, Vancouver, Canada, Sept. 2000.

[17] M. Magnor and B. Girod, "Model-based coding of multi-viewpoint imagery", *Proc. Visual Communications and Image Processing 2000(VCIP'2000)*, Perth, Australia, June 2000.

Indexes for Tables and Figures

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

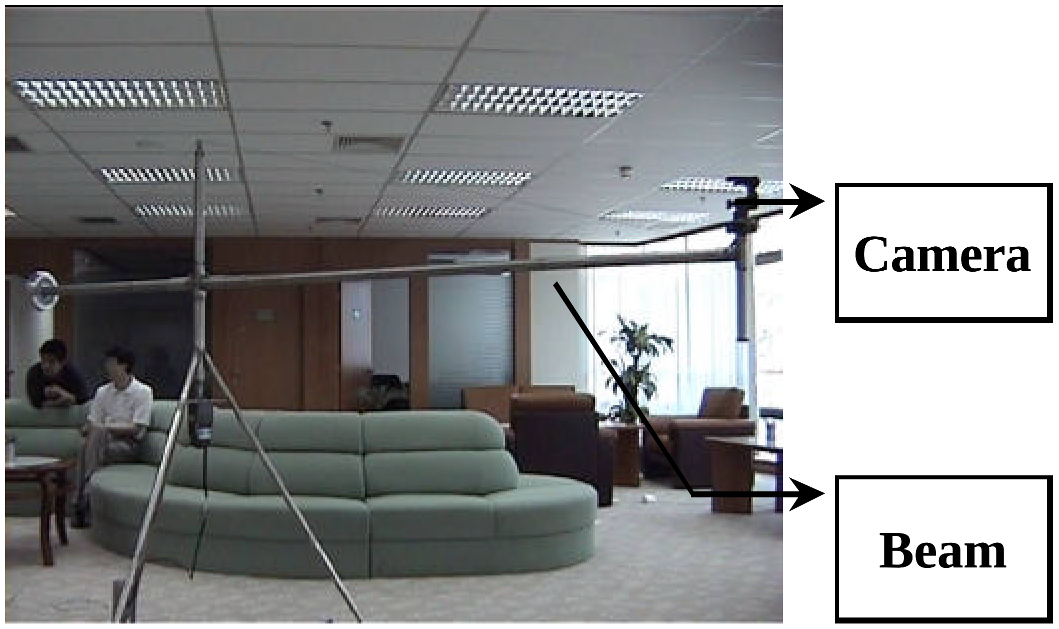

Figure 1: Capturing device of concentric mosaic.

