3-D VIDEO CODING SYSTEM WITH ENHANCED RENDERED VIEW QUALITY

by

Woo-Shik Kim

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(ELECTRICAL ENGINEERING)

August 2011

Dedication

To my lovely Rachel

To my lovely Lynn

To my lovely Su

And to my parents

Acknowledgements

First I would like to express my sincere gratitude to my advisor, Professor Antonio Ortega, for his excellent guidance. Through his invaluable teaching I could learn how to set up and develop an original research project. His vigorous challenges and insightful advice not only formed me as a Ph. D. but also will be priceless asset to my ongoing research career. I also appreciate committee members, Professors C.-C. Jay Kuo, Ulrich Neumann, Shrikanth Narayanan, and Jerry M. Mendel for their valuable comments during the Ph. D. defense and qualifying examination.

It has been my great pleasure to communicate with many Ph. D. students, alumni, and other researchers in my field during my Ph. D. study. Especially, I thank Dr. PoLin Lai, Dr. Godwin Shen, Sunil Narang, Dr. Anthony Vetro, Dr. Dong Tian, and Prof. Gene Cheung for the collaboration during my research work. I also thank for help, support, and friendship to many people. I thank to Prof. Rae-Hong Park who inspired me as a researcher, to Dr. Hyun Mun Kim who helped me to start Ph. D. study. Thanks to Hyung-Suk Kim, Dr. Ngai-Man Cheung, Dr. Jae Hoon Kim, Dr. Roger Pique, Dr. In Suk Chong, Dr. Hye-Yeon Cheong, Dr. Alexis Tourapis, Dr. Zihong Fan, Dr. Sean McPherson, Sungwon Lee, SeongHo Cho, Jewon Kang, Hyunsuk Ko, Dr. Do-Kyoung Kwon, Dr. Byung-Ho Cha, Dr. Jongdae Oh, Joohyun Cho, and many others for their help and friendship. Also thanks to Rev. John Kim and other friends in St. James church for their love and spiritual support.

I gratefully acknowledge financial support during my Ph. D. from Dept. Electrical Engineering, USC, Thomson Corporate Research, Samsung Advanced Institute of Technology, and Texas Instruments. I also thank Mitsubishi Electric Research Laboratories, Dolby Laboratories, and Thomson Corporate Research for providing research internships.

Finally, I want to express my heartful thanks to my family. Without their love and sacrifice this would have not been possible. I love you so much.

Table of Contents

List of Tables

List of Figures

Abstract

The objective of this research is to develop a new 3-D video coding system which can provide better coding efficiency with improved subjective quality as compared to existing 3-D video systems such as the depth image based rendering (DIBR) system. Clearly, one would be able to increase overall performance by focusing on better "generic" coding tools. Instead, here we focus on techniques that are specific of 3-D video. Specifically, we consider improved representations for depth information as well as information that can directly contribute to improved intermediate view interpolation.

As a starting point, we analyze the distortions that occur in rendered views generated using the DIBR system, and classify them in order to evaluate their impact on subjective quality. As a result, we find that the rendered view distortion due to depth map coding has non-linear characteristics (i.e., increases in intensity errors in the interpolated view are not proportional to increases in depth map coding errors) and is highly localized (i.e., very large errors occur only in a small subset of pixels in a video frame), which can lead to significant degradation in perceptual quality. A flickering artifact is also observed due to temporal variation of depth map sequence.

To solve these problems, we first propose new coding tools which can reduce the rendered view distortion by defining a new distortion metric to derive relationships between distortions in coded depth map and rendered view. In addition, a new skip mode selection method is proposed based on local video characteristics. Our experimental results show the efficiency of the proposed method with coding gains of up to 1.6 dB in interpolated frame quality as well as better subjective quality with reduced flickering artifacts.

We also propose a new transform coding using graph based representation of a signal, which we name as graph based transform. Considering depth map consists of smooth regions with sharp edges along object boundaries, efficient transform coding can be performed by forming a graph in which the pixels are not connected across edges. Experimental results reveal that coding efficiency improvement of 0.4 dB can be achieved by applying the new transform in a hybrid manner with DCT to compress a depth map.

Secondly, we propose a solution in which *depth transition data* is encoded and transmitted to the decoder. Depth transition data for a given pixel indicates the camera position for which this pixel's depth will change. For example in a pixel corresponding to foreground in the left image, and background in the right image, this information helps us determine in which intermediate view (as we move left to right), this pixel will become a background pixel. The main reason to consider transmitting explicitly this information is that it can be used to improve view interpolation *at many different intermediate camera positions.* Simulation results show that the subjective quality can be significantly improved using our proposed depth transition data. Maximum PSNR gains of about 2 dB can also be observed. We foresee further gains as we optimize the amount of depth transition data being transmitted.

Chapter 1

Introduction

3-D video services are becoming reality in the consumer market thanks to recent improvements on high quality 3-D display technologies and 3-D content production skills, which is a significant step towards a more realistic multimedia experience [13, 52]. For the advanced 3-D video services, two application scenarios have been discussed [67]; first, providing ability to vary the baseline distance for stereo video to adjust the depth perception to help avoiding fatigue and other viewing discomforts, and secondly, supporting auto-stereoscopic displays, which do not require for the viewer to wear special glasses to feel the depth. In both cases, multiple number of views are required at the display side.

As 3-D video requires multiple video sequences captured from different camera positions, it becomes challenging to transmit and store such a large amount of data. This has lead to significant interest in investigating efficient view rendering methods. For this purpose it has been studied how to sample the plenoptic function or the light field [1, 80] to reconstruct missing views using image based rendering techniques [6]. To improve the rendered view quality, multiview video plus depth formats are being developed, where only selected views are coded along with their corresponding depth maps, and other views are interpolated at the decoder using depth image based rendering (DIBR) technique [76, 83].

Since depth maps have to be sent to the decoder, it is necessary to develop efficient compression methods for depth maps. Even though a depth map can be treated as a gray scale image, and compressed using standard image or video coding techniques, better coding efficiency can be achieved if depth-map-specific characteristics are exploited, since some of these properties are quite different from those of standard image or video. Many researchers have aimed to improve the efficiency of depth map coding by exploiting these characteristics. For example, typical depth maps tend to lack texture so that they can generally be well approximated as piecewise smooth signals, with relatively constant depth areas separated by sharp edges where each smooth depth region may correspond to an object at a different depth. Many approaches have been proposed along these lines. Morvan *et al* [29] and Merkle *et al* [28] used platelet coding, and Maitre and Do [27], Daribo *et al* [10], and Sanchez *et al* [44] used edge-adaptive wavelet transforms, which seek to design transform that avoid filtering across edges or object boundaries in a depth map. In our work, we have shown that better coding efficiency can be achieved by edge adaptive coding tools such as the edge adaptive intra prediction [50] and the edge adaptive transform [49]. Other such approaches include compressed sensing based methods [45] and the reduced resolution with edge-preserving upsampling [12]. Besides, other depth map specific characteristics are utilized to achieve better coding efficiency in the researches, such as dynamic range reshaping [23], 3-D motion estimation [62], warping based inter-view prediction [30], reuse of video motion information to reduce encoding complexity [34], and sparsity-based in-loop filtering [24].

In these previous works depth-map-specific characteristics are utilized to improve coding efficiency. However, it is also very important to understand how the depth map is used for the view synthesis. In standard video compression, quantization errors directly affect the rendered view quality by adding noise to the luminance or chrominance level of each pixel. In contrast, the distortion in the depth map will affect indirectly the rendered view quality. Considering that the depth map itself is not displayed but used to provide additional geometry information to help the view synthesis process, it would be inefficient to maintain same level of quality for the whole depth map area using the aforementioned methods. For example, a small amount of geometry error can cause large distortion in the rendered view in regions of complex texture or at object boundaries with high contrast to the background intensity. Instead, large amounts of geometry error can lead to negligible artifacts in the rendered view in the case of flat regions. Therefore, it is crucial to analyze the relationship between the depth map error and the distortion in the rendered view in order to achieve optimal performance of depth map coding.

There has been research to evaluate the impact of depth error to the rendered view distortion. Müller *et al* [31] studied experimentally the impact of bitrate distribution between the video and depth map on the rendered view quality, but no theoretical analysis was given to find the relationship between the depth map error and the rendered view distortion. Merkle *et al* [28] measured geometry error caused by depth map error by calculating the distance between the 3-D surfaces constructed from the original and the coded depth map, respectively, using the Hausdorff distance; however, no method was given to find how the geometry error causes the rendered view distortion, and the depth map distortion itself is used to optimize the depth map coding.

Nguyen and Do [33] derived an upper bound on the mean squared error in the rendered view due to geometry error using Taylor series expansion. Ramanathan and Girod [40] used the power spectral density with Gaussian modeling of image signal in order to estimate the distortion in rendered view due to geometry error, where global relationship can be found between the geometry error and the rendered view distortion. These approaches can be used to analyze the effect of geometry error on the rendered view quality; however, both the global Gaussian model and the upper bound derivation do not provide a precise estimation of how local depth coding distortion (e.g., within a block) lead to distortion in the rendered view. Because of this, distortion estimates obtained using the aforementioned methods may not be sufficiently accurate for rate-distortion (RD) optimization in depth map coding.

In our previous work [18], the relationship between depth map error, geometry error, and distortion in the rendered view was analyzed, with a global linear model used to characterize the distortion in the rendered view as a function of depth map error. A problem with such a global distortion is that there may exist significant local mismatches, since the rendered view distortion varies according to local characteristics of the reference video. For example, the amount of the distortion caused by the geometry error will be small for a smooth region of the video as compared to a region with complex textures.

In this thesis, we start by focusing on the different types of distortion affecting the DIBR process. Clearly, the quality of decoded video frames that are used for view interpolation is an important factor, but since they are likely to be coded using standard tools (e.g., H.264/AVC) it is relatively simple to control their distortion. Instead, we focus on how the quality of depth maps transmitted to the decoder affects overall quality, which has non-linear characteristics (i.e., increases in intensity errors in the interpolated view are not proportional to increases in depth map coding errors) and is highly localized (i.e., more significant errors occur only in a small subset of pixels in a video frame).

The main contribution of our research is to improve the rendered view quality using the new coding tools and new data format specifically designed to reduce the distortion occurred in the rendered view. First, we propose a simple and precise local estimation method to estimate the distortion generated in the rendered view due to depth map coding. Of course, the rendered view distortion can be exactly measured if the intermediate view is synthesized and compared to the ground truth. However, this is not practical, since the ground truth for an arbitrary view position may not exist, and the view synthesis process would be too complex to be used during the depth map coding. Instead, we propose a simple and precise estimation method, which reflects local video characteristics, so that it can be used during depth map coding to achieve optimal performance.

First, we derive a relationship between errors in the depth map and geometry errors in rendering using the camera parameters. We then estimate the resulting intensity distortion in the rendered view. The estimation has to reflect local video characteristics corresponding to the local depth map area. For example, if there is a depth error, but the intensity of the image to be synthesized using the depth information is locally almost constant, distortion caused by depth error will be very small. On the other hand, even one pixel displacement caused by depth error can cause large distortion in the rendered view, if the image area to be synthesized consists of object boundaries or complex textures. This will be more noticeable if there is high contrast between different objects or within the texture.

Two methods are proposed to accurately estimate the rendered view distortion by reflecting the local video characteristics. In the first method, the geometry error is translated into a "pixel displacement" in the interpolation process, i.e. pixels used for interpolation are shifted with respect to those that would be chosen if the exact depth map were used. Then, the rendered view distortion is estimated by computing the error between each pixel and the pixel in a shifted position, where the shift corresponds to the geometry error caused by the exact depth error at that pixel location. This method provides precise estimation results at the cost of random pixel access in the reference frame memory, which would break the burst access mode of dynamic random access memory. In the second method, instead of accessing pixel by pixel, the local video characteristics are modeled using an autoregressive model, so that the estimation can be done with reduced computational complexity for real time processing. These approaches are thus local in nature, taking into account both the local value of depth map distortion *and* local video characteristics.

Then, the proposed distortion metrics are applied to depth map coding to improve the coding efficiency of the 3-D video system. To select the optimal coding mode for the depth map the Lagrangian optimization is performed with the proposed distortion metrics. The new Lagrange multiplier is also derived using the proposed distortion metric based on the autoregressive model as a function of the quantization step size. Note that the proposed methods are not restricted to a specific coding method, but can be applied to various coding methods to achieve optimal results.

Experiments are performed using various test materials to evaluate the performance of the proposed methods. Improved coding efficiency is observed when applying our proposed methods with the average peak signal to noise ratio (PSNR) gains of 0.6 and 0.2 dB with the first and the second estimation methods, respectively, when compared to the standard H.264 [54] approach without any rendered view distortion estimation.

We also propose a new skip mode for depth map coding, which can reduce the flickering artifact in the rendered view. The flickering artifact occurs when the temporal variation is larger in depth map than in the video due to lack of precision in depth estimation. By choosing a skip mode in depth map coding based on video information, the flickering artifact can be efficiently suppressed as shown in the experimental results.

To improve the efficiency of depth map coding we also propose a new transform coding based on graph representation of signal. Considering a depth map consists of smooth regions separated by sharp edges along the object boundaries, a graph can be formed so that no connection is made among pixels across those edges in a given coding block. Then, the eigenvectors of Laplacian matrix of the graph can be used as transform kernel. We name this new transform coding as the graph based transform (GBT). The advantage of GBT is that it can be applied to the block based coding, which is a dominant design of standard video codecs such as MPEG-4 and H.264. We propose to apply GBT and DCT in a hybrid form so that the best transform can be applied to each block. With the proposed method 0.4 dB PSNR gain or 21 % bitrate saving is observed on average when applied to various depth map sequences.

One of the key novelties in our research is to propose a new 3-D video format in which additional information is sent to the decoder (in addition to video from selected views and depth information). Based on the observation that the rendered view distortion due to depth map errors has non-linear and localized characteristics, we develop a new data format, *depth transition data* (DTD) [20, 21]. To provide some intuition, this type of data considers a scenario where reference views are transmitted and depth information for them is available. Focusing on a specific pixel in one view, we can determine its corresponding location in the other view. Given their corresponding depth information, if depth information for the second view indicates a significantly different depth value than for the first view, this is a sign that there is a depth transition somewhere between these two views. For these situations we transmit to the decoder the *view location* at which this transition occurred (from an object at a certain depth to another at a different depth).

Providing explicit information about where this transition occurs has several advantages. First, it leads to improved interpolation for *arbitrary* intermediate camera positions. Second, in cases where no depth information is available for intermediate views, DTD could also be derived from depth information at the transmitted views. However, by explicitly representing transition information we are able to improve quality (by using more bits) at those locations where this information matters the most. Finally, in cases where depth or video information is available for intermediate views (but is not going to be transmitted), the DTD can be corrected using these information, so that improved interpolation can be achieved without having to transmit additional depth maps or video. Our experimental results show that our proposed DTD representation can lead to significant perceptual quality improvement in the rendered frames, with PSNR gain of up to 2 dB.

The rest of the thesis is organized as follows. The distortion in the rendered view is analyzed in Chapter 2. The proposed rendered view distortion estimation methods and the new skip mode method are described in Chapter 3. The new 3-D video format is described in Chapter 4. Conclusions and future work are discussed in Chapter 5.

Chapter 2

Distortion Analysis in Rendered View

In this chapter, the view synthesis process using depth map is reviewed, and the distortion in the rendered view is analyzed. Through the analysis various factors affecting the rendered view quality are determined. Among them we focus our effort on investigating the relationship between the depth map error and rendered view distortion, since the analysis reveals that the depth map error has great impact on rendered view quality. We qualitatively analyze the depth map quantization error, which reveals non-linear and localized features of the rendered view distortion from depth map error, and propose a quality measurement method to reflect these features. In addition, flickering artifacts are observed due to depth map estimation error, and we propose a quantitative way to measure this.

