Hindawi Publishing Corporation
EURASIP Journal on Image and Video Processing
Volume 2009, Article ID 251081, 12 pages
doi:10.1155/2009/251081

Research Article

Rendering-Oriented Decoding for a Distributed Multiview Coding System Using a Coset Code

Yuichi Taguchi and Takeshi Naemura

Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1, Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

Correspondence should be addressed to Yuichi Taguchi, yuichi@hc.ic.i.u-tokyo.ac.jp

Received 1 May 2008; Revised 10 November 2008; Accepted 3 February 2009

Recommended by Stefano Tubaro

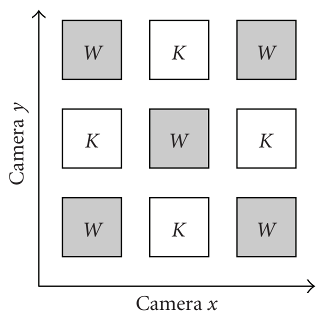
This paper discusses a system in which multiview images are captured and encoded in a distributed fashion and a viewer synthesizes a novel image from this data. We present an efficient method for such a system that combines decoding and rendering processes in order to directly synthesize the novel image without having to reconstruct all the input images. Our method jointly performs disparity compensation in the decoding process and geometry estimation in the rendering process, because they are essentially equivalent if the camera parameters for the input images are known. Our method keeps both encoder and decoder complexity as low as that of a conventional intracoding method, while attaining better coding performance owing to the interimage decoding. We validate our method by evaluating the coding performance and the processing time for decoding and rendering in experiments.

1. Introduction

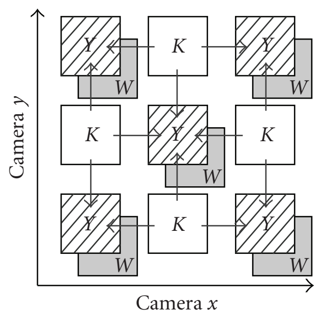
Camera array systems can capture multiview images of a 3D scene, which allow a viewer to observe the scene from arbitrary viewpoints by using image-based rendering techniques [1, 2]. Such systems require efficient coding schemes owing to the large amount of data, typically consisting of hundreds of views. Since they capture an identical scene from slightly different viewpoints, significant correlations exist among the multiview images. Most of conventional coding methods, as well as currently developed MPEG standard, exploit these correlations at the encoder using the concept of disparity compensation [2]. However, they require high encoding complexity and communication between cameras with large data volume.

Distributed multiview coding methods provide a solution for such problems [3–6]. In these methods, each image is encoded independently, but decoded jointly at a central decoder. Since the intercamera communication is avoided, low complexity encoding and a simple system configuration can be achieved. The interimage correlation is exploited at the decoder. Therefore, compression efficiency is still higher than that possible by conventional intracoding methods. In previous works, however, the decoder seems to pay an unnecessary computational cost when the viewer only observes a novel image synthesized at a desired viewpoint, instead of the decoded images themselves. This is because it first reconstructs input camera images and then synthesizes the novel image with a general renderer using the decoded images. To our knowledge, there is no approach so far that synthesizes a novel image directly from the encoded data.

In this paper, we consider a system in which multiview images are captured and encoded in a distributed fashion and a viewer synthesizes a novel image at a desired viewpoint by using this data. We propose an efficient method that combines decoding and rendering processes so that the novel image can be directly synthesized without having to reconstruct all the input images. This method, called rendering-oriented decoding, jointly performs two key techniques, disparity compensation in the decoding process and geometry estimation in the rendering process, because they are essentially equivalent if the camera parameters for the multiview images are known. When the viewer only synthesizes a novel image, our method requires lower computational cost than a typical method that performs the above two processes separately. Our method keeps the complexity of both the encoder and decoder as low as a conventional intracoding method, while attaining better coding performance thanks to the interimage decoding.



(a) Encoder



(b) Decoder

Figure 1: A typical structure of distributed multiview coding systems.

The rest of this paper is organized as follows. Section 2 briefly describes two basic schemes for this study: distributed multiview coding techniques and an image-based rendering algorithm. Section 3 presents our rendering-oriented decoding method. Section 4 evaluates the coding efficiency and processing time of our method compared to a conventional intracoding method, and Section 5 concludes the paper.

2. Background

2.1. Distributed Multiview Coding. Figure 1 shows a typical structure of distributed multiview coding systems. The images are classified into two categories: key images (K) and Wyner-Ziv images (W). The key images are encoded and decoded independently with a conventional intraimage coder. The Wyner-Ziv images are encoded independently by applying a channel coder for their pixel values or transformed coefficients, and the resulting parity bits are transmitted to the decoder. To decode the Wyner-Ziv image, its estimate, called side information (Y), is generated through disparity-compensated prediction using the previously decoded key images, and the prediction error is corrected by using the parity bits of the image.

The compression efficiency of the distributed coding methods greatly depends on the accuracy of the side information, because only a few parity bits are needed to correct small prediction errors. If a geometry model of the target scene is available, accurate side information can be generated by warping the neighboring views [4]. For multiview video sequences, to improve the quality of side information, the motion-compensated prediction can be combined with the disparity-compensated one [5, 6].

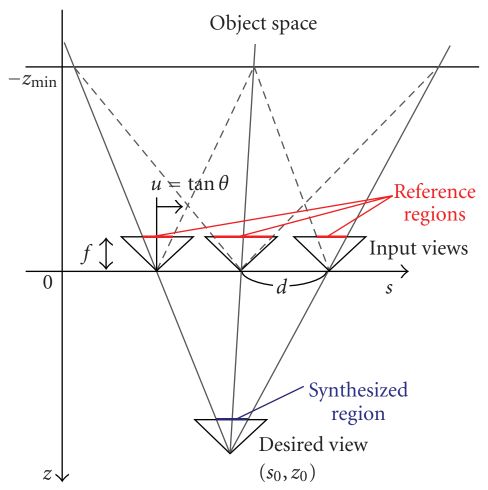


Figure 2: Light field parameterization and the reference regions used for interpolating the synthesized region.

2.2. Rendering Using Multiview Images. We assume that multiview images are captured with calibrated cameras that roughly lie on a plane and are arranged on a 2D grid (e.g., [7–13]), and that there is no prior knowledge of the scene geometry. The light rays included in the multiview images can be parameterized as a light field [14, 15] (s, t, u, v), where (s, t) and (u, v) denote the positions and directions of the light rays, respectively. Figure 2 shows a subspace (s, u) of a light field constructed with input cameras arranged on a regular grid with the same pose, for simplicity. For synthesizing a novel image at a desired viewpoint (s_0, z_0), light rays that pass through the viewpoint need to be gathered. They must satisfy

$$u = \frac{f}{z_0}(s - s_0), \tag{1}$$

where f is the focal length of the input cameras. Since a light field is usually composed of a finite number of input cameras, geometry (depth) estimation is widely adopted to appropriately interpolate the light rays that are not actually captured with the cameras. Here, we first describe a rendering method that estimates a per-pixel depth map depending on the desired viewpoint [13, 16], and then explain the locality of light rays used in the rendering method.