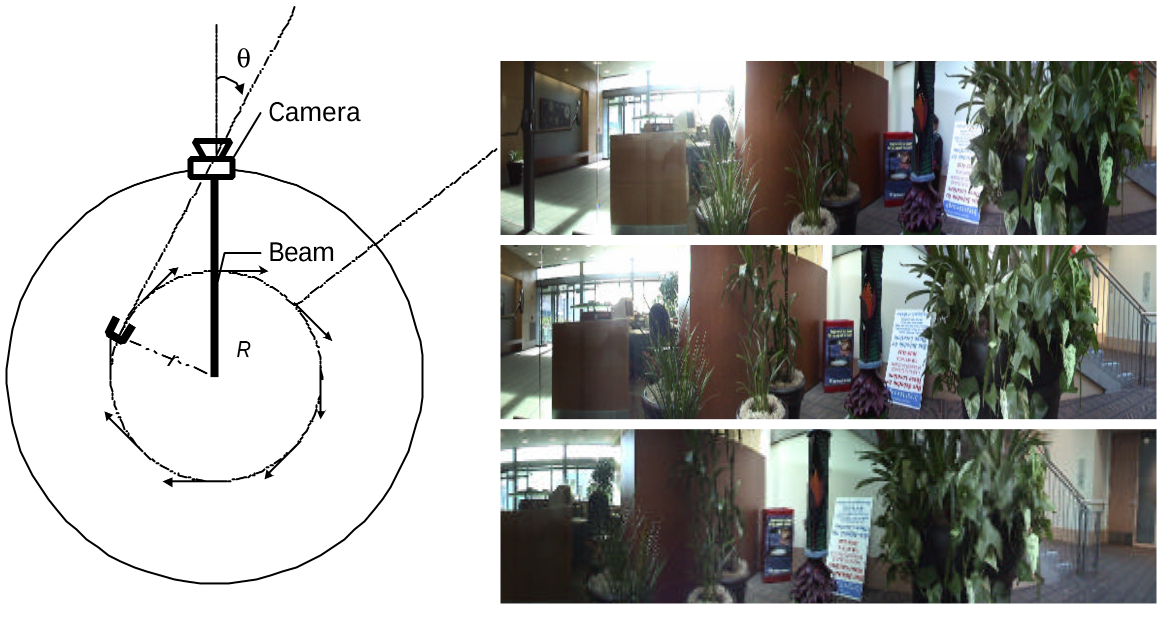

Figure 2 The concentric mosaic imaging geometry

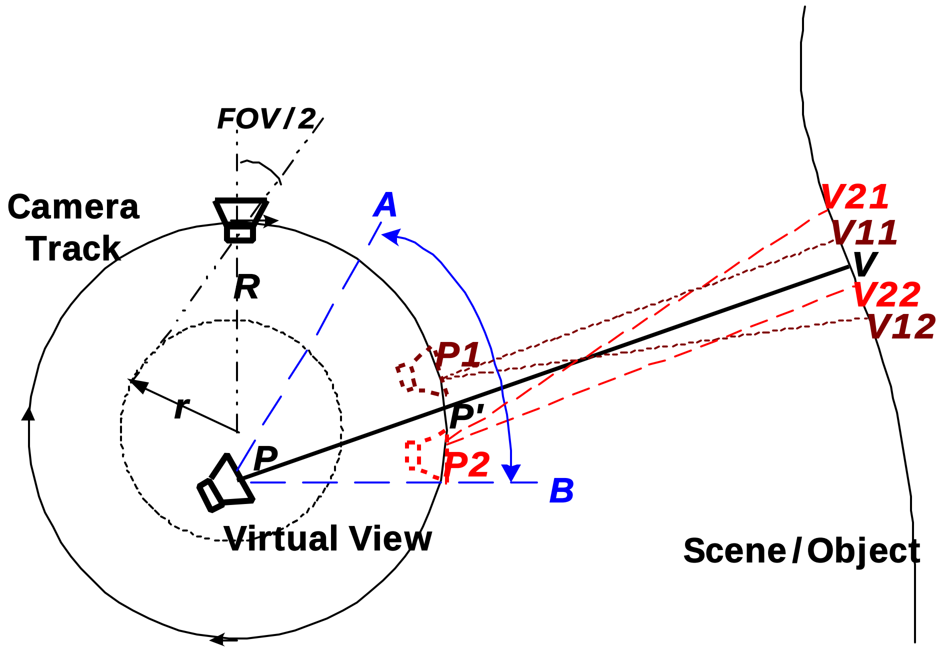

Figure 3 Rendering with concentric mosaic

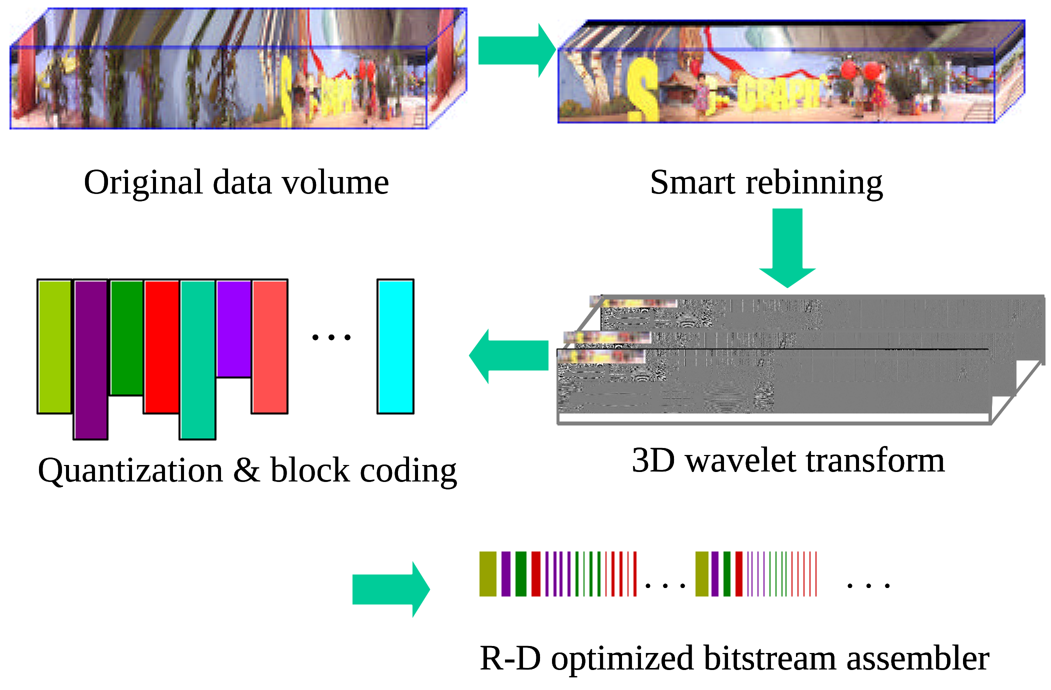
Original data volume
Smart rebinning
3D wavelet transform
Quantization & block coding
R-D optimized bitstream assembler