2.1 View synthesis process using depth maps

In a DIBR system, a few views are transmitted to the decoder, while the other views are synthesized using the decoded views [76,83]. A depth map sequence is included for each transmitted view along with video sequence in order to improve the quality of the synthesized video quality. To analyze the distortion in the rendered view,

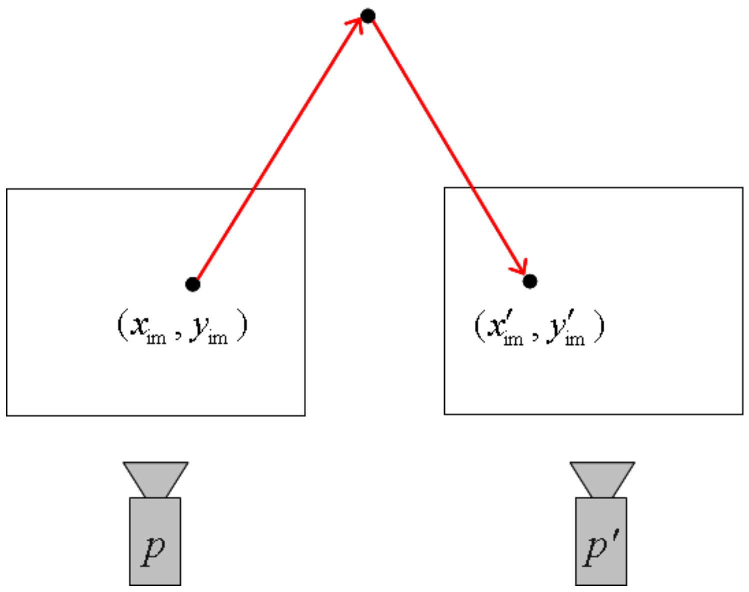

$(x_{\mathrm{im}}, y_{\mathrm{im}})$
$(x'_{\mathrm{im}}, y'_{\mathrm{im}})$
p
p'

Figure 2.1: Mapping of a pixel from reference view (p) to the world coordinate and to the target view (p'), where $(x_{\mathrm{im}}, y_{\mathrm{im}})$ and $(x'_{\mathrm{im}}, y'_{\mathrm{im}})$ are the location of pixel in the reference view and the target view, respectively.

it will be helpful to understand how an intermediate view is generated using the neighboring views that are available at the decoder side. In this section the view synthesis process using depth map is reviewed.

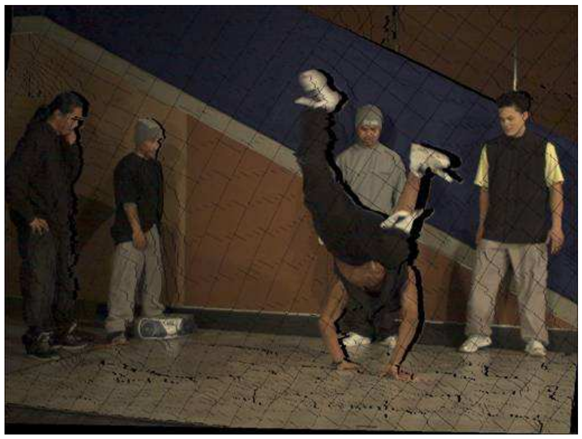
To synthesize a view (target view) using the decoded view (reference view), each pixel in the decoded video frame can be mapped to the target view using the camera parameters, such as baseline distance, focal length, and rotation and translation matrix, and per-pixel depth value. This is done by first mapping from the reference view to the world coordinate, and then mapping from the world coordinate to the target view, as illustrated in Fig. 2.1. A more detailed mapping procedure is given in Section 3.1.1.

To improve rendered view quality, it is important to obtain depth information with enough precision. There are various ways to acquire depth information. For example per-pixel depth values can be estimated from neighboring video using stereo triangulation, or it can be directly captured using various range cameras such as laser scanner and time-of-flight camera [4]. Per-pixel depth values can be converted into depth map value for efficient storing. The relationship between the actual depth value, z, and the 8-bit depth map value, $L_p(x_{\text{im}}, y_{\text{im}})$, is given as

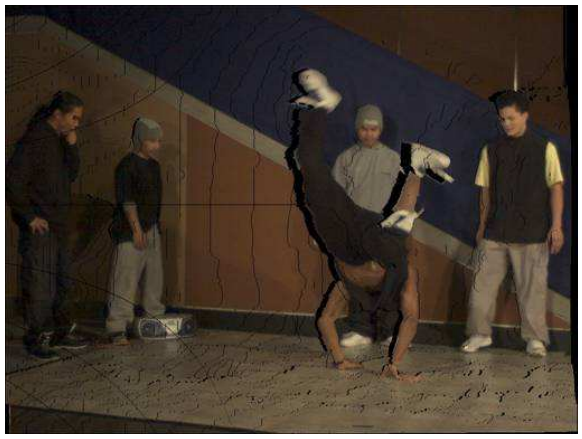
$$z = \left(\frac{L_p(x_{\text{im}}, y_{\text{im}})}{255} \left(\frac{1}{Z_{\text{near}}} - \frac{1}{Z_{\text{far}}} \right) + \frac{1}{Z_{\text{far}}} \right)^{-1}, \tag{2.1}$$

where Z_{near} and Z_{far} are the nearest and the farthest clipping planes, which correspond to value 255 and 0 in the depth map, respectively, with the assumption that z, Z_{near} and Z_{far} have all positive or all negative values [66].

Figs. 2.2 and 2.3 show an example of the view synthesis procedure [57], where Fig. 2.2 (a) and (b) are the warped views from left and right views to the target view, and Fig. 2.3 is generated by combining these two warped views. When all the pixels in the reference view are mapped to the target view, it is possible that multiple of them are mapped to the same position in the target view. On the other hand it is also possible that none of them can be mapped to certain positions in the target view, which is called as the hole area. The hole area in the left warped view is filled from the right warped view, and vice versa. Methods to generate the target view by combining more than one warped view are called as a blending processes, in which a weighted averaging can be used with the baseline distance or depth value of each reference pixel used as the weight. It is also possible to select only one reference pixel to avoid blurring artifact. When there remains a hole area after blending, it can be filled using a hole filling process, e.g., based on the inpainting [2].



(a) Left reference view warped to the target view



(b) Right reference view warped to the target view

Figure 2.2: View synthesis procedure by warping neighboring reference views.

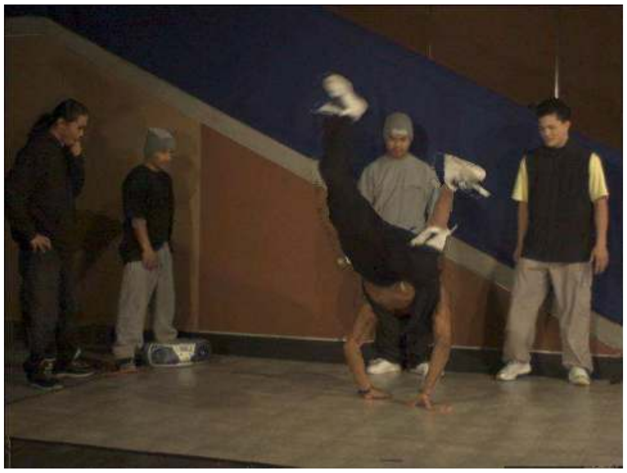


Figure 2.3: Rendered view after blending left and right reference views warped to the target view.

2.2 Classification of distortion in rendered view

In a DIBR system, reference video sequences and corresponding depth maps are used to synthesize an intermediate view. We have analyzed the distortion in the rendered view and found various factors that affect the rendered view quality. Distortion in the rendered view can be due to:

- *Distortion in the reference video.* This distortion could be due to sensor noise during the acquisition of the reference video, as well as quantization error due to coding. These distortions will be directly reflected in the rendered view. The quantization error will affect the rendered view quality significantly and is controllable by adjusting a quantization step size at the encoder side. Therefore, how to choose the quantization step size will be an optimization problem at the encoder side, which can be addressed with extensions of conventional techniques.

- *Distortion due to synthesis.* We have found two main reasons for this inaccuracy. First, when we interpolate a view between left and right reference views, an occlusion may occur, especially for an object near the camera or the area behind this object, due to large difference in projection angle. In this case each reference view can compensate the occlusion caused by the other reference view, but this can result in large distortion, especially when the reference views have different illumination conditions. Secondly, the blending process to generate the synthesized pixel value using reference pixels can cause distortion. For example, by averaging left and right references, blurring can occur.

- *Distortion due to depth map inaccuracy.* This could be due to inaccuracy in the initial depth estimation as well as quantization errors due to lossy representation of the depth map, which result in geometry error.

Since the first factor is directly controllable at the encoder side, the focus of the analysis is on the other two factors.

To analyze the effect of various distortion sources, the subjective quality is evaluated using Ballet sequence as shown in Figs. 2.4-2.7, where Fig. 2.4 is the original 5th view, Fig. 2.5 is the rendered view using 4th and 6th view without coding of video and depth maps, Fig. 2.6 is the rendered view with coded depth map and uncompressed video, and Fig. 2.7 is the difference image between Fig. 2.5 and Fig. 2.6. For depth map coding H.264/AVC [54] is used with quantization parameter (QP) set to 36.

First, by comparing Fig. 2.4 and Fig. 2.5 it can be observed that the view synthesis process can cause the following types of distortion *even without any coding involved*:

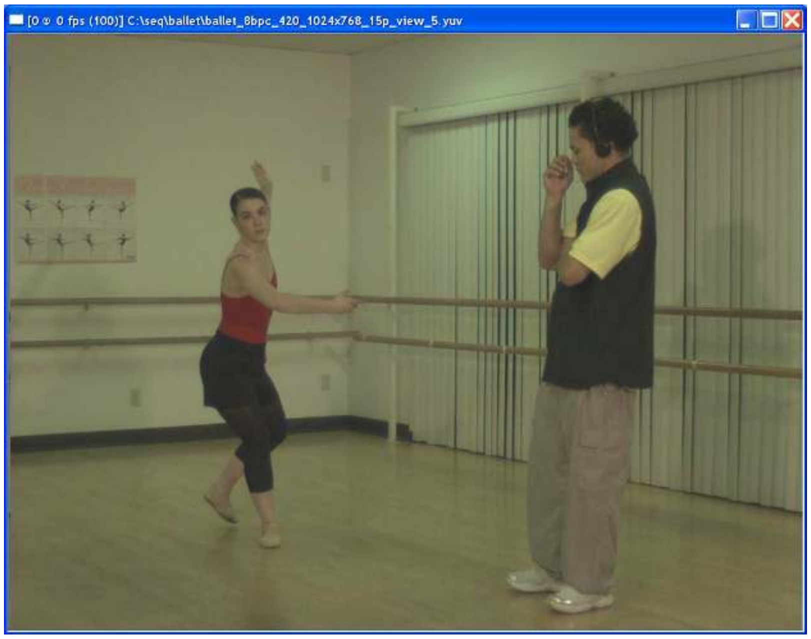

[0 @ 0 fps (100)] C:\seq\ballet\ballet_8bpc_420_1024x768_15p_view_5.yuv

Figure 2.4: Subjective quality comparison of Ballet sequence; Original 5th view

- High frequency components tend to be blurred throughout the image due to the blending process during the view synthesis.
- There are very clear errors in the occluded regions, for which the blending is not used, and only one reference video is used instead to fill those regions. This is due to differences in illumination between different views (if no averaging is performed across views, the illumination of the interpolated will be similar to that of one of the reference view, which may not reflect the true illumination).
- There exists distortion around object boundaries. This is due to occlusion and/or depth map estimation error.

Secondly, from Fig. 2.6 it can be observed that severe distortion occurred along the object boundary. Since the difference between Fig. 2.5 and Fig. 2.6 lies only in depth map coding, we can infer that this artifact comes from the depth map

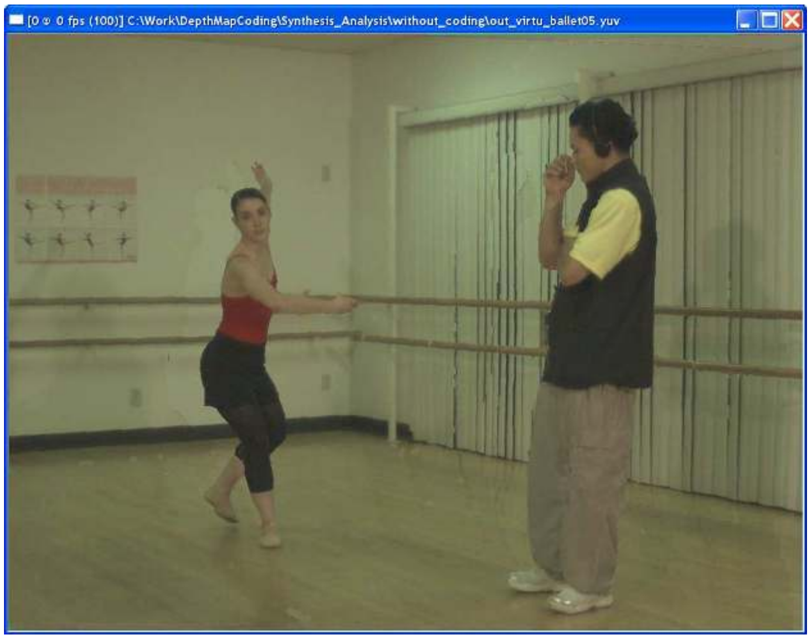

[0 ② 0 fps (100)] C:\Work\DepthMapCoding\Synthesis_Analysis\without_coding\out_virtu_ballet05.yuv

Figure 2.5: Subjective quality comparison of Ballet sequence; Synthesized 5th view without coding reference views. This results in 33.2 dB (Y PSNR).

coding distortion. The difference image Fig. 2.7 clearly shows that large distortion occurred along the object boundary, and it can be noticed that the foreground object boundary is eroded by background area. Therefore, we call this an *erosion artifact.*

2.3 Erosion artifact

Fig. 2.8 illustrates how erosion artifacts occur. When a depth map is compressed, distortion occurs due to transform and quantization. Since the boundary area contains more energy in high frequency components, it is more likely to be damaged by compression. In the distorted depth map, along the object boundary there will be both eroded and dilated regions. In an eroded region pixels corresponding to the

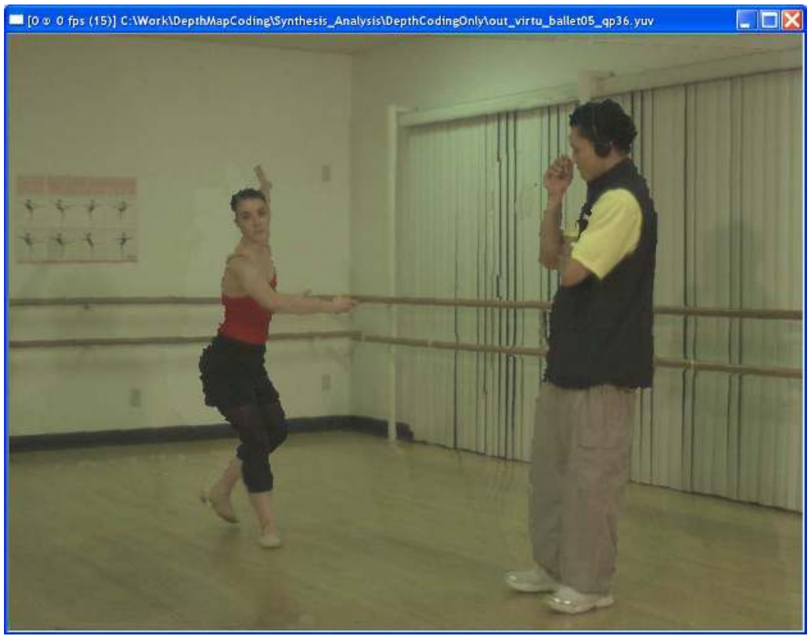

[0 ⊘ 0 fps (15)] C:\Work\DepthMapCoding\Synthesis_Analysis\DepthCodingOnly\out_virtu_ballet05_qp36.yuv