2.2.1. Rendering Method. As shown in Figure 3, a layered depth model, $z = \{z_n \mid n = 1, 2, \dots, N\}$, is assumed in the object space to equally divide the disparity space as

$$\frac{1}{z_n} = \frac{1}{z_{\max}} + \frac{n - 1/2}{N}\left(\frac{1}{z_{\min}} - \frac{1}{z_{\max}}\right), \tag{2}$$

where $z_{\max}$ and $z_{\min}$ are the maximum and minimum depths of the scene. We estimate the depth for each target light

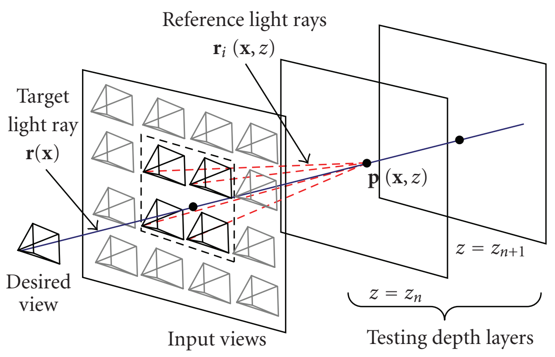


FIGURE 3: Configuration for rendering a desired view.

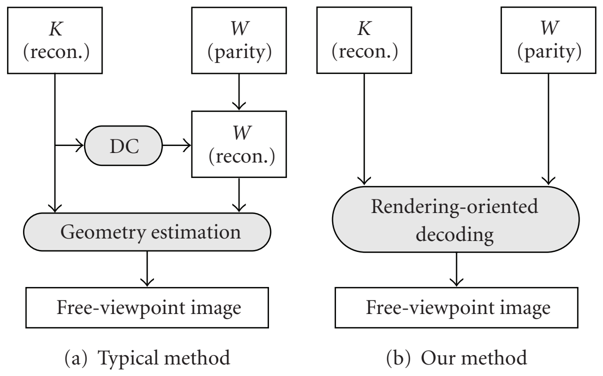


FIGURE 4: Process flow for synthesizing a free-viewpoint image (DC: disparity compensation).

ray, $\mathbf{r}(\mathbf{x})$, where $\mathbf{x}$ represents the position of the light ray in the desired view. At the intersection of the target light ray with each of the depth layers ($\mathbf{p}(\mathbf{x}, z)$), we evaluate the color consistency of the reference light rays, which correspond to the back-projections of the intersection point to the input cameras. The light rays are denoted by $\mathbf{r}_i(\mathbf{x}, z)$ where i is the camera index. To prevent the occlusion effect and keep computational cost low, this evaluation is only performed on the k-nearest cameras (reference cameras). The color consistency cost is therefore given by

$$C(\mathbf{x}, z) = \text{consistency}\left(I(\mathbf{r}_i(\mathbf{x}, z))\,|_{i \in V}\right), \tag{3}$$

where V is the set of camera indices near the target light ray and $I(\cdot)$ denotes the color of the light ray. In our implementation, we used the sum of variances for each RGB component as the consistency measure, and set $|V| = k = 4$ as shown in Figure 3.

This cost function is smoothed in each depth layer in order to reduce noise effects. For this smoothing, we use a normal block filter

$$\overline{C}(\mathbf{x}, z) = \frac{1}{|S|} \sum_{\mathbf{x}' \in S} C(\mathbf{x}', z), \tag{4}$$

where S is a rectangular window whose center is $\mathbf{x}$. Finally, the depth value that minimizes the cost is selected for each target light ray:

$$z_{\text{opt}}(\mathbf{x}) = \arg\min_{z} \overline{C}(\mathbf{x}, z). \tag{5}$$

As in the depth estimation, we use k-nearest reference light rays to interpolate the color of the target light ray. This approach keeps the view-dependent components of the target scene and prevents an unnecessarily blurred result [17]. We use bilinear interpolation of the colors of the reference light rays for the optimal depth:

$$I(\mathbf{r}(\mathbf{x})) = \sum_{i \in V} w_i(\mathbf{x}) I\left(\mathbf{r}_i(\mathbf{x}, z_{\text{opt}}(\mathbf{x}))\right). \tag{6}$$

Here, $w_i(\mathbf{x})$ is the weight for the ith reference light ray $\mathbf{r}_i(\mathbf{x}, z_{\text{opt}}(\mathbf{x}))$, and it takes a floating-point value between 0 and 1 depending on the positions of the reference cameras and the target light ray; $w_i(\mathbf{x})$ takes 1 if the target light ray passes through the ith camera position, while it takes 0 if it passes through the other neighboring camera positions, and $\sum_{i \in V} w_i(\mathbf{x}) = 1$.

Note that the reference camera set V depends on the position of each target light ray x. Therefore, the number of input cameras used for rendering the entire view depends on the desired viewpoint. This rendering method, however, has constant computational complexity regardless of the number of input cameras, because it calculates the color and cost for each target light ray. The computational complexity is determined by the number of target light rays (i.e., the resolution of the desired view) and the number of depth layers.

2.2.2. Reference Region. For synthesizing a novel image, the above rendering method does not require all light rays acquired with the input cameras; instead, it only requires the light rays in reference regions, which we define as segments in the input images that include all of the reference light rays used to synthesize a desired view. When we use the regular camera arrangement shown in Figure 2, the reference regions are described as

$$\left| u - \frac{f}{z_0}(s - s_0) \right| \leq \frac{z_{\min} + z_0}{z_{\min}\,|z_0|} f d, \tag{7}$$

where d is the interval between the input cameras. This means that the reference region in an input image is a rectangular segment whose size is determined by the parameters on the right-hand side of the equation. For an irregular (practical) camera arrangement, the reference regions are similarly defined as quadrangular segments in the input images.

Based on the locality of the reference regions, several camera array systems [8–10] use a region of interest (ROI) approach that only transmits or decodes image segments including the reference regions to reduce the data amount. However, they do not address inter-view prediction. Our method, by contrast, decodes the light rays in the reference regions with inter-view prediction based on a distributed coding approach. Moreover, since the inter-view prediction is incorporated into the geometry estimation in the rendering

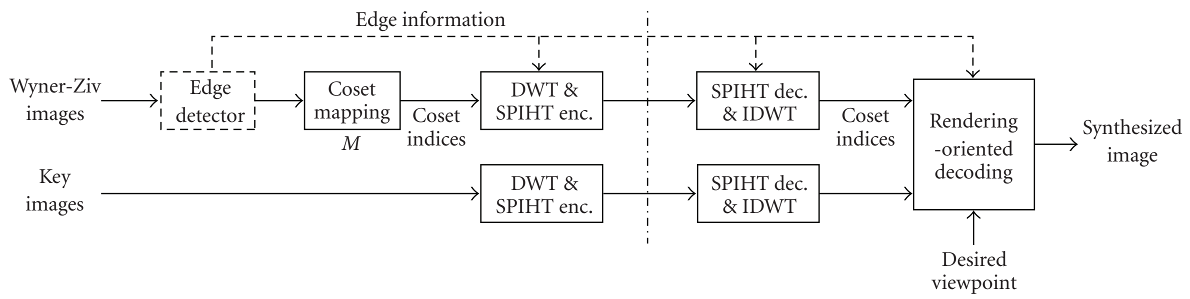


Figure 5: Implementation diagram.