Figure 4 Framework of smartly rebinned 3D wavelet coder.

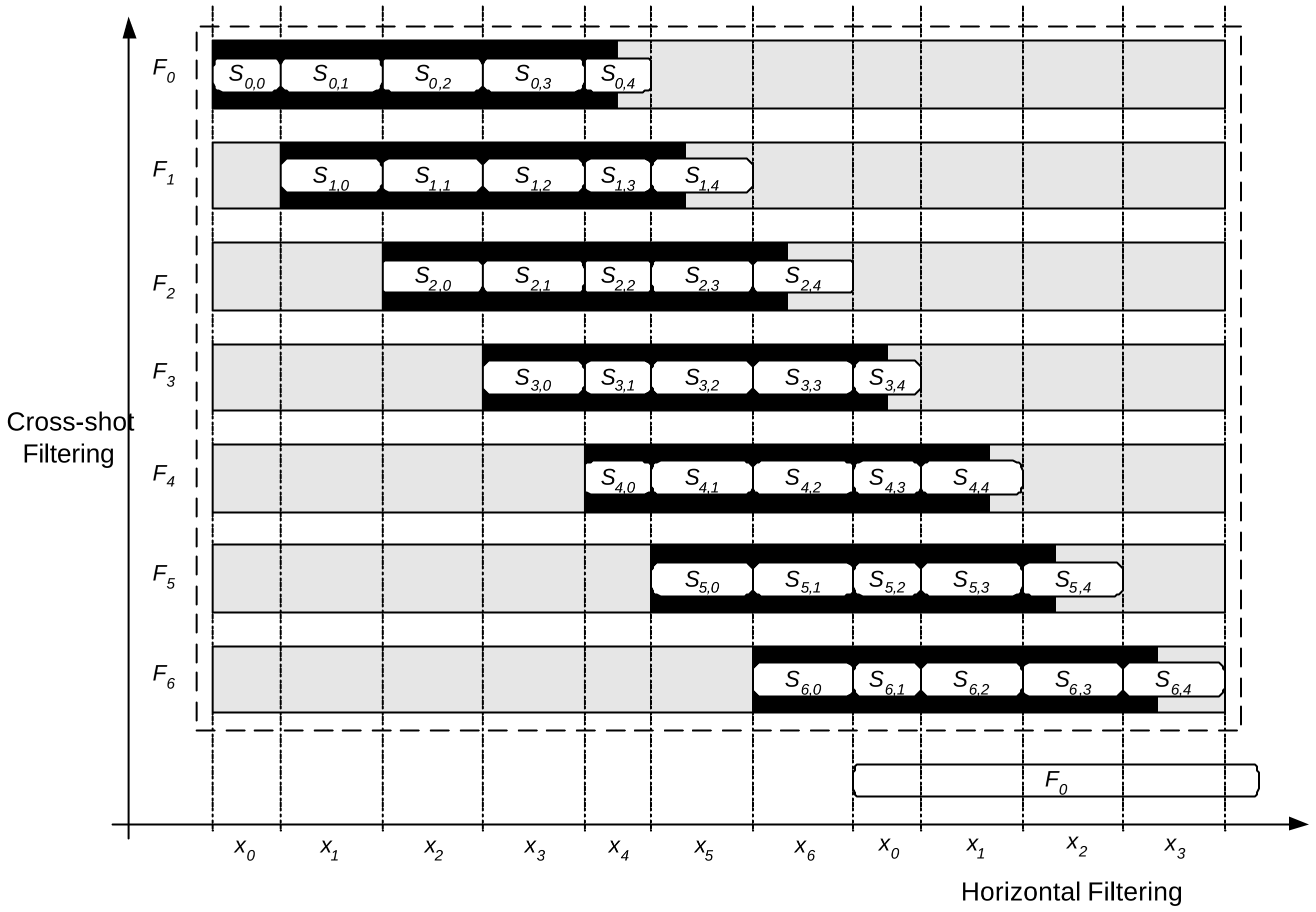

Figure 5 Horizontal shot alignment of concentric mosaic image shots.

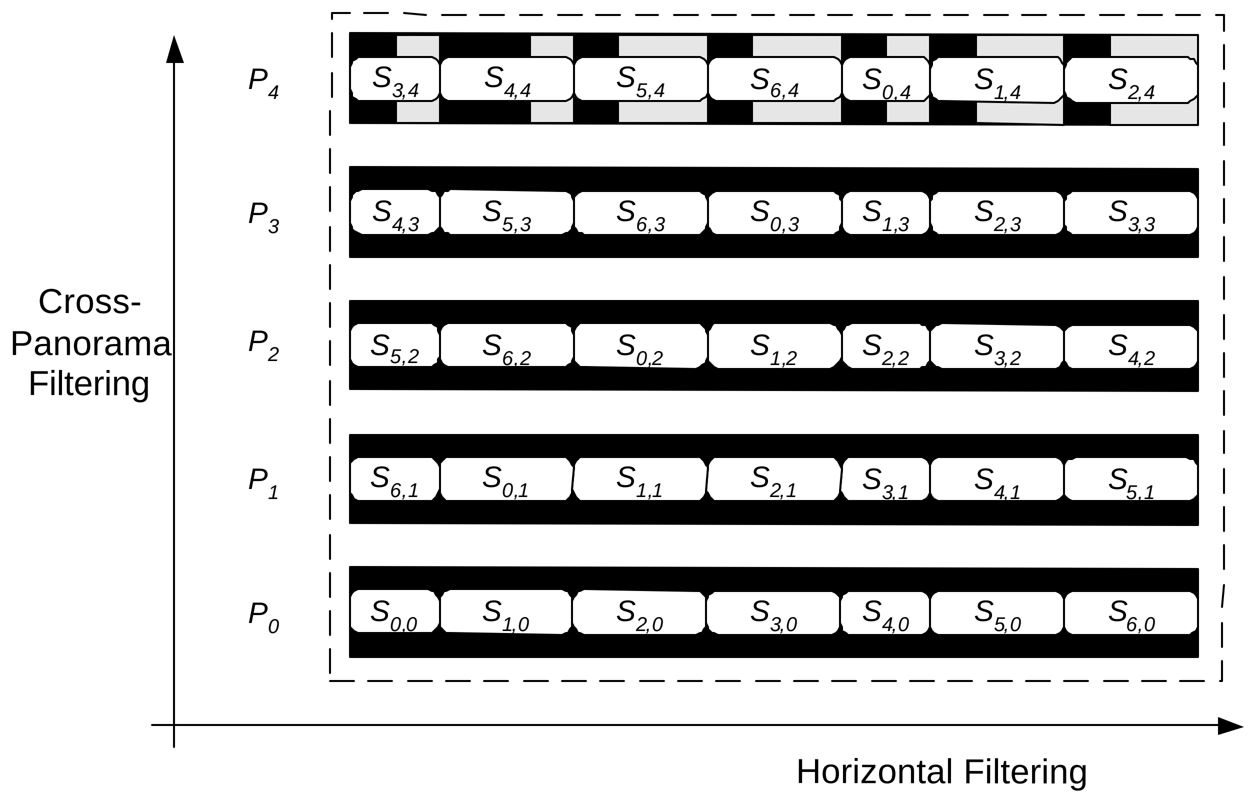
P_4
$S_{3,4}$ $S_{4,4}$ $S_{5,4}$ $S_{6,4}$ $S_{0,4}$ $S_{1,4}$ $S_{2,4}$
P_3
$S_{4,3}$ $S_{5,3}$ $S_{6,3}$ $S_{0,3}$ $S_{1,3}$ $S_{2,3}$ $S_{3,3}$
P_2
$S_{5,2}$ $S_{6,2}$ $S_{0,2}$ $S_{1,2}$ $S_{2,2}$ $S_{3,2}$ $S_{4,2}$
P_1
$S_{6,1}$ $S_{0,1}$ $S_{1,1}$ $S_{2,1}$ $S_{3,1}$ $S_{4,1}$ $S_{5,1}$
P_0
$S_{0,0}$ $S_{1,0}$ $S_{2,0}$ $S_{3,0}$ $S_{4,0}$ $S_{5,0}$ $S_{6,0}$
Cross-Panorama Filtering
Horizontal Filtering