Figure 2.6: Subjective quality comparison of Ballet sequence; Synthesized 5th view with coded depth maps (QP=36) and uncompressed reference views (Y PSNR 32.1 dB),

object will now have depth values associated to the background. Correspondingly, in the dilated region parts of the background, pixel data will be associated to foreground depth values. When rendering a view with this distorted depth map, the regions corresponding to foreground depth (i.e., the object minus the eroded regions) will be warped with larger displacement than the background region (for example if the background has “infinite” depth, there will be no displacement from view to view). Note that this leads to different artifacts. Since the eroded region no longer has foreground depth value it will not move with the foreground object. Therefore, the object that is displaced will itself be eroded in its boundary. Instead, note that the error is less visible for dilated regions. This is because in those dilated regions, background intensity data is associated to foreground depth and is

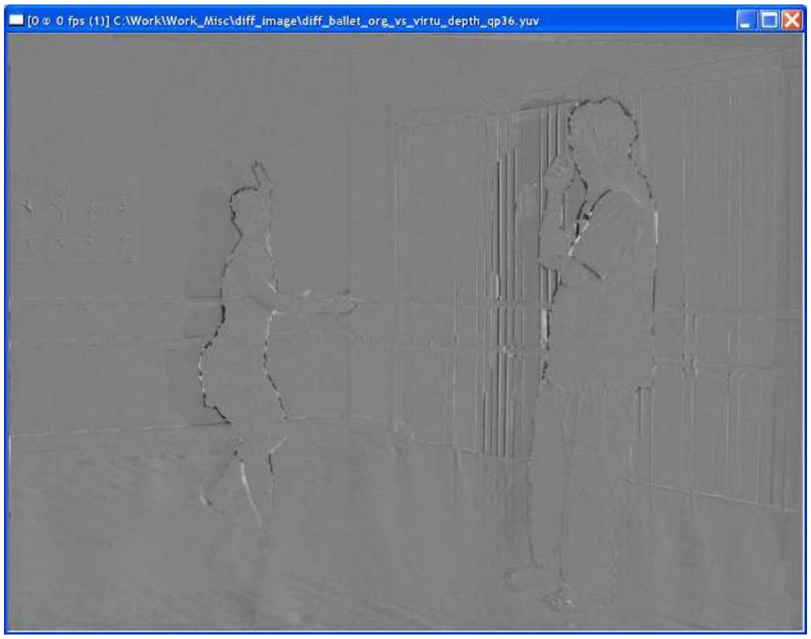

[0 @ 0 fps (1)] C:\Work\Work_Misc\diff_image\diff_ballet_org_vs_virtu_depth_qp36.yuv

Figure 2.7: Difference image between the synthesized views with and without depth map coding. 128 is added for visualization.

displaced along with the foreground object. This will not cause errors in the object shape, although it may mean (if the background is not uniform) that there will be a mismatch between the interpolated background and the true one. In summary, the foreground object boundary will suffer from erosion errors, which significantly degrades the subjective quality of the rendered view.

2.4 Characteristics of rendered view distortion due to depth map error

In the previous sections it is examined that depth map coding error can cause significant distortion in the rendered view. In this section we further analyze the distortion in the rendered view due to depth map coding error. Since the depth map

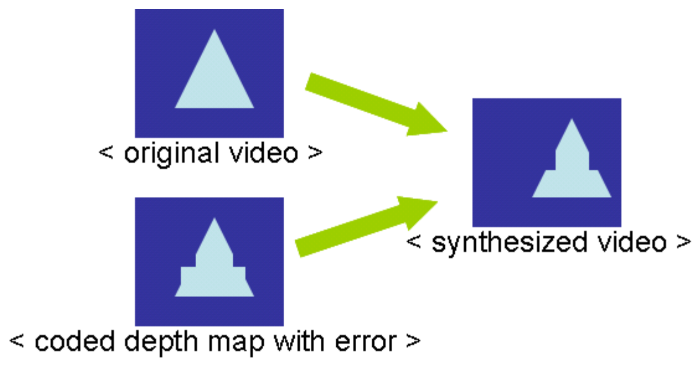


Figure 2.8: Occurrence of erosion artifact from depth map distortion. Depth map coding error generates eroded and dilated region along object boundary as can be seen in 'coded depth map with error'. This causes erosion artifact in 'synthesized video'.

provides geometry information to warp pixels in the reference view (reference pixels) to the target view in order to be synthesized, quantization error in depth map will cause the reference pixel to be mapped to the wrong place in the target view. The distortion in the rendered view from the incorrectly mapped reference pixel depends on local video characteristics. For example in a flat region the intensity value of the incorrect reference pixel is likely to be similar to the ground truth, while around object boundary it is less likely if different object has different intensity value. Therefore, larger distortion is expected in the object boundary area than in the flat region, which means that the distortion in the rendered view is localized.

In addition, the amount of distortion in the rendered view depends on various factors. First, the amount of geometry error depends on camera parameters such as baseline distance and focal length, then its effect on the rendered view distortion depends on local video characteristics such as amount of texture and contrast between neighboring pixels. Therefore, the rendered view distortion has non-linear feature.

Figs. 2.9 and 2.10 illustrate non-linear and localized characteristics, where the absolute difference between the synthesized views with and without depth map coding is represented as a temperature image. Even though quantization is applied uniformly throughout the frame, its effect on view interpolation is much more significant near edges. As the quantization step size Q increases, the geometry error increases and thus the rendered view distortion also increases. Note, however, that even though quantization error in depth map increases, the rendered view distortion still remains localized in a relatively small portion of pixels in the frame.

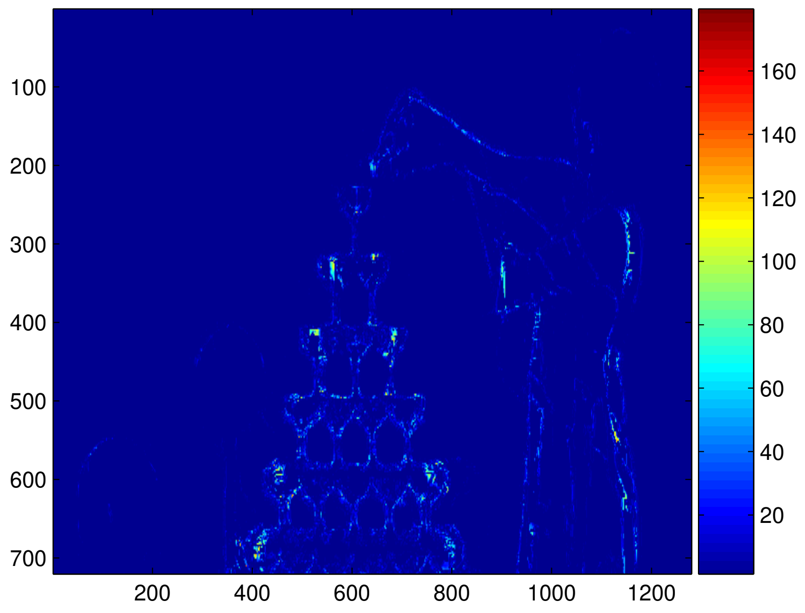
2.5 Quality measurement of rendered view

3-D video quality measurement can be different from that of 2-D video. However, some studies show that the quality of rendered view, which can be treated as 2-D image or video, significantly affect overall 3-D video quality. In this section we will discuss how to measure the quality of rendered view. Previous works are briefly reviewed and a method is proposed to efficiently measure the distortion in the rendered view that significantly degrades subjective quality.

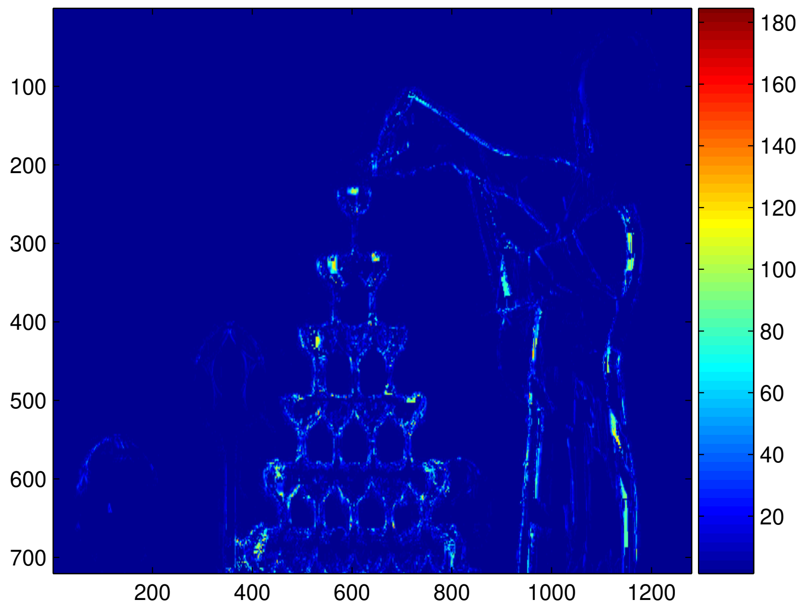
2.5.1 Review of previous work

We review some existing quality measurement methods for 2-D image or video first, and review how they can be extended to measure 3-D video quality.

To assess the quality of the general 2-D image or video, various quality measurement methods have been proposed to reflect the characteristics of the human visual system into objective quality [48, 68], such as NTIA General Video Quality Metric

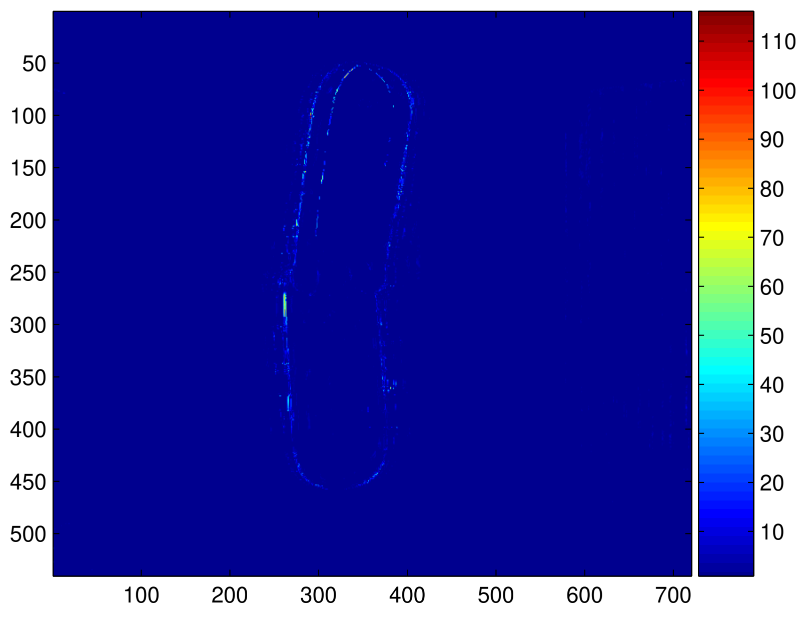


(a) Champagne Tower, QP = 24

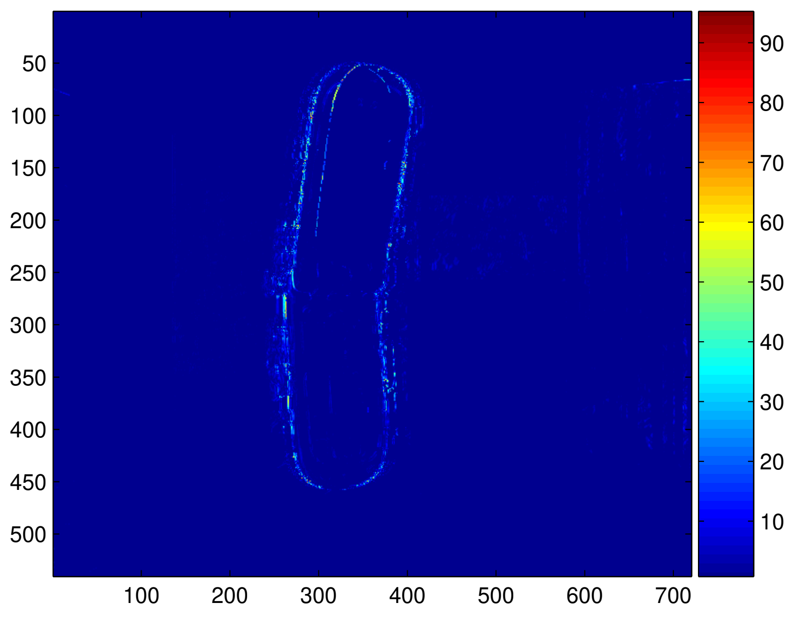


(b) Champagne Tower, QP = 36

Figure 2.9: Temperature images of the absolute difference between the synthesized view with and without depth map coding, Champagne Tower.



(a) Mobile, QP = 24



(b) Mobile, QP = 36

Figure 2.10: Temperature images of the absolute difference between the synthesized view with and without depth map coding, Mobile.

(VQM) [38], and structural similarity index (SSIM) [70] and multi-scale structural similarity index (MS-SSIM) [71], which are also adopted in the H.264/AVC reference software [22]. However, mean square error (MSE) and PSNR are widely used for objective quality metrics both in the academia and industry [56, 69]. They can be calculated as

$$\begin{aligned} \text{MSE} &= \frac{\sum_{i=1}^{N} |x_i - \tilde{x}_i|^2}{N}, \\ \text{PSNR} &= 10\log_{10}\frac{x_{\max}^2}{\text{MSE}} \text{ (dB)}, \end{aligned} \tag{2.2}$$

where x_i is the reference value, $\tilde{x}_i$ is the value to measure the distortion, N is the total number of pixels in a frame, and $x_{\max}$ is the peak signal value, e.g., 255 in 8 bit image or video.

There have been many studies on 3-D video quality measurement including rendered view quality evaluation. Goldmann *et al* [15] studied 3-D video acquisition procedure and subjective quality evaluation methodology, and Leon *et al* [25] evaluated subjective quality of 3-D video with varying depth map quality. Various 2-D image or video quality metrics are used to assess 3-D video quality. PSNR and MSE are used to evaluate the rendered view distortion from video and depth map compression [26, 31], and SSIM and VQM are applied to assess 3-D video quality [17, 60]. When the ground truth is available the distortion can be measured with respect to the ground truth [64]. But, when an arbitrary view is synthesized of which the ground truth is not available, the rendered view without any compression (no compression of video or depth map) can be used as the reference to measure the distortion due to compression [65].

These previous studies reveal that the rendered view quality has great impact on the overall 3-D video perceptual quality, and both video and depth map quality play important roles on rendered view quality. Therefore, to evaluate the performance of the proposed algorithms in the following chapters, we measure the 2-D rendered view quality. For simplicity, we mainly use PSNR of luminance component, which has been widely accepted as an effective objective quality metric.

2.5.2 Measuring localized distortion

In general video coding, quantization errors usually spread throughout a frame. However, as examined in Section 2.4, the rendered view distortion has non-linear and localized features, which causes significant subjective quality degradation. To improve overall subjective quality, it would be more important to reduce this localized distortion rather than reduce distortion all over the frame. Therefore, it would nice if the distortion in these area can be measured separately.