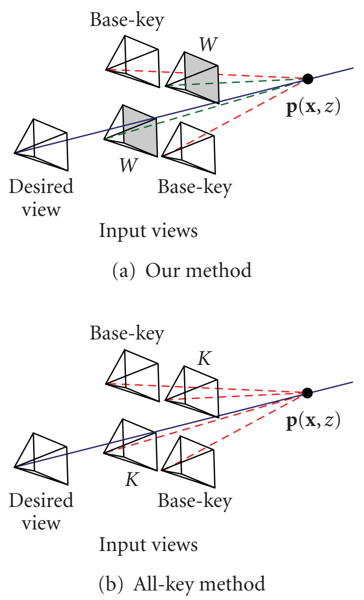


Figure 6: Methods compared in the experiments. Both methods share *base-key* images encoded in the same way at the same positions. The other images, referred to as *nonbase* images, are encoded in different ways.

process, our method keeps the decoder complexity as low as an intracoding method.

3. Rendering-Oriented Decoding

The rendering method described in Section 2.2.1 is applicable if all reference regions are reconstructed and available. Therefore, as shown in Figure 4(a), typical methods first reconstruct the multiview images by using the decoding method described in Section 2.1, and then perform rendering using the reconstructed images. However, they seem to pay an unnecessary computational cost, because disparity compensation in the decoding process and geometry estimation in the rendering process are essentially equivalent if the camera parameters for the multiview images are known, and not all the reconstructed images are used for the rendering.

To synthesize a desired view directly, we propose rendering-oriented decoding method, in which the decoding of the Wyner-Ziv images is incorporated into the rendering process, as shown in Figure 4(b). The Wyner-Ziv images are therefore not reconstructed explicitly, and only the reference light rays in the Wyner-Ziv images are reconstructed implicitly in the rendering process. Our method uses a simple coset code for the Wyner-Ziv images. As with a conventional intracoding method, it keeps both the encoder and decoder low complexity.

3.1. Rendering Method with a Coset Code.

3.1. Rendering Method with a Coset Code. The input multiview images are divided into key images and Wyner-Ziv images. At the encoder, the key images are encoded using a conventional intraimage coder. For the Wyner-Ziv images, each RGB value of a pixel is represented by M cosets, C_m $(m = 1, 2, \ldots, M)$, in a memoryless fashion [18].

At the decoder, we first reconstruct the key images and coset indices for the Wyner-Ziv images. The side information for each target light ray and each depth layer, $Y(\mathbf{x}, z)$, is then calculated by interpolating the colors of the reference light rays in the key images as follows:

$$Y(\mathbf{x}, z) = \frac{\sum_{i \in V_K} w_i(\mathbf{x}) I(\mathbf{r}_i(\mathbf{x}, z))}{\sum_{i \in V_K} w_i(\mathbf{x})}. \tag{8}$$

Here, V_K is the set of camera indices for the key images in the reference camera set V. This side information is used to reconstruct the reference light rays of near Wyner-Ziv images in a maximum likelihood sense by

$$\hat{I}\left(\mathbf{r}_i(\mathbf{x}, z)\big|_{i \in V_W}\right) = \arg \min_{c_j \in C_{m,q}} \left(c_j - Y_q(\mathbf{x}, z)\right)^2 \Big|_{q \in \{R,G,B\}}, \tag{9}$$

where V_W is the set of camera indices for the Wyner-Ziv images in V, and c_j is a codeword in the coset $C_{m,q}$ of the light ray $\mathbf{r}_i(\mathbf{x}, z)|_{i \in V_W}$ for each RGB component q. This equation means that our method reconstructs only the reference light rays in the Wyner-Ziv images. We then evaluate the color consistency cost of the reconstructed reference light rays (3), smooth the cost (4), and estimate the depth and color for each target light ray (5) and (6). Since the extra computational cost for (8) and (9) is not too high, we can keep the complexity of this rendering method as low as that of the original one described in Section 2.2.1. In the experiments, we arranged the key images and Wyner-Ziv images as shown in Figure 1; therefore, $|V_K| = |V_W| = 2$ for all target light rays.

(a) *City* (b) *Santa*

Figure 7: Parts of (a) *City* and (b) *Santa* image sets, which are captured on a regular 2D grid by moving a single camera.

Figure 8: Parts of *Meeting room* image set, which are captured with multiple cameras that roughly lie on a 2D grid.

3.2. Improving Coding Efficiency by Using Edge Information. When the side information for the Wyner-Ziv images is generated, smooth regions can be easily predicted, while edge regions are difficult to predict because of occlusions. In other words, the predicted color (side information) given by (8) is accurate enough in the smooth regions, but it includes a larger error in the edge regions [6]. We therefore use an algorithm that performs the coset decoding only in the edge regions and uses the predicted color itself as the interpolated color in the smooth regions. This reconstruction algorithm is described as follows:

$$\hat{I}(\mathbf{r}_i(\mathbf{x},z)|_{i\in V_W}) = \begin{cases} \arg\min_{c_j\in C_{m,q}} \left(c_j - Y_q(\mathbf{x},z)\right)^2 \Big|_{q\in\{R,G,B\}} & \\ \qquad \text{if } \mathbf{r}_i(\mathbf{x},z) \text{ is in edge regions} \\ Y(\mathbf{x},z), \quad \text{otherwise.} \end{cases} \tag{10}$$

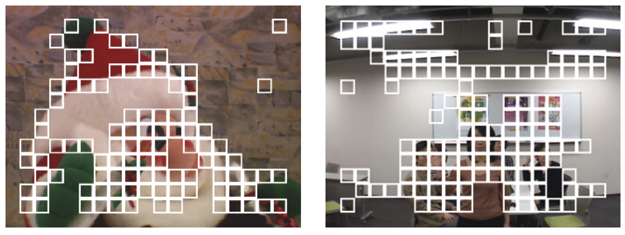


Figure 9: Extracted edge regions in an input image of (a) *Santa* and (b) *Meeting room* image sets.

Table 1: Specifications of the input image sets and parameters of the edge detection and rendering methods used in the experiments.

	City, Santa	*Meeting room*
Number of input images	81 (9×9)	64 (8×8)
Resolution of input images	640×480	320×240
Edge detection block size	32×32	16×16
Edge detection threshold	200	200
Res. of synthesized images	640×480	300×300
Number of depth layers (N)	20	15
Smoothing window size (S)	15×15	11×11

The encoder only needs to send coset indices that correspond to edge regions of the Wyner-Ziv images, as well as mask information that indicates the position of the edge regions. This algorithm therefore improves coding efficiency.

3.3. Implementation. Figure 5 shows the implementation diagram of our method. We encode the key images by using a standard intraimage coder consisting of discrete wavelet transform (DWT) and SPIHT for each RGB component (we used the implementation in QccPack [19]). For the Wyner-Ziv images, we first map each RGB value of a pixel, v_q, to a coset $C_{m,q}$ by the following function:

$$C_{m,q} = \begin{cases} v_q \bmod M, & \text{if } \left\lfloor \dfrac{v_q}{M} \right\rfloor \text{ is even}, \\ M - 1 - (v_q \bmod M), & \text{otherwise}. \end{cases} \tag{11}$$

The coset indices are then encoded with DWT and SPIHT for each RGB component. Since we use the lossy coder for encoding the coset indices, we choose the above mapping function, instead of the regular modulo M function, to prevent drastic changes in codewords with a small error in the coset index. A similar technique is also used in [20]. At the decoder, we decode the SPIHT and perform the rendering-oriented decoding with the key images and the decoded coset indices of the Wyner-Ziv images. In the experiments, we only set M to numbers to the power of two, which is described as $\overline{M} = \log M$.