Figure 6 Smartly rebinned data volume

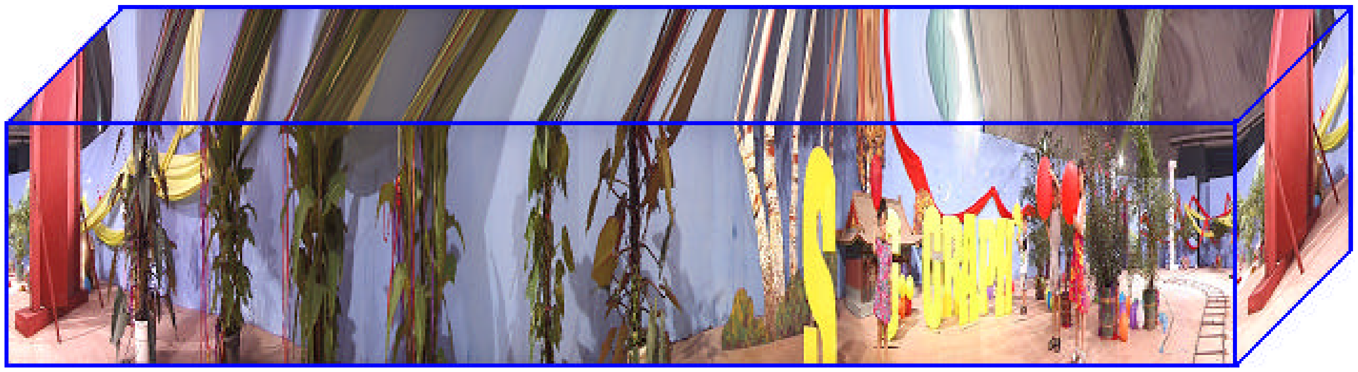

Figure 7 The volume of the concentric mosaic

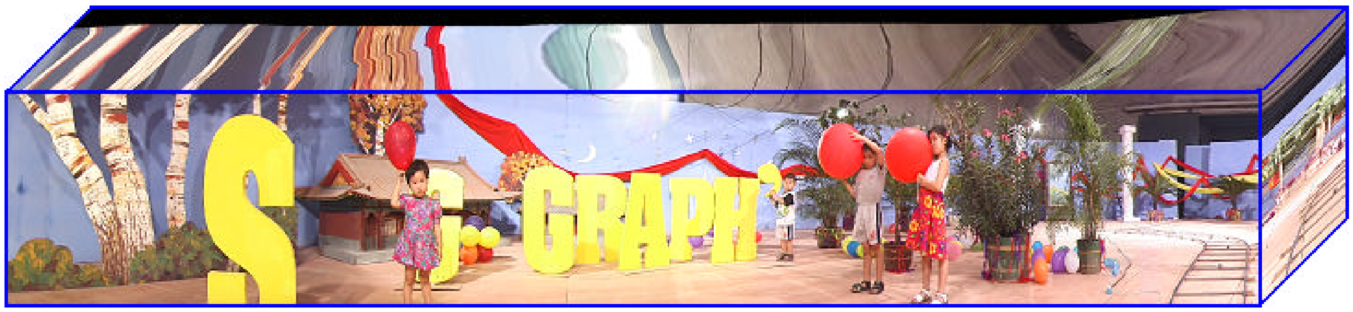

Figure 8 Part of the volume of the rebinned multi-perspective panorama set

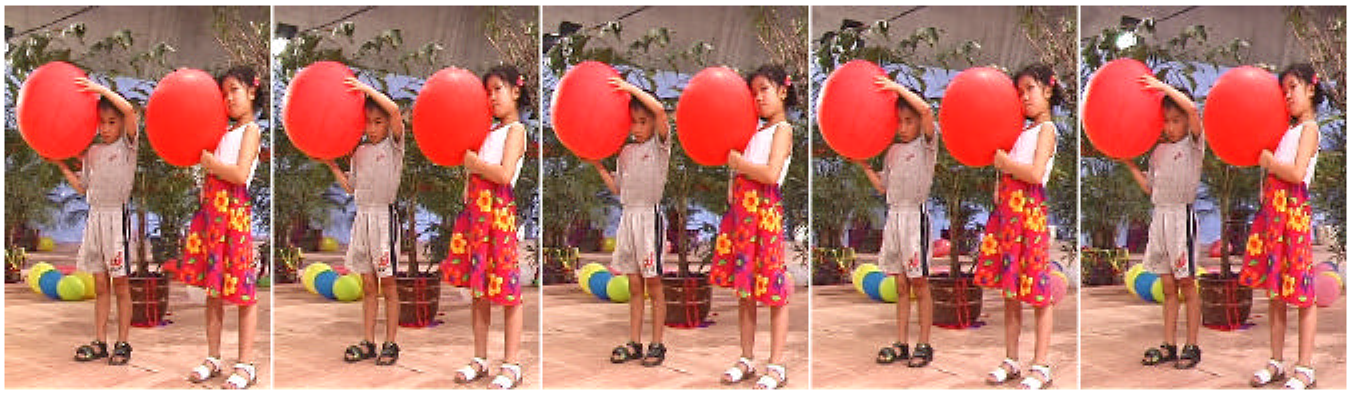