Spatial alignment in VQM cannot measure this correctly, since different local areas have different amount of translational error in the rendered view, and SSIM may not work, since the local area in the rendered view can have the same spatial structure as the ground truth but with translational error.

We propose a simple way to assess the localized rendered view distortion from depth map quantization error based on MSE or PSNR metric. Note that this can be simply extended to other existing methods such as VQM and SSIM. To distinguish the localized distortion from general distortion, we simply apply a thresholding technique. Since these localized distortion that can degrade subjective quality would have larger magnitude than that of distortion in other area, we only count the distortion larger than a preset threshold value.

There can be two ways to calculate PSNR only with distortion larger than the threshold value. First way is to calculate PSNR for a frame after making the distortion smaller than the threshold zero. We name this as the Noticeable-PSNR (N-PSNR) that is defined as

$$\begin{aligned} \text{N-MSE} &= \frac{\sum_{i=1}^{N} \alpha \cdot |x_i - \tilde{x}_i|^2}{N}. \\ \text{N-PSNR} &= 10\log_{10}\frac{x_{\max}^2}{\text{N-MSE}} \text{ (dB)}, \end{aligned} \tag{2.3}$$

where α is 0 when $|x_i - \tilde{x}_i|$ is less than or equal to the preset threshold value, Th, otherwise, α is 1. In (2.3) it can be noticed that N-MSE is divided by total number of pixels, N.

Second way is to calculate the PSNR only considering the local area of which the deviation is larger than the threshold value. We denote this as n-PSNR:

$$\begin{aligned} s &= \frac{\sum_{i=1}^{N} \alpha_i}{N}, \\ \text{n-MSE} &= \frac{\sum_{i=1}^{N} \alpha_i \cdot |x_i - \tilde{x}_i|^2}{s \cdot N}, \\ \text{n-PSNR} &= 10\log_{10}\frac{x_{\max}^2}{\text{n-MSE}} \text{ (dB)}, \end{aligned} \tag{2.4}$$

where s denotes the portion of pixels with deviation larger than the threshold.

While N-PSNR can be used by itself, n-PSNR only provide quality measurement of local area. Therefore, n-PSNR can be used together with s to measure the distortion in the rendered view. Then, each can convey different information. For example, if s is large, this means a large geometrical error occurred, and the rendered view distortion will increase proportionally. If n-PSNR is large, this implies

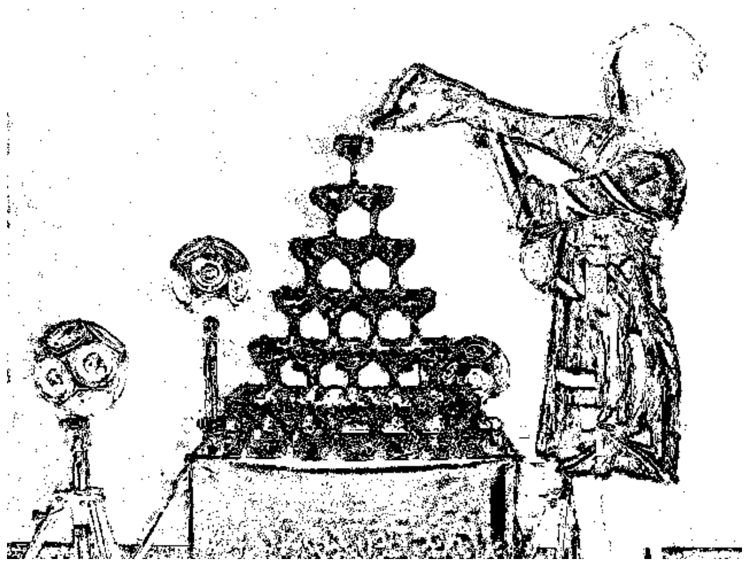
Table 2.1: PSNR and the proposed metric, N-PSNR, with threshold values of 5 and 10, generated by comparing the luminance (Y) component of the ground truth and the rendered view using various test sequences, where both video and depth map are compressed using H.264/AVC with QP set to 24 and 36.

Sequence	QP	PSNR (dB)	N-PSNR (dB)	
			Th=5	Th=10
Cafe	24	30.6	30.9	31.1
	36	30.2	30.6	30.9
Mobile	24	35.7	36.6	37.5
	36	31.9	32.3	33.4
Newspaper	24	28.4	28.6	29.5
	36	28.0	28.2	29.0
Balloons	24	33.5	34.3	35.1
	36	32.5	33.2	34.2
Champagne Tower	24	25.7	25.8	25.9
	36	25.9	26.1	26.2
Pantomime	24	35.2	35.9	36.6
	36	34.4	35.1	35.9

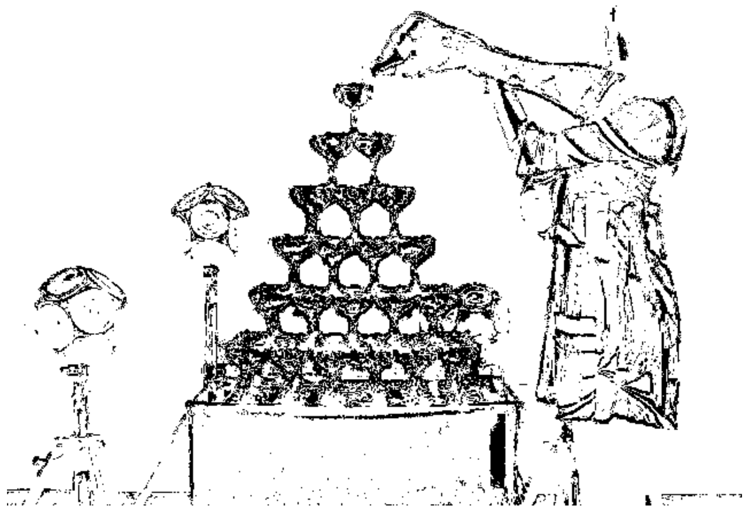
that large deviation occurred due to local video characteristics, e.g., high contrast across object boundaries, and this will also increase the rendered view distortion proportionally. N-PSNR will represent the amount of rendered view distortion as combination of s and n-PSNR.

Fig. 2.11 shows the area where the distortion is larger than the threshold value. It can be noticed that these area matches well with the temperature image in Fig. 2.9 in Section 2.4.

Table 2.1 shows the PSNR and the proposed metric, N-PSNR, and Table 2.2 shows s and n-PSNR, with threshold values of 5 and 10, generated by comparing the luminance (Y) component of the rendered view and the ground truth using various test sequences, where both video and depth map are compressed using H.264/AVC with QP set to 24 and 36.



(a) Th = 5



(b) Th = 10

Figure 2.11: Area where distortion is larger than a threshold value, Th, marked as black; Champagne Tower, synthesized with both video and depth map quantized using QP = 36.

Table 2.2: Measurement of noticeable distortion using s (%) and n-PSNR (dB), with threshold values of 5 and 10, generated by comparing the luminance (Y) component of the ground truth and the rendered view using various test sequences, where both video and depth map are compressed using H.264/AVC with QP set to 24 and 36.

Sequence	QP	Th=5		Th=10	
		s	n-PSNR	s	n-PSNR
Cafe	24	9.1	20.5	4.5	17.7
	36	12.7	21.6	5.1	18.0
Mobile	24	6.5	24.8	1.7	19.7
	36	21.8	25.7	8.0	22.4
Newspaper	24	40.9	24.7	13.4	20.7
	36	45.6	24.8	16.6	21.2
Balloons	24	10.4	24.5	3.3	20.3
	36	15.7	25.2	4.6	20.8
Champagne Tower	24	17.5	18.2	10.2	16.0
	36	19.7	19.0	10.9	16.6
Pantomime	24	7.6	24.8	3.4	22.0
	36	9.3	24.8	4.2	22.1

From Table 2.1 it can be noticed that different sequences have different N-PSNR value, which is combination of s and n-PSNR. This is because different sequences have different amount of area with noticeable distortion, s, and different magnitude of distortion reflected in n-PSNR, as can be noticed in Table 2.2. The size of the area would depend on camera settings and the magnitude of distortion would depend on local video characteristics. The relationship between the rendered view quality and various factors is analyzed in detail in Chapters 3 and 4.

2.6 Flickering artifact

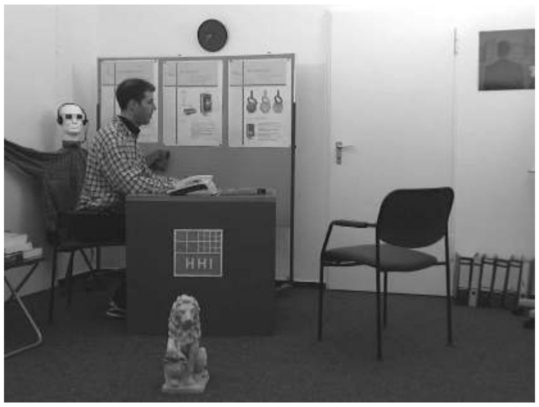
2.6.1 Flickering artifact due to temporal variation in depth map

In the case of video, if sensor noise is negligible, the distortion at the decoder is mainly due to quantization. In contrast, in the case of depth map estimated from video data (i.e., not captured directly with special devices such as range cameras), the estimated depth itself can be very noisy. For example, using stereo matching to obtain depth will lead to more significant errors in the boundaries of near objects, as compared to the background area. This is due to large differences in projection angles between left and right cameras for near objects, which leads to large occlusion. Moreover, for areas in the scene that are predominantly flat and contain limited amounts of texture, it will be difficult to find matching points between left and right views, which will make the depth information less reliable. In addition, if the depth maps are estimated on a frame by frame basis, i.e., depth/video information from other timestamps are not considered, unreliable estimates of depth are more likely to lead to stronger temporal variations, i.e., depth estimates may vary even when the "ground truth" does not.

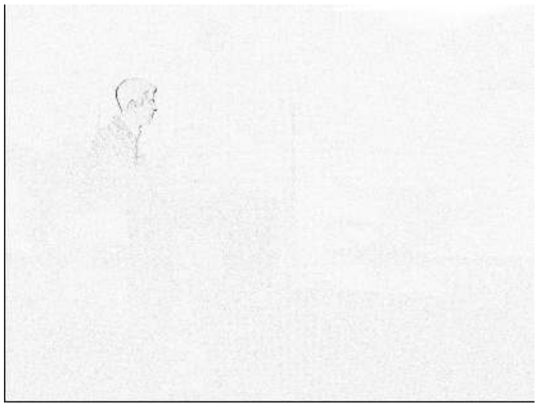
Fig. 2.12 helps illustrate these issues. From Fig. 2.12 (b) and (d) (where the absolute value of the temporal differences is scaled by 5 and inverted for easier visualization), it can be easily noticed that temporal variation in the depth map is very significant, even though there is practically no motion in this video. Most of these changes in the depth map can be attributed to errors in the stereo matching process. Note in particular that more errors can be observed around object boundaries and in the flat regions with less texture, where the stereo matching suffers due to occlusion and lack of matching features, respectively. This temporal variation in the depth map not only increases the coding bitrate but also deteriorates the subjective quality of the synthesized views by creating flickering artifacts in the flat region. However, as will be seen in Section 3.3, because these temporal variations in depth estimates do not correspond to changes in actual depth, efficient coding of depth can be achieved (e.g., by not coding many of these estimated depth changes), without significant impact on interpolated view quality. Even though it would be possible to improve depth map quality using more advanced systems such as range cameras, it will be still useful for algorithms to be robust to errors in depth map acquisition, which could be inherent to many acquisition systems.

2.6.2 Measurement of false temporal variation in depth map

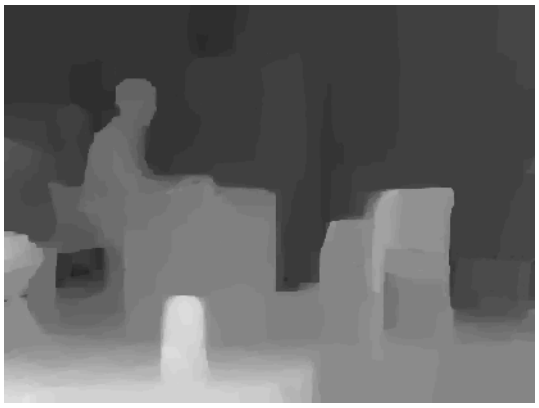
We now propose a method to measure the amount of false temporal variation in a depth map. The amount of temporal variation can be measured by taking the absolute difference between two consecutive frames in temporal domain. This will contain both true motion and false motion (noise). We assume that temporal



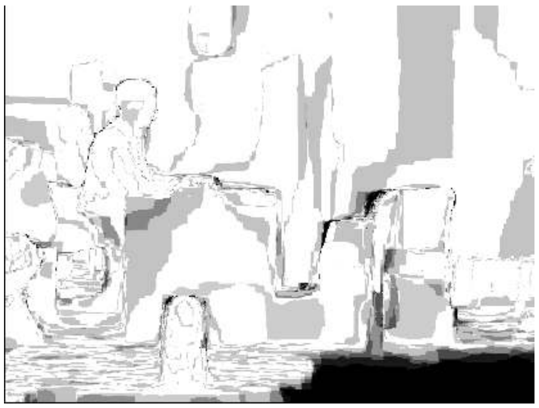
(a) Video frame



(b) Temporal difference



(c) Depth map



(d) Temporal difference

Figure 2.12: Example of temporal variation in depth map. (a) frame in the 'Door Flowers' video sequence, (b) difference between the first and second frame of the video, (c) corresponding depth map, and (d) difference between the depth maps of the first and second frames.

variation in video is mostly from true motion and noise is negligible. Then, one way to measure the false temporal variation in depth map would be subtracting the temporal variation in video from the temporal variation in depth map, i.e.,

$$\begin{aligned}\tau_{\text{DM,false}} &= \tau_{\text{DM,total}} - \tau_{\text{video,total}} \\ &= \tau_{\text{DM,false}} + \tau_{\text{DM,true}} - \tau_{\text{video,true}}, \end{aligned} \tag{2.5}$$

where τ_{DM} and τ_{video} denote average temporal variation in depth map and video, respectively.

However, since the variation ranges of depth map and video generally disagree, (2.5) cannot provide a precise measurement. Instead, we can define the area of true motion using video. For example, the pixels where the temporal difference is larger than a threshold value can be set as the true motion area. Then, the temporal variation in depth map excluding the true motion area can be averaged and used as the amount of false temporal variation in depth map. The threshold value will reflect the sensitivity of the true motion in video to noise. Below is the pseudo code to calculate the amount of false temporal variation in depth map.

Table 2.3 lists the amount of false temporal variation generated using various MPEG multiview test sequences according to the algorithm described above. 15 frames in one view is used for each sequence. Note that τ_{video}, τ_{DM}, and $\tau_{\text{DM,false}}$ are per-pixel absolute temporal variation on average. $\tau_{\text{DM,false}}$ is calculated using three threshold values of 0, 1, and 2. It can be noticed that the portion of false temporal variation area increases as the threshold increases, since larger threshold will allow less number of pixels to be included in the true motion area. However, it can be

Algorithm 1 Pseudo-code to calculate the amount of false temporal variation in depth map. $x_{i,n}$ and $d_{i,n}$ are i-th pixel in n-th frame of video and depth map, respectively, ϵ is the threshold to determine true motion in video, and m is number of pixels in false motion area.

```
$\tau_{\text{DM,false}} \leftarrow 0$
$m \leftarrow 0$
loop
  if $|x_{i,n} - x_{i,n+1}| \leq \epsilon$ then
    $\tau_{\text{DM,false}} \leftarrow \tau_{\text{DM,false}} + |d_{i,n} - d_{i,n+1}|$
    $m \leftarrow m + 1$
  end if
end loop
$\tau_{\text{DM,false}} \leftarrow \tau_{\text{DM,false}}/m$
```

Table 2.3: List of temporal variation in video, $\tau_{\text{video,total}}$, temporal variation in depth map, $\tau_{\text{DM,total}}$, and false temporal variation in depth map, $\tau_{\text{DM,false}}$ generated using threshold value, $\epsilon = 0, 1, 2$, where b represents the portion of false temporal variation area in percentage (%).