For exploiting edge information as described in Section 3.2, we implemented a simple edge detector for the Wyner-Ziv images. The Wyner-Ziv images are divided into a set of small rectangular blocks. If the sum of RGB color variances within a block exceeds a threshold, the block is considered as an edge region. The coset indices within the extracted edge regions are encoded by using shape-adaptive SPIHT [19] with a mask image for the edge regions.

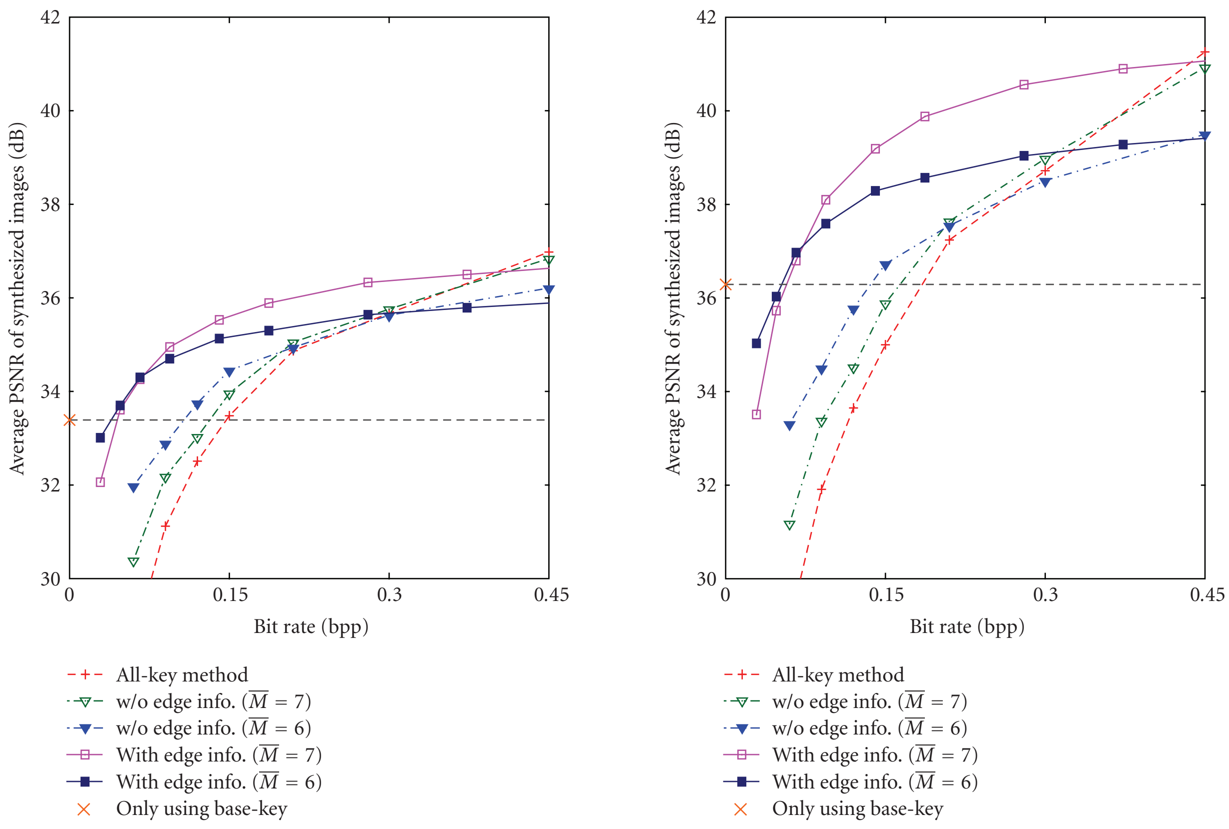
4. Experiments

Compared to a typical method that performs a straightforward decoding and rendering, as shown in Figure 4(a), our rendering-oriented decoding method is of low complexity because it does not perform disparity compensation explicitly and does not reconstruct all of the light rays in the Wyner-Ziv images. Instead, our method has a similar complexity to a method that encodes all images as the key images and synthesizes a novel image with a normal renderer described in Section 2.2.1, which is referred to as *all-key method*. In the following experiments, we therefore compare the coding performance and processing time of these two methods, as shown in Figure 6.

We used two types of input image sets, as shown in Figures 7 and 8. The *City* and *Santa* image sets (Figure 7) are captured by moving a single camera on a control stage, which is an ideal condition for generating accurate side information. Since they are captured on a regular 2D grid with a fixed camera pose, we used a simple geometry for calculating the position of the reference light rays in the input images. On the other hand, the *Meeting room* image set (Figure 8) is captured with our 64-camera array [13], which corresponds to a more practical situation. The image set has large color variations due to individual differences between cameras, and some of them suffer from lens blur. We performed geometry calibration of the cameras by using Tsai's method [21]. For the *Meeting room* image set, we implemented our rendering-oriented decoding method and the all-key method on a GPU (described in Section 4.2 in detail) and evaluated the coding performance and processing time using the GPU implementations. Table 1 summarizes the parameters used in the following experiments, and Figure 9 shows some examples of the edge regions extracted with these parameters.

4.1. Coding Performance. As shown in Figure 6, we divided input images into *base-key* images and the other (*nonbase*) images. The base-key images were identical in both our method and the all-key method; they were encoded by using DWT and SPIHT or assumed to be losslessly available for comparing the influence of the quality of the base-key images on the rendering quality. The nonbase images were encoded as Wyner-Ziv images in our method, as shown in Figure 5, while as key images in the all-key method. The only difference between the two encoding methods is therefore whether they use the coset mapping and edge detection or not. In the experiments, the bit rate of the base-key images was fixed, while that of the nonbase images was controlled by truncating the SPIHT bitstream.

Figures 10, 11, and 12 plot the rate-distortion performance of our method either with or without the edge detector (our method without the edge detector encodes the



(a) *City*, using lossy base-key images (0.45 bpp, 35.77 dB)

(b) *City*, using lossless base-key images

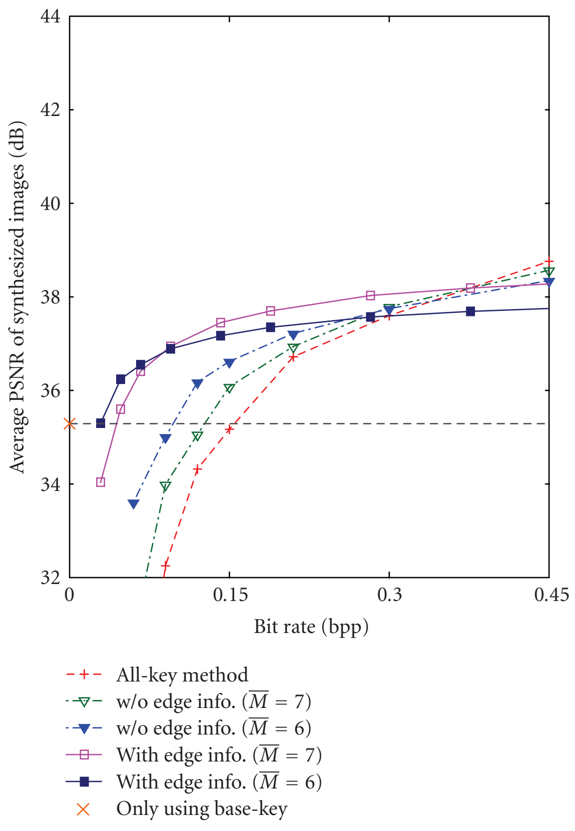
Figure 10: Rate-distortion curves for the *City* image set, obtained using (a) lossy and (b) lossless base-key images. The bit rate of the lossy base-key images was 0.45 bpp and their average quality was 35.77 dB.