Figure 9 A set of smartly rebinned panoramas at the same horizontal location (note the parallax shown by the lightbulb and the balloon behind the leg of the girl).

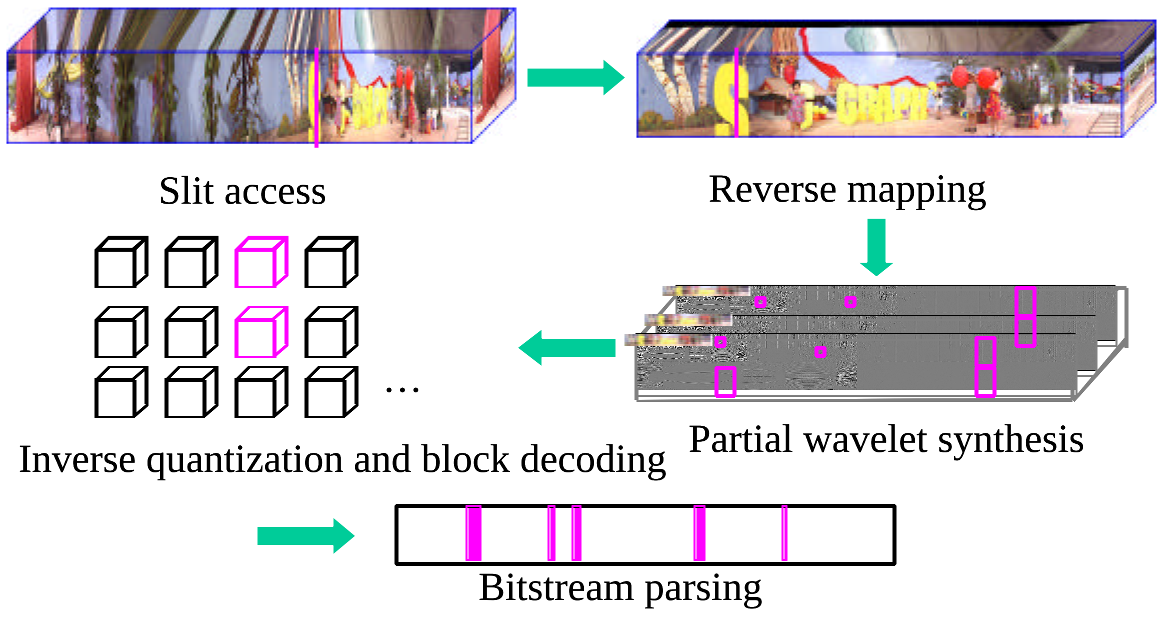
Slit access
Reverse mapping
Partial wavelet synthesis
Inverse quantization and block decoding
Bitstream parsing

Figure 10 Framework of the selective rendering engine.