Sequence	τ_{video}	τ_{DM}	$\epsilon = 0$		$\epsilon = 1$		$\epsilon = 2$	
			b	$\tau_{\text{DM,false}}$	b	$\tau_{\text{DM,false}}$	b	$\tau_{\text{DM,false}}$
Balloons	1.6	0.7	23	0.7	61	0.7	82	0.7
Beergarden	3.7	0.7	62	0.1	69	0.1	74	0.2
Book Arrival	2.8	0.7	17	0.6	47	0.5	68	0.5
Cafe	2.9	1.2	23	0.6	58	0.6	80	0.6
Car Park	2.2	2.8	16	2.7	44	2.7	67	2.7
Champagne Tower	1.4	0.2	27	0.2	66	0.2	86	0.2
Hall1	2.1	1.6	17	1.6	48	1.6	71	1.6
Hall2	3.4	3.5	15	3.2	42	3.2	63	3.2
Kendo	4.1	3.4	20	3.0	51	3.1	70	3.2
Lovebird 1	0.3	1.2	92	1.0	93	1.0	98	1.1
Mobile	3.2	4.7	61	0.1	86	0.1	90	0.2
Newspaper	1.3	0.4	29	0.3	69	0.4	88	0.4
Pantomime	4.8	2.9	24	2.8	61	2.8	79	2.8
Street	2.3	1.5	16	1.4	44	1.4	65	1.4

noticed that the false temporal variation, $\tau_{\text{DM,false}}$, is not sensitive to the threshold value.

When the false temporal variation in depth map, $\tau_{\text{DM,false}}$, is used to represent the overall amount of flickering artifact, not only per-pixel false temporal variation, $\tau_{\text{DM,false}}$, but also its portion, b, should be considered. In addition, according to the depth range covered by depth map, the amount of flickering artifact due to depth map can be different. For example, if a scene includes from near object to very far object, e.g. as would happen in an outdoor scene, depth map will cover wide range of depth, and temporal variation in depth map will also imply large variation in actual depth value. On the other hand, in case of indoor scene where depth range is restricted, temporal variation in depth map will imply less variation in actual depth value than that of the outdoor scene case. Considering the relationship between the actual depth value and the depth map value given in (2.1), a scaling factor, η, can be set as

$$\eta = \left(\frac{1}{Z_{\text{near}}} - \frac{1}{Z_{\text{far}}} \right) \cdot f, \tag{2.6}$$

where the focal length divided by the effective pixel size, f, is multiplied. Wide range of depth will result in large scale factor, and accordingly more flickering artifacts. Then, the amount of flickering artifact due to false temporal variation in depth map, F, can be represented as

$$F = \tau_{\text{DM,false}} \cdot b \cdot \eta. \tag{2.7}$$

Table 2.4 lists the amount of flickering artifact due to false temporal variation in depth map, F, and other values to calculate this such as b, η, and $\tau_{\text{DM,false}}$ with threshold value ϵ set to 1.

We will use the proposed measurement of false temporal variation in depth map and the resulting level of flickering in Section 3.3 to analyze test sequences

Table 2.4: List of per-pixel false temporal variation, $\tau_{\text{DM,false}}$, calculated with $\epsilon = 1$, b, η, and F.

Sequence	$\tau_{\text{DM,false}}$	b	η	F
Balloons	0.7	61	4.8	2.0
Beergarden	0.1	69	66.7	6.8
Book Arrival	0.5	47	34.8	8.8
Cafe	0.6	58	0.9	0.3
Car Park	2.7	44	33.9	40.3
Champagne Tower	0.2	66	1.1	0.2
Hall1	1.6	48	33.9	26.0
Hall2	3.2	42	33.9	46.4
Kendo	3.1	51	4.8	7.6
Lovebird 1	1.0	93	0.9	0.8
Mobile	0.1	86	24.9	2.6
Newspaper	0.4	69	0.8	0.2
Pantomime	2.8	61	0.4	0.7
Street	1.4	44	33.9	20.7

and discuss the performance of the proposed coding tool to reduce the flickering artifact.

2.7 Conclusion

In this chapter distortion in the rendered view is analyzed, and it is found that depth map error can cause significant distortion in the rendered view. Specifically, depth map quantization error can cause distortion along the object boundary in the rendered view, which has non-linear and localized characteristics. Based on this observation a noticeable distortion measurement method in the rendered view is proposed. In addition, depth map estimation error can cause flickering artifact, which reduces subjective quality of rendered video. It is also proposed to measure the amount of false temporal variation in depth map. From these analyses it can be noticed that the distortion in the depth map would be very different from the distortion incurred in the rendered view. Therefore, when a depth map is coded, it will be desirable to consider the rendered view distortion from depth map error rather than the depth map distortion itself. It will be also possible to think about an efficient way to compensate the localized distortion by wisely spending bits to provide additional information in those areas. In the following chapters, we propose the depth map coding tools considering the rendered view distortion, and the new data format which can efficiently compensate the localized distortion.

Chapter 3

Depth Map Coding Tools Using New Distortion Metric

As analyzed in the previous chapter, depth map errors can cause significant distortion in the rendered view. In standard video compression, quantization errors directly affect the rendered view quality by adding noise to the luminance or chrominance level of each pixel. In contrast, the distortion in the depth map affects indirectly the rendered video quality: the depth map error leads to a geometric error in the interpolation, which in turn is translated into errors in the luminance or chrominance of the rendered view. In this chapter, first, the geometry error from the depth map quantization error is derived, and the rendered view distortion due to the geometry error is estimated. Then, this new distortion metric is applied to an RD optimized mode selection scheme to improve the coding efficiency by considering the rendered view quality. A new skip mode selection scheme is also proposed to reduce the flickering artifact. In addition, a new transform using graph representation is applied to depth map coding to achieve better coding efficiency.

3.1 Effect of depth map distortion on rendered view

In this section we derive the relationship between the depth map error and geometry error using the global camera parameters, then propose a method to estimate the rendered view distortion by taking into account local video characteristics. We also show that computational complexity reduction is achieved by using an autoregressive model to model the video signal.

3.1.1 Derivation of geometry error from depth map distortion

When a depth map L is encoded using lossy compression, the resulting distortion in the decoded map causes geometry errors when it is used for view interpolation. The geometry error due to the depth map distortion can be calculated using intrinsic and extrinsic camera parameters. Table 3.1 lists the symbol notations used in the equations hereinafter.

A camera coordinate $(x, y, z)^{\mathrm{T}}$ can be obtained from the world coordinate $(X, Y, Z)^{\mathrm{T}}$ as

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{AM} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}, \tag{3.1}$$

where $\mathbf{A}$ and $\mathbf{M}$ are respectively the intrinsic and extrinsic camera matrices, where $\mathbf{M}$ consists of a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$ [61]. The image coordinate $(x_{\mathrm{im}}, y_{\mathrm{im}})^{\mathrm{T}}$ can be expressed from the camera coordinate $(x, y, z)^{\mathrm{T}}$ as

Table 3.1: Table of notations.

Symbol	Explanation
(x, y, z) (x', y', z')	camera coordinates
(X, Y, Z)	world coordinates
$(x_{\text{im}}, y_{\text{im}})$ $(x'_{\text{im}}, y'_{\text{im}})$	image coordinates
$\mathbf{A}$	intrinsic camera matrix
$\mathbf{M}$	extrinsic camera matrix
$\mathbf{R}$	rotation matrix
$\mathbf{t}$	translation vector
p, p'	view indices
$Z_p(x_{\text{im}}, y_{\text{im}})$	depth value at $(x_{\text{im}}, y_{\text{im}})$ in p-th view
$L_p(x_{\text{im}}, y_{\text{im}})$	depth map value at $(x_{\text{im}}, y_{\text{im}})$ in p-th view
Z_{near}	the nearest depth value in the scene
Z_{far}	the farthest depth value in the scene
$\Delta\mathbf{P}(x_{\text{im}}, y_{\text{im}})$	translational geometry error at $(x_{\text{im}}, y_{\text{im}})$
δ_x, δ_y	translations in horizontal/vertical directions
k_x, k_y	scaling factors relating depth map error to horizontal/vertical translations
(o_x, o_y)	the coordinates in pixel of the image center (the principal point)
f_x, f_y	focal length divided by the effective pixel size in horizontal/vertical direction
Δt_x	camera baseline distance
$\mathbf{X}_n$	a vector of video pixels with index n
N	size of vector or number of pixels in a vector
T	vector transpose operator
ρ	correlation coefficient
cov( )	covariance
σ and σ^2	standard deviation and variance
J and λ	Lagrangian cost and Lagrange multiplier
$\tilde{D}$	estimated distortion
D_{depth} and R_{depth}	depth map distortion and bitrate
w	weight applied to a view
Q	quantization step size

$$\begin{pmatrix} x_{\text{im}} \\ y_{\text{im}} \end{pmatrix} = \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix}. \tag{3.2}$$

Therefore, if we know the per pixel depth value, z, we can map a pixel position into a point in the world coordinate, and that position can be remapped into another camera coordinate that belongs to the view to be rendered. Lai *et al* [24] derived the geometry error from depth map distortion in the parallel camera case. In this section, we propose a more precise representation of geometry error as a function of distortion in the depth map under the assumption of approximately parallel cameras, so that the depth values in different camera coordinates, z and z', will be approximately equal to the depth values in the world coordinate, Z, i.e., $z \approx z' \approx Z$, without translation factor in z-axis. First, a camera coordinate in the p-th view can be mapped into a camera coordinate in the p'-th view using the camera intrinsic parameters $\mathbf{A}_p$ and $\mathbf{A}_{p'}$, and extrinsic parameters $\mathbf{R}_p$, $\mathbf{R}_{p'}$, $\mathbf{t}_p$, and $\mathbf{t}_{p'}$ of both cameras as

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \mathbf{A}_{p'}\mathbf{R}_{p'} \left\{ \mathbf{R}_p^{-1}\mathbf{A}_p^{-1} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \mathbf{t}_p - \mathbf{t}_{p'} \right\}. \tag{3.3}$$

Hence, the image coordinate in the p'-th view with the assumption of no translation in z-axis, i.e. $z = z'$, can be computed as:

$$\begin{pmatrix} x'_{\text{im}} \\ y'_{\text{im}} \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{x'}{z'} \\ \frac{y'}{z'} \\ \frac{z'}{z'} \end{pmatrix} = \mathbf{A}_{p'}\mathbf{R}_{p'}\mathbf{R}_p^{-1}\mathbf{A}_p^{-1} \begin{pmatrix} x_{\text{im}} \\ y_{\text{im}} \\ 1 \end{pmatrix} + \frac{1}{z'}\mathbf{A}_{p'}\mathbf{R}_{p'} \left\{ \mathbf{t}_p - \mathbf{t}_{p'} \right\}. \tag{3.4}$$

When a depth map is quantized, this causes depth error, Δz, and the corresponding camera coordinate is

$$\begin{pmatrix} x + \Delta x \\ y + \Delta y \\ z + \Delta z \end{pmatrix} = \begin{pmatrix} x_{\mathrm{im}} \\ y_{\mathrm{im}} \\ 1 \end{pmatrix} (z + \Delta z). \tag{3.5}$$

If this is mapped to the p'-th view, the camera coordinates becomes

$$\begin{pmatrix} x' + \Delta x' \\ y' + \Delta y' \\ z' + \Delta z' \end{pmatrix} = \mathbf{A}_{p'} \mathbf{R}_{p'} \left\{ \mathbf{R}_p^{-1} \mathbf{A}_p^{-1} \begin{pmatrix} x + \Delta x \\ y + \Delta y \\ z + \Delta z \end{pmatrix} + \mathbf{t}_p - \mathbf{t}_{p'} \right\}, \tag{3.6}$$

and the corresponding image coordinates are

$$\begin{aligned} \begin{pmatrix} x'_{\mathrm{im}} + \Delta x'_{\mathrm{im}} \\ y'_{\mathrm{im}} + \Delta y'_{\mathrm{im}} \\ 1 \end{pmatrix} &= \begin{pmatrix} \frac{x' + \Delta x'}{z' + \Delta z'} \\ \frac{y' + \Delta y'}{z' + \Delta z'} \\ \frac{z' + \Delta z'}{z' + \Delta z'} \end{pmatrix} \\ &= \mathbf{A}_{p'} \mathbf{R}_{p'} \mathbf{R}_p^{-1} \mathbf{A}_p^{-1} \begin{pmatrix} x_{\mathrm{im}} \\ y_{\mathrm{im}} \\ 1 \end{pmatrix} + \frac{1}{z' + \Delta z'} \mathbf{A}_{p'} \mathbf{R}_{p'} \left\{ \mathbf{t}_p - \mathbf{t}_{p'} \right\}. \end{aligned} \tag{3.7}$$

The relationship between the actual depth value, z, and 8-bit depth map value, $L_p(x_{\mathrm{im}}, y_{\mathrm{im}})$, is given in (2.1). Then the position error due to a depth map error in the p-th view, ΔL_p, can be calculated by subtracting (3.4) from (3.7) as follows:

$$\begin{pmatrix} \Delta x_{\mathrm{im}} \\ \Delta y_{\mathrm{im}} \\ 1 \end{pmatrix} = \left(\frac{1}{z' - \Delta z'} - \frac{1}{z'} \right) \mathbf{A}_{p'} \mathbf{R}_{p'} \left\{ \mathbf{t}_p - \mathbf{t}_{p'} \right\}$$

$$= \frac{\Delta L_p \left(x_{\mathrm{im}}, y_{\mathrm{im}} \right)}{255} \left(\frac{1}{Z_{\mathrm{near}}} - \frac{1}{Z_{\mathrm{far}}} \right) \mathbf{A}_{p'} \mathbf{R}_{p'} \left\{ \mathbf{t}_p - \mathbf{t}_{p'} \right\}. \qquad (3.8)$$

This reveals that there is a linear relationship between the depth map distortion ΔL, and the translational rendering position error $\Delta \mathbf{P}$ in the rendered view as

$$\Delta \mathbf{P} \left(x_{\mathrm{im}}, y_{\mathrm{im}} \right) = \begin{pmatrix} \delta_x \left(x_{\mathrm{im}}, y_{\mathrm{im}} \right) \\ \delta_y \left(x_{\mathrm{im}}, y_{\mathrm{im}} \right) \end{pmatrix} = \Delta L_p \left(x_{\mathrm{im}}, y_{\mathrm{im}} \right) \begin{pmatrix} k_x \\ k_y \end{pmatrix}, \qquad (3.9)$$

where $\delta_x \left(x_{\mathrm{im}}, y_{\mathrm{im}} \right)$ and $\delta_y \left(x_{\mathrm{im}}, y_{\mathrm{im}} \right)$ are the resulting geometry error in horizontal and vertical directions at $\left(x_{\mathrm{im}}, y_{\mathrm{im}} \right)$, respectively, from the depth map distortion ΔL at the camera position p, and k_x and k_y are the scale factors determined from the camera parameters and the depth ranges as shown in (3.8).