coset indices in all regions of the Wyner-Ziv images) and that of the all-key method for different image sets, obtained using lossy and lossless base-key images. The plots show the reconstruction quality of synthesized images averaged for 10 random viewpoints (except the original viewpoints of the key and Wyner-Ziv images), where the quality is calculated with respect to the image synthesized from the uncompressed data and expressed as peak signal-to-noise ratio (PSNR). The bit rate of the nonbase images is expressed on the horizontal axis. The bit rate of edge information is included in the plots of our method using it.

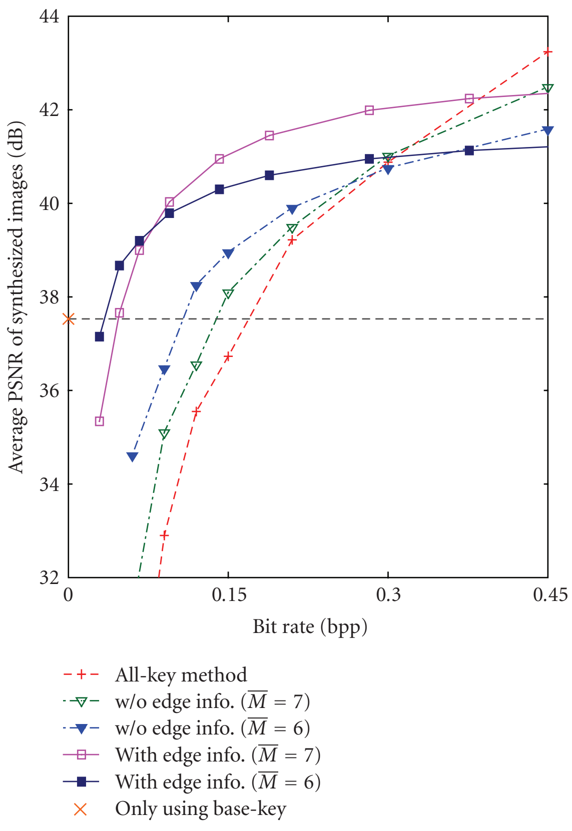
As it can be seen from the plots, our method shows superior coding performance compared to the all-key method especially at low bit rates. Smaller $\overline{M}$ yields better performance at low bit rates, because small errors in the smooth regions can be corrected by a coset code with small $\overline{M}$, but it restricts the maximum quality which is important at high bit rates. As for our method, the edge information provides additional gain at low bit rates, since the edge regions include larger errors than the smooth regions. When comparing the results obtained using the lossy and lossless base-key images, we can see that all of the methods similarly benefit from the increase of the quality of the base-key images, and the shapes of the rate-distortion curves maintain their relationship to each other regardless of the quality of the base-key images.

The plot "only using base-key" in each graph shows the reconstruction quality when we render the novel image by using the base-key images only (i.e., the bit rate of the nonbase images is zero). In this case, the color is interpolated in the same way as for generating the side information (8), and the color consistency cost is calculated as the sum of absolute difference of the reference light ray's colors in the base-key images. This reconstruction quality therefore corresponds to the quality of the side information without error correction. At very low bit rates, our method and the all-key method produce lower-quality images than the side information (under the dashed line). This means that the novel images synthesized at those bit rates are negatively affected from the reconstructed low-quality nonbase images.

This negative effect can be explained with the reconstructed synthesized images and their error images (difference from the synthesized image obtained using uncompressed data), as shown in Figure 13. Here, we used lossless base-key images and set the bit rate of the nonbase images to 0.15 bpp for all methods. If we only use the base-key images, many of the errors appear in the edge regions; in particular, some large structure errors can be seen in those regions (e.g., the bottom-left building in Figure 13(1a) and around the head of the candle in Figure 13(2a)). The all-key method produces larger errors in the smooth regions than the rendering method only using the base-key images (e.g.,



(a) *Santa*, using lossy base-key images (0.45 bpp, 36.75 dB)



(b) *Santa*, using lossless base-key images

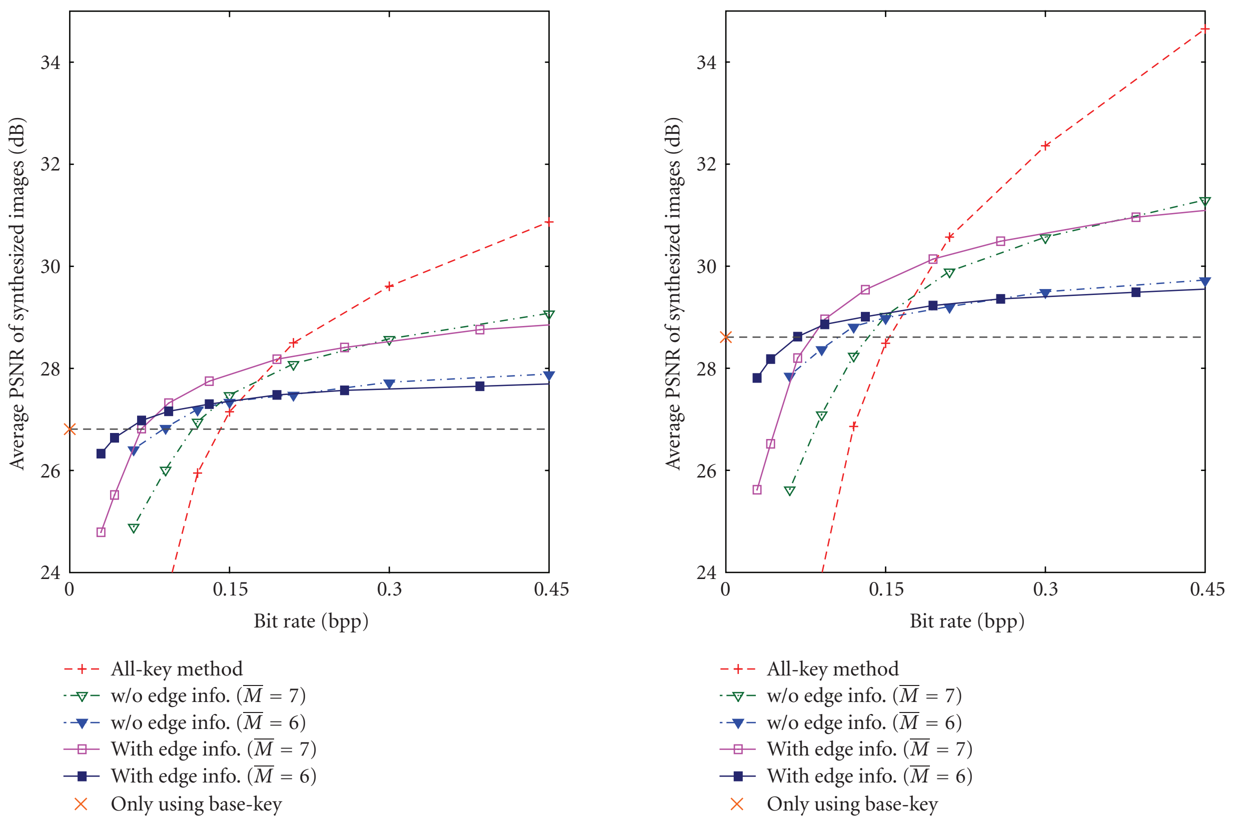
FIGURE 11: Rate-distortion curves for the *Santa* image set, obtained using (a) lossy and (b) lossless base-key images. The bit rate of the lossy base-key images was 0.45 bpp and their average quality was 36.75 dB.