When the cameras are in parallel positions, further simplification can be made, as there will be no translation other than horizontal direction. In this case, there will be a difference only in horizontal direction between the translation vectors. In addition, the rotation matrix in (3.7) becomes an identity matrix. Neglecting radial distortion in the camera intrinsic matrix, the scaling factor, k_x, in (3.9) becomes

$$k_x = \frac{1}{255} \left(\frac{1}{Z_{\mathrm{near}}} - \frac{1}{Z_{\mathrm{far}}} \right) f_x \Delta t_x, \qquad (3.10)$$

where f_x is the focal length divided by the effective pixel size in horizontal direction, and Δt_x is the camera baseline distance.

Since the scale factors depend on the camera settings and depth ranges, the same amount of depth map distortion can cause different amount of geometry error. For example, if the distance between cameras, $|\mathbf{t}_p - \mathbf{t}_{p'}|$ is large, or the camera captures a near object so that $\frac{1}{Z_{\text{near}}}$ becomes large, then the geometry error will increase as k_x and k_y will be large. This indicates that a dense camera setting and a farther away scene composition will tend to be more robust to depth coding distortion in the rendering process.

In addition, the effect of geometry error on the quality of rendered view will depend on local characteristics of the video. For example, in areas of a video frame with complex textures and objects, the distortion caused by the geometry error will be significant, as different positions should have quite different pixel values. On the other hand, in the areas with simple textures or flat (homogeneous) regions, the amount of the distortion due to geometry error will be small since pixels at different positions are similar. Therefore, it is necessary to link the geometry error to the rendered view distortion according to the local video characteristics, which will be studied in the next subsections.

3.1.2 Estimation of distortion using reference video

In a DIBR system, a view can be rendered using a set of reference video frames and their corresponding depth maps. The exact amount of distortion in the rendered view can be measured if we compare the rendered view with the ground truth, i.e., the captured video by a camera at that position. However, the ground truth may not be available since the rendered view can be generated for any arbitrary viewpoint. It would also be too much computationally complex if the actual view rendering process have to be performed during the depth map coding process.

Instead, we propose to estimate the rendered view distortion using the reference video frame. The reference video is the captured and encoded video at the encoder side, transmitted to the decoder along with the depth map, and can be used to synthesize other views. Therefore, the reference video belonging to the same viewpoint as the depth map is always available. As shown in the previous subsection, when a depth map value is compressed with the quantization error, ΔL, this causes geometry error, $\Delta\mathbf{P}$. Therefore, considering a depth map value at $(x_{\text{im}}, y_{\text{im}})$, the video pixel value at the same location, $(x_{\text{im}}, y_{\text{im}})$, can be compared to the video pixel value translated by the geometry error, i.e. the one at $(x_{\text{im}} + \delta_x, y_{\text{im}} + \delta_y)$. Fig. 3.1 illustrates this. Note that the two dimensional location index, $(x_{\text{im}}, y_{\text{im}})$, is replaced with the one dimensional index, i, for simplicity.

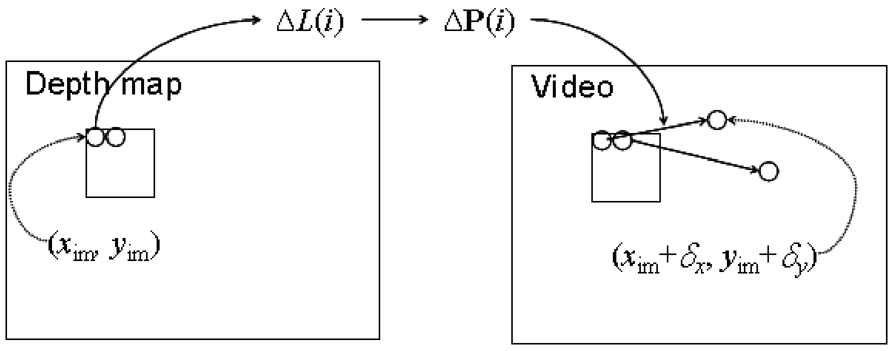

$\Delta L(i)$
$\Delta\mathbf{P}(i)$
Depth map
Video
$(x_{\text{im}}, y_{\text{im}})$
$(x_{\text{im}}+\delta_x, y_{\text{im}}+\delta_y)$

Figure 3.1: Illustration of rendered view distortion estimation process using reference video.

When encoding a particular block of the depth map, the distortion of the rendered view corresponding that block can be approximated using the reference video frame by calculating the sum of squared error (SSE) between the block in the video frame collocated with the depth map block to be coded, $\mathbf{X}_n = [x_i, x_{i+1}, \cdots, x_{i+N}]^{\mathrm{T}}$, and the block $\mathbf{X}_m = [x_{i+\Delta\mathbf{P}(i)}, x_{i+1+\Delta\mathbf{P}(i+1)}, \cdots, x_{i+N+\Delta\mathbf{P}(i+N)}]^{\mathrm{T}}$, formed by translating each pixel in $\mathbf{X}_n$ by the corresponding geometry error vector $\Delta\mathbf{P}$ calculated in Section 3.1.1, i.e.:

$$\mathrm{SSE} = (\mathbf{X}_n - \mathbf{X}_m)^{\mathrm{T}} (\mathbf{X}_n - \mathbf{X}_m) = \mathbf{X}_n^{\mathrm{T}}\mathbf{X}_n - 2\mathbf{X}_n^{\mathrm{T}}\mathbf{X}_m + \mathbf{X}_m^{\mathrm{T}}\mathbf{X}_m, \tag{3.11}$$

where the block in a video frame is represented as a column vector that consists of pixels in the block. Note that each pixel x_i in the video block can have a different translation error $\Delta\mathbf{P}(i)$, since the translation error for a given pixel depends on the error in the corresponding position in the depth map.

The assumption underpinning this approach is that the local video characteristics of the synthesized video would be very similar to those of the reference video, so that the distortion due to the geometry error can be estimated using the reference video. Therefore, a precise estimation of the rendered view distortion due to depth map error can be achieved using both global camera parameters and local video characteristics.

3.1.3 Estimation of distortion using autoregressive model

The distortion estimation method described in the previous subsection is conceptually straightforward, but it requires accessing the video frame pixel by pixel. Because each pixel in the block $\mathbf{X}_n$ can have different amount of geometry error $\Delta\mathbf{P}(i)$, the corresponding pixels after displacement (i.e., $\mathbf{X}_m$) are not likely to be located in consecutive memory positions. Access to non-consecutive memory locations can be expensive in many practical implementations. Let us take an example of a random access memory to store video pixels, which can access 64 bits or 8 pixels with one addressing operation with 5 cycles and next successive 64 bits with 1 cycle with a burst mode access. For 16×16 pixels, a total $16 \times 16 \times 5 = 1280$ cycles will be consumed when each pixel location is not successive, while only $16 \times (5+1) = 96$ cycles will be required when pixels are in successive location for each row, a factor of 13 difference in memory access time. In this section we propose a simpler method to estimate the distortion in the rendered view, which does not require accessing non-consecutive memory locations.

The problem is to estimate the distortion due to displacement caused by geometry error. Instead of accessing video pixels in random location as in the previous subsection, the difference between the video pixels with and without the displacement can be estimated by calculating the correlation of spatially neighboring video pixels and modeling video signal using autoregressive model. Under stationarity and zero mean assumptions for the video signal, the correlation coefficient ρ between $\mathbf{X}_n$ and $\mathbf{X}_m$ can be expressed as:

$$\rho = \frac{\mathrm{cov}\left(\mathbf{X}_n, \mathbf{X}_m\right)}{\sigma_{\mathbf{X}_n}\sigma_{\mathbf{X}_m}} = \frac{\mathrm{cov}\left(\mathbf{X}_n, \mathbf{X}_m\right)}{\sigma^2_{\mathbf{X}_n}} = \frac{\mathbf{X}_n^{\mathrm{T}}\mathbf{X}_m}{\mathbf{X}_n^{\mathrm{T}}\mathbf{X}_n}. \tag{3.12}$$

Note that the zero mean condition, which can be achieved by removing the mean from the video signal, will not change the resulting SSE. Therefore from (3.11) and (3.12), the SSE can be expressed in terms of the correlation coefficient and the variance of the video block as:

$$\begin{aligned} \mathrm{SSE} &= \left(N-1\right)\sigma^2_{\mathbf{X}_n} - 2\left(N-1\right)\rho\sigma^2_{\mathbf{X}_n} + \left(N-1\right)\sigma^2_{\mathbf{X}_m} \\ &= 2\left(N-1\right)\left(1-\rho\right)\sigma^2_{\mathbf{X}_n}, \end{aligned} \tag{3.13}$$

where N represents the number of pixels in the block. Then, (3.13) can be approximated using the first order autoregressive model for the correlation coefficient as

$$\text{SSE} \approx 2\,(N-1)\left(1 - \frac{1}{N}\sum_{i=1}^{N} \rho_1^{|\Delta \mathbf{P}(n)|}\right)\sigma_{\mathbf{X}_n}^2, \qquad (3.14)$$

where ρ_1 represents the video correlation when translated by one pixel.

Note that in (3.14), based on the autoregressive model, the correlation decreases as the geometry error increases. Local video characteristics are captured by the correlation coefficient. Thus, it is expected that the distortion will be greater if there is little spatial correlation in the video frame. This makes sense, since the area with complex texture or object boundary area with high contrast between objects have little correlation, and are sensitive to geometry error.

The correlation coefficient, ρ_1, and the variance of the video block can be calculated first for a given block, which involves consecutive memory access, then the distortion in the rendered view can be estimated as in (3.14). The correlation coefficient and variance for each block can be also calculated and stored beforehand to ease the real time processing. But, the estimation will be less precise than with the first method, since the local video characteristics are represented block by block using the autoregressive model, while the first method accurately calculates the error for each pixel. Thus, the second method provides good compromise between estimation accuracy and computational complexity.

3.2 Rate-distortion optimization for depth map coding

In the previous section new distortion estimation methods are proposed to estimate the rendered view distortion from depth map quantization error. In this section, these new distortion metrics are applied to depth map coding to improve coding efficiency by using Lagrange optimization to select optimal coding mode. In addition a new Lagrange multiplier is derived using the new distortion metric based on the autoregressive model to improve coding efficiency.

3.2.1 Definition of rate and distortion

State of the art video codecs make use of various coding modes in order to improve coding efficiency. For example, H.264 provides various spatial prediction directions for intra prediction modes and various block sizes and temporal prediction directions for inter prediction modes. To achieve the best coding performance, it is important to select the optimal coding mode by considering bitrate and distortion at the same time. For this purpose Lagrangian optimization has been widely used [35, 55] to optimize video coding performance.

When Lagrangian techniques are applied to depth map coding, it is appropriate to consider the rendered view distortion caused by depth map error rather than the distortion in the depth map itself, since the depth map is not displayed and is only used in order to help view synthesis. Therefore, instead of using the depth map distortion directly, we propose to use the estimation of the distortion in the rendered view to select the optimal coding mode for depth map compression. The estimation of the rendered view distortion can be performed as described in Section 3.1, where the rendered view distortion is estimated from the depth map error using global camera parameters and local video characteristics. In this subsection we discuss how the new distortion metric can be applied to optimize depth map coding.

Let us start from a simple example. Assume that we wish to synthesize a view using two reference frames, i.e. left and right coded views (video and depth map). Though the rendered view distortion can be affected by various factors such as noise in the reference video and occlusion artifacts (see Section 2.2), we want to consider exclusively the distortion from the depth map quantization error, because what we want to optimize here is the depth map coding. Similarly, we only consider the bitrate to code the depth map. Then, the Lagrangian cost for depth map coding can be computed using the distortion occurred in the rendered view due to depth map error and the sum of the bitrates for two depth maps as:

$$J = \tilde{D}\left(D_{\text{depth,left}}, D_{\text{depth,right}}\right) + \lambda R_{\text{depth,left+right}}, \tag{3.15}$$

where the estimated distortion in the rendered view, $\tilde{D}$, is expressed as a function of the depth map distortion, D_{depth} for both left and right images, and $R_{\text{depth,left+right}}$ denotes the total bitrate to code left and right depth maps. In general, it is not straightforward to estimate the rendered view distortion from two reference frames, since the view synthesis procedure involves non-linear operations. For example, for the occluded regions, only one reference frame may be used for view synthesis, while other areas can be synthesized by a blending process such as weighted averaging using both reference frames. Instead, when we code a single depth map, it is easier to model the rendered view distortion caused by quantization error of the single depth map. Thus, we propose to linearize the rendered view distortion caused from each depth map, so that the distortion in the rendered view can be represented as a weighted sum of the distortion caused by each depth map as

$$\tilde{D}\left(D_{\text{depth,left}}, D_{\text{depth,right}}\right) = w\tilde{D}\left(D_{\text{depth,left}}\right) + \left(1-w\right)\tilde{D}\left(D_{\text{depth,right}}\right), \tag{3.16}$$

where w denotes the weight applied to a view, $w \in [0,1]$. The weight can be determined based on the distance from the reference view to the target view. For example, if the target view is located at the middle point between the left and the right reference views, we can choose $w = 0.5$. In addition, we assume that the two depth maps are coded independently from each other without inter-view prediction, so that

$$R_{\text{depth,left+right}} = R_{\text{depth,left}} + R_{\text{depth,right}}. \tag{3.17}$$

By putting (3.16) and (3.17) into (3.15), the Lagrangian cost can be written as:

$$J = w\tilde{D}\left(D_{\text{depth,left}}\right) + \left(1-w\right)\tilde{D}\left(D_{\text{depth,right}}\right) + \lambda\left(R_{\text{depth,left}} + R_{\text{depth,right}}\right). \tag{3.18}$$

Finally, (3.18) can be separated for each view as

$$J = w\tilde{D}\left(D_{\text{depth}}\right) + \lambda R_{\text{depth}}. \tag{3.19}$$

Now, the estimated rendered view distortion caused by single depth map coding, $\tilde{D}\left(D_{\text{depth}}\right)$, can be represented using either (3.11) or (3.14).

This can be generalized when M views are synthesized using N coded views as

$$\begin{aligned} J &= \sum_{j=1}^{M} \tilde{D}_j \left(D_{\text{depth},1}, D_{\text{depth},2}, \cdots, D_{\text{depth},N} \right) + \lambda R_{\text{depth,total}} \\ &= \sum_{j=1}^{M} \sum_{i=1}^{N} w_i \cdot \tilde{D}_j \left(D_{\text{depth},i} \right) + \lambda \sum_{i=1}^{N} R_{\text{depth},i}, \end{aligned} \tag{3.20}$$

where w_i is the weight applied to each view indexed by i during the rendering process. However, in practice the weight is difficult to determine since one depth map can be used to synthesize multiple views at arbitrary locations. Therefore, when we code a depth map, a simplified expression as in (3.19) can be used with a fixed weight factor, e.g. one.

3.2.2 Derivation of Lagrange multiplier

The Lagrange multiplier can control the trade-off between the bitrate and distortion to achieve optimal coding performance. In case of video coding, the optimal Lagrange multiplier can be selected by taking derivative of distortion and bitrate of the compressed video [72, 73]. When this is applied to the depth map coding, it is necessary to consider the rendered view distortion instead of the depth map distortion itself. In this subsection, we propose a new Lagrange multiplier selection scheme using the estimated rendered view distortion with the autoregressive model.

If the distortion and the bitrate are expressed as a function of quantization step size, Q, λ can be calculated by taking the derivative of (3.19) and setting it to zero as

$$\lambda = -\frac{d\tilde{D}\left(Q\right)/dQ}{dR_{\text{depth}}\left(Q\right)/dQ}. \tag{3.21}$$

The weight in (3.19) is set to one. Note that D_{depth} is an absolute difference and can be represented as a function of quantization step size, Q, under the uniform distribution assumption as

$$D_{\text{depth}} = \frac{1}{Q}\int_{-\frac{Q}{2}}^{\frac{Q}{2}} |x|\, dx = \frac{Q}{4}. \tag{3.22}$$

However, different depth map sequences can have different statistical properties, which leads to different amounts of distortion due to quantization. This is because what is quantized is usually a transformed version of the prediction residual signal. For example, some depth maps can have large amount of temporal noise, which will generate large residual signals after inter prediction, and potentially significant quantization errors in turn. Alternatively, some depth maps may show little variation in depth, which will result in a small residual signal after intra prediction, and lower quantization errors. We capture this by modeling

$$D_{\text{depth}} = c_1 Q, \tag{3.23}$$

where c_1 is a scaling factor, which can be selected according to sequence characteristics. Then, this can be used to estimate the distortion as squared error in the rendered view. If the autoregressive model derived in (3.13) and (3.14) is used, this becomes:

$$\begin{aligned} \tilde{D}\left(Q\right) &= 2\left(1-\rho_1^{kD_{\text{depth}}(Q)}\right)\sigma_{\text{video}}^2 \\ &= 2\left(1-\rho_1^{kc_1Q}\right)\sigma_{\text{video}}^2. \end{aligned} \tag{3.24}$$

And the depth map bitrate can be expressed using the model proposed in [42] as:

$$R_{\text{depth}}(Q) = \frac{c_2\sigma^2_{\text{depth\_res}}}{Q^2}, \tag{3.25}$$

where c_2 is a scaling factor reflecting sequence characteristics, and $\sigma^2_{\text{depth\_res}}$ is the variance of the depth map residual signal after intra or inter prediction. Hence, if we put (3.24) and (3.25) into (3.21), we get:

$$\lambda = c\ln\rho_1 \cdot \rho_1^{kc_1Q} \cdot Q^3, \tag{3.26}$$

where $c = -\frac{\sigma^2_{\text{video}}}{\sigma^2_{\text{depth\_res}}}\frac{c_1}{c_2}k$. In our experiments in Section 3.5, c_1 and c_2 are calculated by fitting the data points, quantization step size versus sum of absolute difference and bitrate, respectively, using the least square method for each sequence.

3.3 New skip mode selection scheme for reduced flickering artifact

In this section, the new skip mode decision process is proposed by considering local video characteristics. As described in Chapter 2, distortion can occur during depth map estimation. In particular, if there is lack of features to perform stereo matching, the resulting depth map can be noisy, so that it would not be efficient to spend more bits to achieve an accurate representation of the noisy depth map. When the noise is not so much correlated in temporal domain, this can cause a flickering artifact as described in Section 2.6. Therefore, coding such noise would not only increase bitrate but also degrade subjective quality.

To solve this problem, it is necessary to make a decision whether temporal variation in depth map is due to true motion or due to noise. To make this decision we propose to use video data, which would be less noisy than depth map. Before encoding a block of depth data we take into account how the corresponding block of video data was encoded. We note that limited motion regions are also regions where depth information is unlikely to vary over time (in particular if cameras remain fixed). Since limited motion blocks are likely to be encoded using skip mode especially at low rates, we propose to “force” skip mode in depth coding in those blocks for which skip mode was chosen for the video data. Note that in those blocks, skip mode may not have been selected by the conventional encoding methods, because the differences in depth are non-negligible. But, since there is no motion in video, these differences in depth are very likely to be due to unreliable depth estimation, and therefore can be ignored. When a video block is not coded using skip mode, the corresponding depth map block can use all the coding modes including skip mode.

In this way, better coding efficiency can be achieved by taking into consideration depth map unreliability. Flickering artifacts due to temporal variation in depth map are also reduced, leading to overall improvements in perceptual quality. In addition, with this strategy one can select temporal skip in depth automatically, whenever temporal skip in video has been chosen, so that no skip mode information needs to be inserted in the depth bitstream. This leads to reduction in not only bitrate but also encoding complexity, since it is possible to skip the motion estimation and mode decision processes for depth map coding.

The proposed method is simple but efficient to improve coding performance and subjective quality. However, a drawback is that error can propagate by forcing skip mode, if wrong depth value is copied from the previous depth map frame. Another drawback is this scheme cannot detect error in the area of true motion. Therefore, it would be possible to improve the performance by considering these drawbacks. We leave this as future study.

3.4 Graph based transform for depth map coding

DCT has been widely used for block based image and video compression. It provides an efficient way to represent the signal both in terms of coding efficiency and computational complexity as an orthogonal 2-D separable transform. However, it is known to be inefficient for coding blocks containing arbitrary shaped edges. For example, if DCT is applied to a block containing an object boundary which is neither horizontal nor vertical line, e.g. diagonal or round shape, or mixture of these, the resulting transform coefficients tend not to be sparse and high frequency components can have significant energy. This leads to higher bitrate and potentially highly visible coding artifacts if operating at low rate.

To solve this problem variations of DCT have been proposed, such as shape adaptive DCT [37], directional DCT [14,78,79], spatially varying transform [81,82], variable block-size transform [74], direction-adaptive partitioned block transform [7], etc., in which the transform block size is changed according to edge location or the signal samples are rearranged to be aligned to the main direction of dominant edges in a block. Karhunen-Loève transform (KLT) is also used for shape adaptive transform [51] or intra prediction direction adaptive transform [77]. These approaches can be applied efficiently to certain patterns of edge shapes such as straight line with preset orientation angles; however, they are not efficient with edges having arbitrary shapes. Radon transform is used for image coding [41, 53], but perfect reconstruction is only for binary images. Platelets [75] are applied for depth map coding [29], and approximate depth map images as piece-wise planar signals. Since depth maps are not exactly piece-wise planar, this representation will have a fixed approximation error.

Wavelet based approaches have also been studied such as curvelets [5], bandelets [36], contourlet [11, 39], directionlets [63], etc. Edge-adaptive wavelets have been applied for depth map coding by using shape-adaptive lifting [27] and switching between long filters in homogeneous areas and short filters over the edges [10]. Graph-based wavelets are proposed to preserve edge information in a depth map [44]. All these approaches try not to apply transform across the edge; however, these are not amenable to block based coding architecture, which has been widely adopted in international standards for image and video coding such as JPEG, MPEG-2, MPEG-4, H.264/AVC, etc.

To solve these problems, we propose the graph based transform (GBT) as an edge adaptive block transform that represents signals using graphs, where no connection between nodes (or pixels) is set across an image edge. Note that while "edge" can refer to a link or connection between nodes in graph theory, we only use the term "edge" to refer an image edge to avoid confusion. GBT works well for depth map coding since depth map consists of smooth regions with sharp edges between objects in different depths. In this section, it is described how to construct the transform and apply it to depth map coding. Refer to [49] for detailed properties and analysis of the transform.

3.4.1 Construction of graph based transform

The transform construction procedure consists of three steps: (i) edge detection on a residual block, (ii) generation of a graph from pixels in the block using the edge map (iii) construction of transform matrix from the graph.

In the first step, after the intra/inter prediction, edges are detected in a residual block based on the difference between the neighboring residual pixel values. A simple thresholding technique can be used to generate the binary edge map. Then, the edge map is compressed and included into a bitstream, so that the same transform matrix can be constructed at the decoder side.

In the second step, each pixel position is regarded as a node in a graph, G, and neighboring nodes are connected either by 4-connectivity or 8-connectivity, unless there is edge between them. From the graph, the adjacency matrix $\mathbf{A}$ is formed, where $\mathbf{A}(i, j) = \mathbf{A}(j, i) = 1$ if pixel positions i and j are immediate neighbors not separated by an edge. Otherwise $\mathbf{A}(i, j) = \mathbf{A}(j, i) = 0$. The adjacency matrix is then used to compute the degree matrix $\mathbf{D}$, where $\mathbf{D}(i, i)$ equals the number of non-zero entries in the i-th row of $\mathbf{A}$, and $\mathbf{D}(i, j) = 0$ for all $i \neq j$.

In the third step, from the adjacency and the degree matrices, the Laplacian matrix is computed as $\mathbf{L} = \mathbf{D} - \mathbf{A}$ [16]. Then, projecting a signal G onto the eigenvectors of the Laplacian $\mathbf{L}$ yields a spectral decomposition of the signal, i.e., it provides a "frequency domain" interpretation of the signal on the graph. Thus, a transform matrix can be constructed from the eigenvectors of the Laplacian of the graph. Since the Laplacian $\mathbf{L}$ is symmetric, the eigenvector matrix $\mathbf{E}$ can be efficiently computed using the well-known cyclic Jacobi method [43], and its transpose, $\mathbf{E}^{\mathrm{T}}$, is taken as GBT matrix. Note that the eigenvalues are sorted in descending order, and corresponding eigenvectors are put in the matrix in order. This leads to transform coefficients ordered in ascending order in frequency domain.

It is also possible to combine the first and second steps together. Instead of generating the edge map explicitly, we can find the best transform kernel for the given block signal by searching the optimal adjacency matrix. When 4-neighbor connectivity is considered in a 4×4 block, there are 12 horizontal edges and 12 vertical edges. Accordingly there are 2^{24} possible adjacency matrices. Instead of searching the whole space to find the optimal adjacency matrix, a greedy algorithm can be applied. By defining a cost function, the cost for including each edge can be calculated. Then, the number of edges are increased from zero to 24, leading to stages 0 to 24. At stage 0 the cost is calculated when there is no edge. At stage 1 the cost is calculated for each edge location by setting one of them as an edge at a time. The one with the minimal cost is selected as the optimal edge at stage 1. Then at stage 2, the edge found in stage 1 is included, and an additional edge is found as in the stage 1 excluding the edge found in stage 1. We can calculate the cost for each stage by including additional edges, and choose the best stage which results in the minimal cost. Then, this will give the optimal adjacency matrix.

The equations below show the cost function to search the optimal adjacency matrix using the greedy algorithm.

$$\mathrm{Cost}_{\mathrm{coeff}} = \mathbf{f}^{\mathrm{T}}\mathbf{L}\mathbf{f} = \sum_i \lambda_i \alpha_i^2 = \frac{1}{2}\sum_{i,j} a_{ij}(f_i - f_j)^2, \tag{3.27}$$

$$\mathrm{Cost}_{\mathrm{coeff\_rate}} = \log_2\left(\frac{\sum_{i,j} a_{ij}(f_i - f_j)^2}{2Q^2}\right), \tag{3.28}$$

$$\begin{aligned} \mathrm{Cost} &= \mathrm{Cost}_{\mathrm{coeff\_rate}} + k\mathrm{Cost}_{\mathrm{edge\_rate}} \\ &= \log_2\left(\frac{\sum_i a_{ij}(f_i - f_j)^2}{2Q^2}\right) + km, \end{aligned} \tag{3.29}$$

where $\mathbf{f}$ is a vector of the input depth map block, f_i is the i-th element in this vector, a_{ij} is the corresponding element in the adjacency matrix, and Q is quantization step size. The edge rate is that needed to code the adjacency matrix, which can be represented using 24 bits and further compressed using entropy coding. The scaling factor k can be applied to balance the coefficient rate and edge rate.

Transform coefficients are computed as follows. For an $N \times N$ block of residual pixels, form a one-dimensional input vector $\mathbf{x}$ by concatenating the columns of the block together into a single $N^2 \times 1$ dimensional vector, i.e., $\mathbf{x}(Nj + i) = X(i, j)$ for all $i, j = 0, 1, ..., N - 1$. The GBT transform coefficients are then given by $\mathbf{y} = \mathbf{E}^{\mathrm{T}} \cdot \mathbf{x}$, where $\mathbf{y}$ is also an $N^2 \times 1$ dimensional vector. The coefficients are quantized with a uniform scalar quantizer followed by entropy coding. Unlike DCT which uses zigzag scan of transform coefficients for entropy coding, GBT does not need any such arrangement since its coefficients are already arranged in ascending order in frequency domain.

To achieve the best performance one can choose between DCT and GBT. For example for each block the RD cost can be calculated for both DCT and GBT, and the best one can be selected. Overhead indicating the transform that was chosen can be encoded into the bitstream for each block, and the edge map is provided only for blocks coded using GBT.

3.4.2 Graph based two-channel transform

In the previous subsection, an image block is represented as a graph using an edge map, and the transform matrix is formed using the eigenvector of Laplacian matrix, which is a spectral representation of graph. Even though this approach can perform better than the conventional DCT for blocks containing complex shape edges, one drawback is the computational complexity of finding all the eigenvectors of the Laplacian matrix. To resolve this issue we propose an alternative transform, which has similar performance to GBT in terms of the coding efficiency, but with less computational complexity.

The transform bases of the GBT consist of eigenvectors of the Laplacian matrix of a graph. Narang and Ortega [32] showed that an alternative transform can be found by properly defining polynomial kernels and applying them to the Laplacian matrix. One restriction is that this transform should be applied to a connected graph. We propose applying this transform to image blocks as follows.

First, an image block is represented as a graph as described in Section 3.4.1. Then, using its adjacency matrix, we find connected components, which are the pixels not separated by an edge. For each group of connected components, the Laplacian matrix is defined as in Section 3.4.1. For example, if there is one edge in a block dividing the block into two regions, there will be two connected components groups and two Laplacian matrices representing each group. Then, for each Laplacian matrix a two-channel transform, $\mathbf{T}_{\text{low}}$ and $\mathbf{T}_{\text{high}}$ can be defined with low pass and high pass operators as [32]:

$$\mathbf{T}_{\text{low}} = a_1\mathbf{L}_i + a_0\mathbf{I}, \tag{3.30}$$

$$\mathbf{T}_{\text{high}} = b_1\mathbf{L}_i + b_0\mathbf{I}, \tag{3.31}$$

where $\mathbf{L}_i$ is the Laplacian matrix corresponding to each connected component, $a_0 = 1$, $a_1 = -1/\left(2 \times d_{\max} + 1\right)$, $b_0 = 0$, $b_1 = 1/\left(2 \times d_{\max} + 1\right)$, with $d_{\max}$ representing the maximum element in the degree matrix of the group, and $\mathbf{I}$ is $N_i \times N_i$ identity matrix with N_i indicating number of components in the group.

Then this two-channel transform is applied to even and odd components of the corresponding connected components in the graph. We define the first node in a connected component as the even node and others as the odd nodes. Then, the transform matrix is formed by combining the first row of $\mathbf{T}_{\text{low}}$ and the others from $\mathbf{T}_{\text{high}}$. This implies that the first transform coefficient from the first row of the transform matrix conveys weighted average of the connected components similar to DC component in DCT, and other coefficients generate weighted difference of the connected components similar to AC components in DCT, where the weights relates to local connectivity. We call this as a graph based two-channel transform (G2T). Since this is not an orthogonal transform, its inverse transform matrix can be found by matrix inversion.

For entropy coding transform coefficients in each group is interleaved in ascending order of frequency, so that all the coefficients in a block can be efficiently compressed using run-length coding. Fig. 3.2 shows an example of coefficient ordering for entropy coding, where there are 4 groups of connected components, and 7, 6, 2, and 1 components are in each group. The number in each square indicates the order.