the top-right part (background) in Figure 13(1b)), because it synthesizes the interpolated color with the low-quality nonbase images. The resulting images look blurred, as shown in Figures 13(1b) and 13(2b). Our method without edge information also produces the errors in the smooth regions, but has better PSNR than the all-key method (Figures 13(1c) and 13(2c)). Our method with edge information provides the best reconstruction quality, where the smooth regions keep high quality as using the base-key images only, and errors in the edge regions are reduced (Figures 13(1d) and 13(2d)). The synthesized images obtained using the *Meeting room* image set, depicted in Figure 14, also show similar results; the all-key method produces too blurred images, while our method with edge information produces higher-quality images.

4.2. Processing Time. To compare the processing times of our method and the all-key method, we implemented the two methods on a GPU. For the all-key method, we used the GPU implementation of the rendering algorithm that we developed for real-time video-based rendering using our camera array [13], because all the input images are reconstructed and available before rendering. For the rendering-oriented decoding method, we modified the GPU implementation so that it can perform coset decoding before evaluating the color consistency of reference light rays. The reconstructed coset indices in the Wyner-Ziv image are uploaded to the GPU texture memory as a texture in the RGB channels, as well as the reconstructed key images. When we use edge information, the edge mask for each Wyner-Ziv image is also uploaded as a texture in the alpha channel together with the coset indices in the RGB channels. We used OpenGL and fragment programs with Cg [22] for the GPU implementation. The measurements were performed on an Intel Xeon 5160 (3 GHz) dual processor machine with 3 GB main memory and an NVIDIA GeForce 8800 Ultra graphics card.

Figure 15 shows the processing time versus the number of depth layers for our method and the all-key method. We measured the average processing time for 100 executions of both rendering methods for the *Meeting room* image set. The processing time only includes the coset decoding and rendering processes; that is, the key images and the coset indices in the Wyner-Ziv images were decoded and uploaded to the GPU texture memory before rendering.

The processing time of our rendering-oriented decoding method is proportional to the number of depth layers. This result is the same as that in the case of the original rendering method, which is used for the all-key method. The processing times of our methods with $\overline{M}$ = 6 and 7 are different. This is



(a) *Meeting room*, using lossy base-key images (0.45 bpp, 29.23 dB)

(b) *Meeting room*, using lossless base-key images

Figure 12: Rate-distortion curves for the *Meeting room* image set, obtained using (a) lossy and (b) lossless base-key images. The bit rate of the lossy base-key images was 0.45 bpp and their average quality was 29.23 dB.

because we only need to check two candidates in coset decoding for $\overline{M} = 7$, while we need to check $2^{(8-\overline{M})}$ candidates (or determine which two candidates should be evaluated based on the higher-order bits of the side information) for $\overline{M} < 7$, resulting in higher complexity. The difference between our method and the all-key method is small: our method takes about 7% and 14% more processing time than the all-key method for $\overline{M} = 7$ and 6, respectively. When our method uses edge information, the processing time becomes slightly faster than that without edge information for $\overline{M} = 6$, because we do not need to correct the reference light rays that are not in the edge regions. On the other hand, the processing time becomes slightly slower for $\overline{M} = 7$, because there are only two candidates for the coset decoding and checking if the reference light ray is in the edge regions causes an overhead.

4.3. Discussion. The experimental results show that our method has better coding performance than the all-key method especially at low bit rates, while performing the decoding and rendering as fast as the all-key method. In particular, the coding performance for the *City* and *Santa* image sets shows a clearer advantage of our method than that for the *Meeting room* image set, because the former image sets are suitable for generating accurate side information. Although the *Meeting room* image set has large color variations among input images, which makes it difficult to generate accurate side information, our method still provides higher quality than the all-key method at low bit rates. In such a case, incorporating a color compensation method among input views (e.g., [23, 24]) into the decoding algorithm could help improve coding efficiency.

The experimental results also show that, at very low bit rates, the rendering method only using base-key images provides higher quality than our method and the all-key method. This means that we can choose an appropriate rendering method depending on the bit rate; the rendering method only using base-key images at very low bit rates, our method with the edge detector and a proper number of cosets ($\overline{M}$) at low and medium bit rates, and the all-key method at high bit rates. Since we do not use a feedback channel to control the bit rate of the Wyner-Ziv images [4, 5], to determine the proper number of cosets at the encoder is still difficult and it would be interesting future work.

Our rendering-oriented decoding method has the same feature of the original rendering method; that is, the processing time is proportional to the number of depth layers and target light rays. This is because the coset decoding (8)–(10) can be performed for each target light ray in a desired view, as well as the original rendering process (3)–(6). This feature is suitable for implementing the decoding and rendering processes all on a GPU, because the GPU can efficiently perform the same instructions for all the

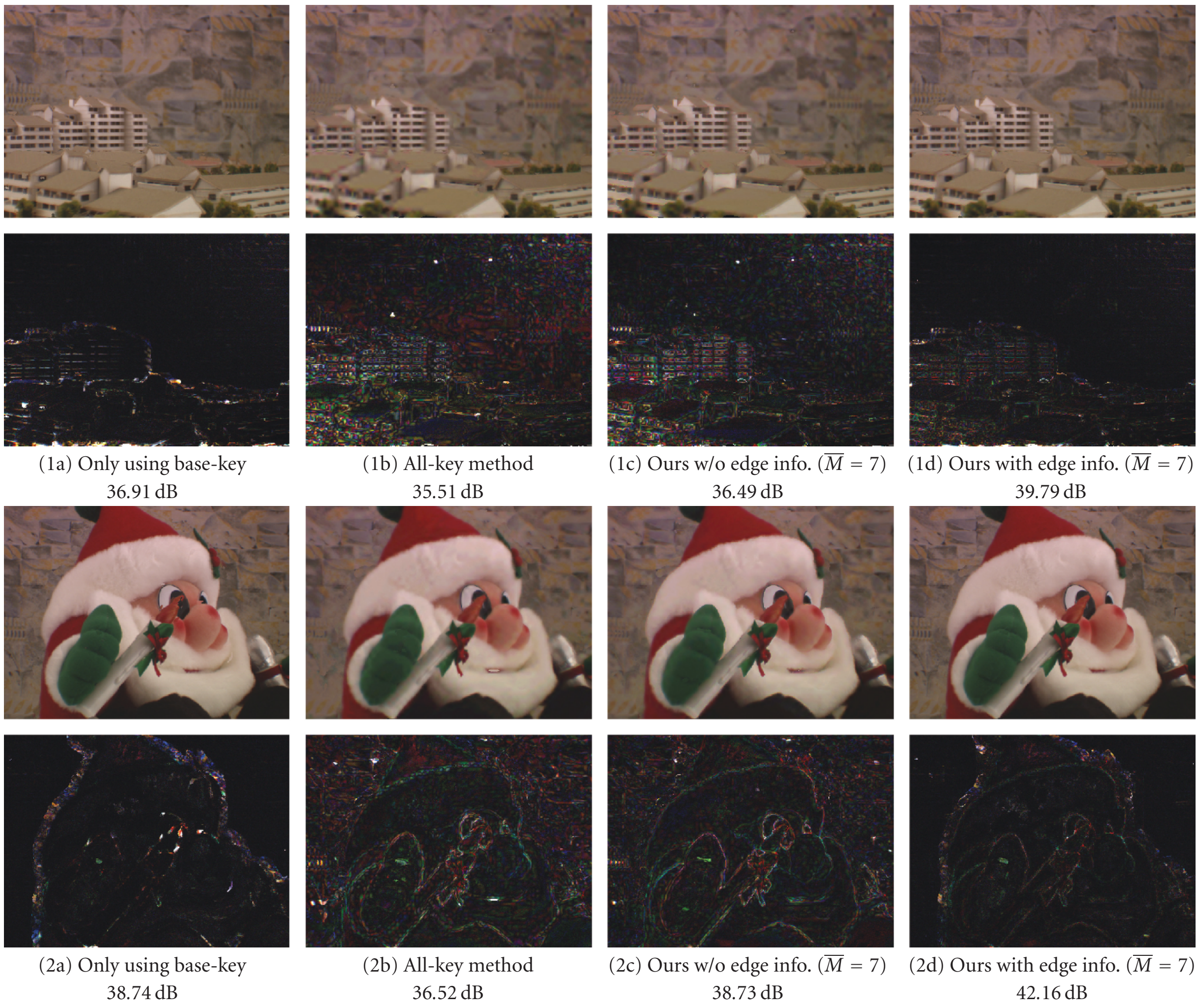
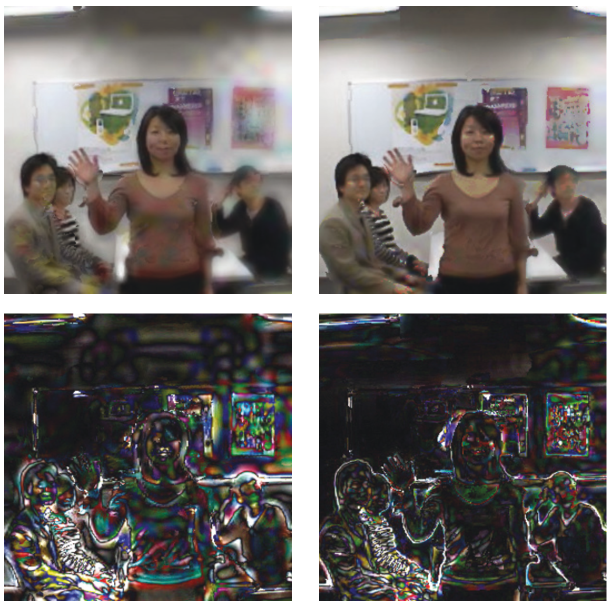

(1a) Only using base-key
36.91 dB
(1b) All-key method
35.51 dB
(1c) Ours w/o edge info. ($\overline{M} = 7$)
36.49 dB
(1d) Ours with edge info. ($\overline{M} = 7$)
39.79 dB
(2a) Only using base-key
38.74 dB
(2b) All-key method
36.52 dB
(2c) Ours w/o edge info. ($\overline{M} = 7$)
38.73 dB
(2d) Ours with edge info. ($\overline{M} = 7$)
42.16 dB

Figure 13: Synthesized images and their difference from that obtained using uncompressed data (multiplied by 8) for the *City* (top) and *Santa* (bottom) image sets.

target pixels in parallel. Thanks to this implementation, our rendering-oriented decoding is fast enough for real-time processing as well as the original rendering method. We have developed a camera array system that enables real-time video-based rendering with the original rendering method [13]. Therefore, if the cameras have a function that maps pixel values to coset indices and encodes them with an intraimage coder (e.g., the Axis 210 camera we used for the camera array has a built-in JPEG encoding function), we could construct a system that performs real-time video-based rendering with improved synthetic quality.

Our method, as well as typical distributed multiview coding methods, would have worse coding performance than conventional methods that perform disparity-compensated prediction at the encoder. However, for the scenario described in this paper (rendering a novel view from encoded data), our method has a clear advantage in computational cost as follows. The conventional method that performs disparity compensation at the encoder needs to separately perform geometry estimation at the decoder for rendering a novel view; there is no way to jointly perform these two processes because the encoder and decoder are separated. The typical distributed multiview coding method performs disparity compensation at the decoder, but still separately performs geometry estimation at the decoder for the rendering, as shown in Figure 4(a). Our method, by contrast, jointly performs disparity compensation and geometry estimation at the decoder, which can make the total computational cost of the encoder and decoder lower than the above two methods.

We compared the coding performance of our method and the all-key method at novel viewpoints, instead of at the viewpoints of the Wyner-Ziv images, because of the following two reasons: (1) to our knowledge, all existing works about distributed multiview coding focus on reconstructing the Wyner-Ziv images; they therefore measure the reconstruction quality at the viewpoints of the Wyner-Ziv images. However, for the free-viewpoint rendering scenario described in this paper, it is more natural to select novel viewpoints that are different from the original viewpoints of


(a) All-key method
27.16 dB

(b) Ours with edge info. ($\overline{M} = 7$)
29.47 dB

Figure 14: Synthesized images and their difference from that obtained using uncompressed data (multiplied by 8) for the *Meeting room* image set.

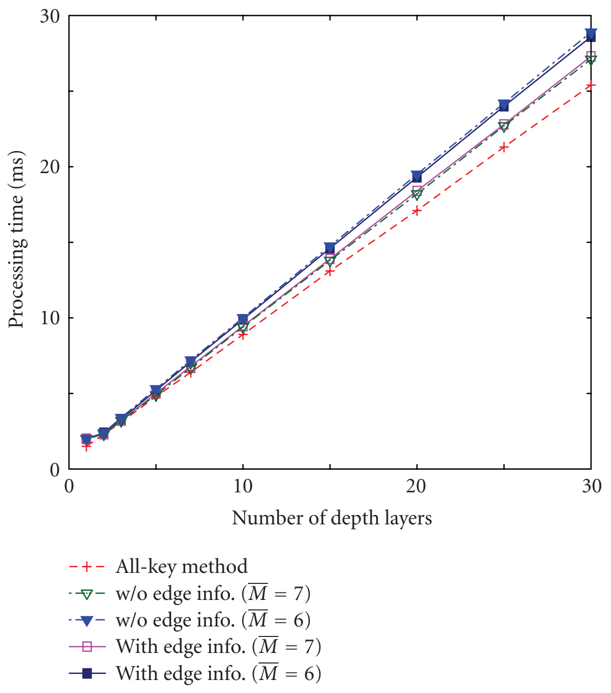


Figure 15: Processing time for different numbers of depth layers.

the key and Wyner-Ziv images; (2) image-based rendering techniques tend to produce images having low PSNR (this does not necessarily mean low visual quality), when we compare the rendered image with the image captured by an actual camera. This is because they do not correctly synthesize view-dependent effects, such as specular components and occluded regions in the scene. Therefore, if we evaluate the reconstruction quality in PSNR at the original viewpoints of the Wyner-Ziv images, our method, which uses an image-based rendering method for reconstructing the images, has a disadvantage compared to the all-key method, which uses the encoded key images themselves as the reconstructed images. If we evaluate the quality at novel viewpoints, as we did in this paper, the disadvantage is avoided, because both our method and the all-key method use an image-based rendering method for the reconstruction and the reference images are also synthesized with the same image-based rendering method (i.e., the view-dependent effects decrease in both the reference images and the reconstructed images).

5. Conclusions

In this paper, we have presented rendering-oriented decoding method for a distributed multiview coding system using a coset code. By incorporating the reconstruction of reference light rays in the Wyner-Ziv images into the rendering process, our method directly synthesizes a novel image without reconstructing all the Wyner-Ziv images explicitly. Our method keeps both encoder and decoder complexity as low as that of a conventional intracoding method, while attaining better coding performance especially at low bit rates. Our future work will be focused on finding a way to incorporate the rendering-oriented decoding method into a real-time video-based rendering system.

Acknowledgments

The authors would like to thank Prof. Hiroshi Harashima and Keita Takahashi for valuable discussions, and the anonymous reviewer for helpful comments that improved the presentation of this paper. The *City* and *Santa* image sets are from the multiview image database provided by courtesy of University of Tsukuba, Japan. A preliminary version of this paper appeared in [25].

References

[1] H.-Y. Shum, S. B. Kang, and S.-C. Chan, "Survey of image-based representations and compression techniques," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 11, pp. 1020–1037, 2003.

[2] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, and C. Zhang, "Multiview imaging and 3DTV," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 10–21, 2007.

[3] A. Jagmohan, A. Sehgal, and N. Ahuja, "Compression of lightfield rendered images using coset codes," in *Proceedings of the 37th Asilomar Conference on Signals, Systems and Computers*, vol. 1, pp. 830–834, Pacific Grove, Calif, USA, November 2003.

[4] A. Aaron, P. Ramanathan, and B. Girod, "Wyner-Ziv coding of light fields for random access," in *Proceedings of the 6th IEEE Workshop on Multimedia Signal Processing (MMSP '04)*, pp. 323–326, Siena, Italy, September 2004.

[5] X. Guo, Y. Lu, F. Wu, W. Gao, and S. Li, "Distributed multi-view video coding," in *Visual Communications and Image*

Processing 2006, vol. 6077 of *Proceedings of SPIE*, pp. 1–8, San Jose, Calif, USA, January 2006.

[6] Z. Jin, M. Yu, G. Jiang, X. Zeng, and Y.-D. Kim, "ROI-based Wyner-Ziv coding with low encoding complexity for wireless multiview video sensor array," in *Proceedings of the 25th Picture Coding Symposium (PCS '06)*, pp. P1–P21, Beijing, China, April 2006.

[7] T. Naemura and H. Harashima, "Real-time video-based rendering for augmented spatial communication," in *Visual Communications and Image Processing 1999*, vol. 3653 of *Proceedings of SPIE*, pp. 620–631, San Jose, Calif, USA, January 1999.

[8] H. Schirmacher, M. Li, and H.-P. Seidel, "On-the-fly processing of generalized lumigraphs," in *Proceedings of the European Association for Computer Graphics (Eurographics '01)*, vol. 20, pp. 165–173, Manchester, UK, September 2001.

[9] J. C. Yang, M. Everett, C. Buehler, and L. McMillan, "A real-time distributed light field camera," in *Proceedings of the 13th Eurographics Workshop on Rendering*, pp. 77–85, Pisa, Italy, June 2002.

[10] C. Zhang and T. Chen, "A self-reconfigurable camera array," in *Proceedings of the 15th Eurographics Symposium on Rendering*, pp. 243–254, Norrkoping, Sweden, June 2004.

[11] B. Wilburn, N. Joshi, V. Vaish, et al., "High performance imaging using large camera arrays," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 765–776, 2005.

[12] T. Fujii, K. Mori, K. Takeda, K. Mase, M. Tanimoto, and Y. Suenaga, "Multipoint measuring system for video and sound—100-camera and microphone system," in *Proceedings of IEEE International Conference on Multimedia and Expo (ICME '06)*, pp. 437–440, Toronto, Canada, July 2006.

[13] Y. Taguchi, K. Takahashi, and T. Naemura, "Real-time all-in-focus video-based rendering using a network camera array," in *Proceedings of 3DTV Conference: The True Vision—Capture, Transmission and Display of 3D Video*, pp. 241–244, Istanbul, Turkey, May 2008.

[14] M. Levoy and P. Hanrahan, "Light field rendering," in *Proceedings of the 23rd ACM Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pp. 31–42, New Orleans, La, USA, August 1996.

[15] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph," in *Proceedings of the 23rd ACM Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pp. 43–54, New Orleans, La, USA, August 1996.

[16] K. Takahashi and T. Naemura, "Layered light-field rendering with focus measurement," *Signal Processing: Image Communication*, vol. 21, no. 6, pp. 519–530, 2006.

[17] C. Buehler, M. Bosse, L. McMillan, S. J. Gortler, and M. F. Cohen, "Unstructured lumigraph rendering," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*, pp. 425–432, Los Angeles, Calif, USA, August 2001.

[18] S. S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): design and construction," *IEEE Transactions on Information Theory*, vol. 49, no. 3, pp. 626–643, 2003.

[19] "QccPack—quantization, compression, and coding library," http://qccpack.sourceforge.net.

[20] R. Bernardini, R. Rinaldo, P. Zontone, D. Alfonso, and A. Vitali, "Wavelet domain distributed coding for video," in *Proceedings of IEEE International Conference on Image Processing (ICIP '06)*, pp. 245–248, Atlanta, Ga, USA, October 2006.

[21] R. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE Journal of Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.

[22] http://developer.nvidia.com/page/cg_main.html.

[23] K. Yamamoto, M. Kitahara, H. Kimata, et al., "Multiview video coding using view interpolation and color correction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 11, pp. 1436–1449, 2007.

[24] J. H. Kim, P. Lai, J. Lopez, et al., "New coding tools for illumination and focus mismatch compensation in multiview video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 11, pp. 1519–1535, 2007.

[25] Y. Taguchi and T. Naemura, "Rendering-oriented decoding for distributed multi-view coding system," in *Proceedings of the 14th IEEE International Conference on Image Processing (ICIP '07)*, vol. 1, pp. 213–216, San Antonio, Tex, USA, September-October 2007.