3.4.3 Application to depth map coding with sparsification

When a depth map is compressed, it is desirable to consider the distortion in the rendered view from depth error. Cheung *et al* [8, 9] defined a "don't care region" based on the sensitivity of the rendered view distortion to depth map errors. Based on this the input depth map signal can be modified to increase the coding gain. A transform domain sparsification (TDS) technique is used to modify the signal as described in [8]. While TDS tries to sparsify the signal by modifying the input signal, GBT works to sparsify from the original signal by not applying transform

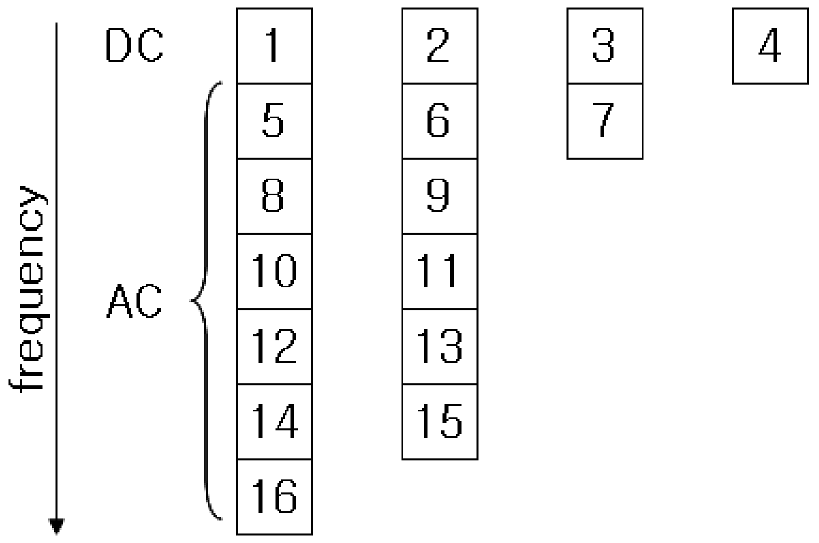


Figure 3.2: Example of G2T transform coefficient ordering for entropy coding, where a block consists of 4 connected component groups.

across the edges. Since they work in different manner, it is expected that additional coding gain can be achieved by combining them.

To combine TDS and GBT, the quantized GBT coefficients are used for TDS for the depth map blocks with prominent edges, which therefore would be coded using GBT. For other blocks, quantized DCT coefficients are used for TDS. One thing to consider is how to determine the GBT kernel on which to perform TDS. Since the edges in the original signal can be modified by TDS, it will be desirable to consider the edges which will not be affected by TDS.

Unlike typical edge detection in computer vision literature for semantic-related tasks such as object segmentation or recognition, the goal here is strictly for coding; i.e., identification of "edges" that cannot be efficiently coded using non-adaptive transforms. For simplicity, we first detect prominent edges in depth maps based on the absolute difference between neighboring pixels; i.e., we check if the difference exceeds a threshold θ [49]. Then, we evaluate each edge location to determine whether it contributes to a sparser representation if included in the transform. To identify a good transform at low computational complexity, we perform the following simple procedure:

1. Given detected edges using threshold θ, construct a graph using *all* detected edges and perform GBT on the block. This results in sparsity count ρ, which is number of non-zero quantized coefficients.

2. Rank the importance of edges based on difference of depth pixel values across the edges. Initialize a no-edge graph (nodes representing neighboring pixels are all connected by links).

3. For given graph, sparsify the graph transform representation using TDS. If sparsity count equals ρ, store the current graph and stop. Otherwise, proceed to the next step.

4. Add the next most important edge to the graph (remove the link between nodes representing neighboring pixels crossing the edge). Go to step 3.

Because TDS finds sparser representations easier as more edges are added, the procedure tends to terminate with sparsity count $\geq \rho$ before all the detected edges are added, resulting in a net-positive coding gain. Moreover, ρ is a sparsity count that is demonstrably achievable, since it was the pre-set value for GBT-only. This means the procedure is guaranteed to exit, in the worst case when all the detected edges are specified.

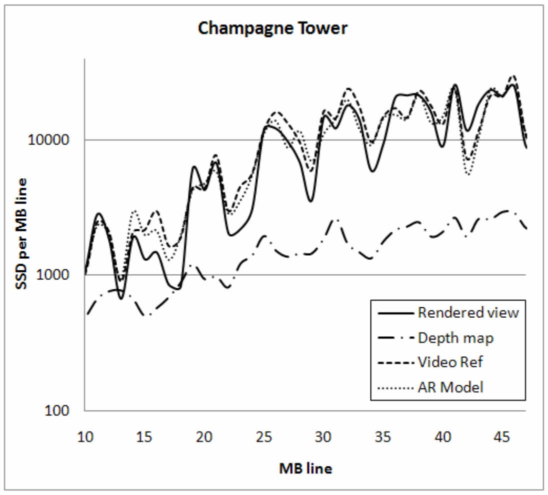
3.5 Experimental results

3.5.1 New distortion metric and RD optimization

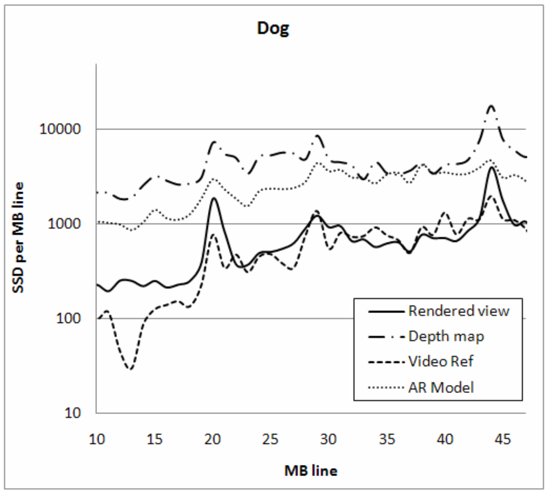
The new distortion metrics and RD optimized mode selection scheme are implemented based on H.264/AVC (joint model reference software ver. 13.2), and verified with experiments using various multiview test sequences. MPEG multiview video test sequences are used to generate the depth map for each sequence using the depth estimation reference software ver. 3.0 [59]. In addition, for the 'Ballet' and 'Breakdancers' sequences, color video and depth maps are provided by Microsoft Research [83]. Each sequence has different characteristics and camera settings, e.g., different amount of texture and object edges, parallel or non-parallel, dense or sparse camera settings, capturing long or short range of distance, etc. Note that different camera settings affect the amount of geometry error caused by depth map distortion, which can be captured by the linear relationship in (3.9) introduced in Section 3.1.1.

To generate the rendered view the view synthesis reference software (VSRS) 3.0 [58] is used. In our simulations, in order to compare rendering results with ground truth (captured view), we select the target view to be positioned at the same location as one of the captured views, and use its two neighboring views and the corresponding depth maps as input for VSRS. For example, we can render view 4 of Ballet using view 3 and view 5 as input. Since the target view is between these two reference views, in order to derive the geometry error from depth coding distortion, the distance between p and p' is set to one camera distance (one baseline) in (3.8). Both input color video and corresponding depth map are coded and decoded using the same QP.

First, in order to evaluate the accuracy of the proposed distortion estimation methods, the new distortion metrics in Sections 3.1.2 and 3.1.3, which we now refer to as ‘Video Ref’ and ‘AR Model’, respectively, are compared to the actual distortion in the rendered view. Note that there are two sources of distortion in the rendered view - depth map distortion and rendering process. Since we want to consider only the effect due to depth map distortion, the actual distortion is calculated by generating the rendered view with and without coding the depth map, and then taking the sum of squared difference (SSD) between these two versions for each macroblock (MB). Fig. 3.3 shows the comparison in terms of distortion in the rendered view, distortion in the compressed depth map, and estimation result based on ‘Video Ref’ and ‘AR Model’, respectively. For better visualization, total SSD for all MBs in a row within a frame is computed and plotted (in log scale) as a function of the row index. In Fig. 3.3 (a), the depth map distortion is much smaller than that of the rendered view, on the other hand, this is the opposite in Fig. 3.3 (b). This is because ‘Champagne Tower’ captures a scene with content much closer to the camera than that of ‘Dog’, which results in larger scaling factor in (3.9), and consequently larger geometry error. At the same time the pixel intensity variance of ‘Champagne Tower’ is larger than that of ‘Dog’. Both factors together lead to the different trends shown in Figs. 3.3 (a) and (b). In both cases, the proposed methods more closely follows the actual distortion occurred in the rendered view than the depth map distortion, i.e., they provide a more precise distortion estimation than using the distortion of the depth map. Furthermore, ‘Video Ref’, which estimates the rendering distortion by referring to the video with pixel by pixel geometry error, gives better results than ‘AR Model’ which approximates the distortion with global autoregressive model.



(a) Champagne Tower

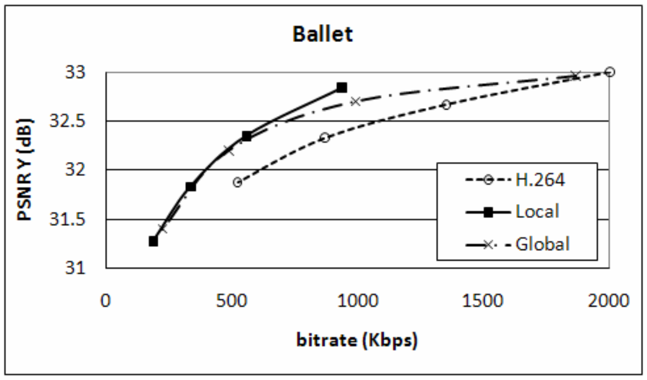


(b) Dog

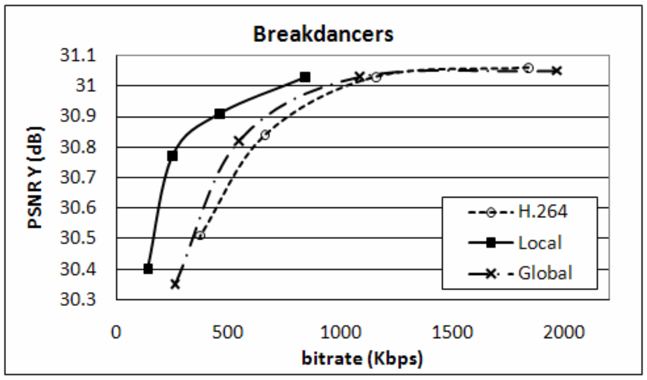
Figure 3.3: Distortion estimation evaluation in comparison with the rendered view distortion. x-axis: MB row index; y-axis: total SSD for all MBs in a row (in log scale).

Secondly, the performance of the proposed distortion estimation method based on local video characteristics is compared to the one based on global video characteristics. In our previous work [18] the rendered view distortion is estimated using global video characteristics, in which the mapping from geometry error to rendered view distortion is determined using a global parameter calculated from a set of frames. However, local video characteristics can affect the rendered view distortion. For example when a local area of a video frame contains complex texture, even a small amount of geometry error can cause large distortion. On the other hand, geometry error would not result in much distortion in flat region in a video frame. Since the proposed methods estimate the rendered view distortion for each pixel in 'Video Ref' and for each block in 'AR Model', they can reflect local video characteristics very well. Fig. 3.4 shows the performance comparison between the global and local estimation methods when they are applied to the RD optimized mode selection scheme, where 'Global' indicates the global estimation method in [18], and 'Local' is the local estimation method as proposed in Section 3.1.2 without applying the new Lagrange multiplier. Compared to 'H.264' where depth map distortion itself is used for mode decision, both global and local distortion estimation methods show good performance. In both test sequences, the local estimation method performs better than the global one. Note that the performance gap between local and global methods is larger in 'Breakdancers' sequence than in 'Ballet' sequence. The former contains a complex scene with many objects and the scene in the latter is simple with flat regions in the background. Therefore, the global estimation can match the performance of the local method in the latter case, while the local method can work significantly better when the video contains a complex scene (as in the former case).

Thirdly, the new Lagrange multiplier derived in Section 3.2.2 is applied along with the new distortion metrics to the RD optimized mode selection. To generate RD curves, two sets of video and depth map are coded using same QP values. Then, using the decoded video and depth map, an intermediate view is rendered, which



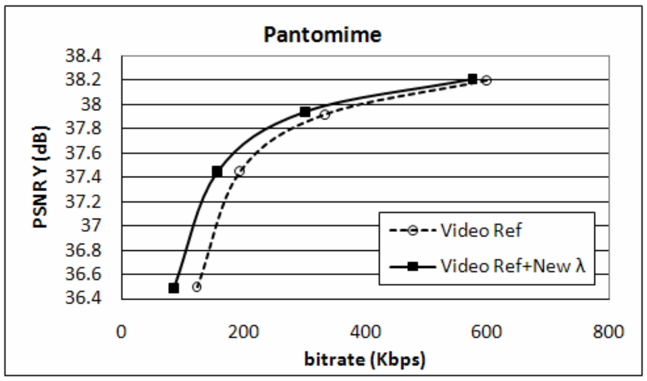
(a) Ballet



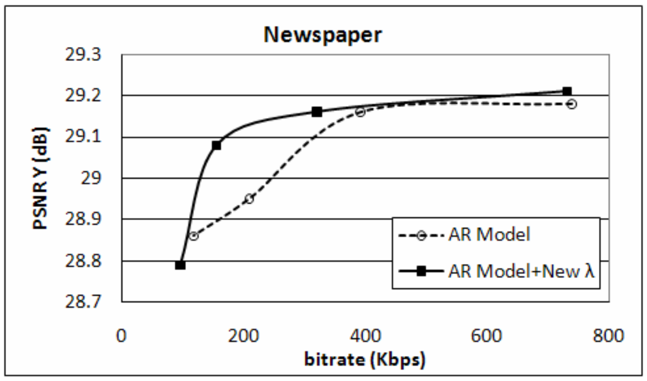
(b) Breakdancers

Figure 3.4: Comparison between global and local distortion estimation methods. x-axis: total bitrate to code two depth maps; y-axis: PSNR of luminance component between the rendered view and the ground truth.

can be compared with the ground truth (the captured view) to generate peak signal to noise ratio (PSNR) of luminance component. Fig. 3.5 (a) is for 'Video Ref' and Fig. 3.5 (b) is for 'AR Model', where RD curves are generated with and without the new Lagrange multiplier. For both methods, the new Lagrange multiplier improved the coding efficiency.



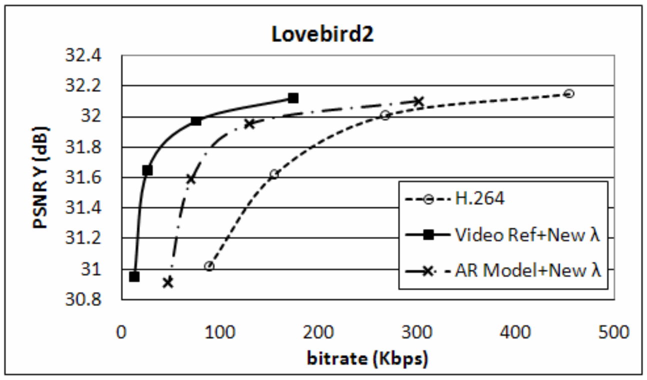
(a) Pantomime



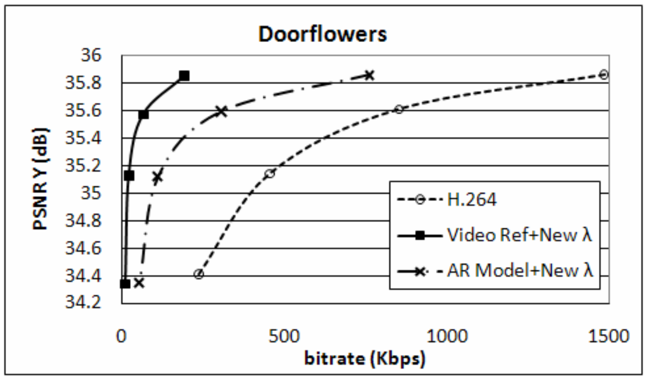
(b) Newspaper

Figure 3.5: Rate-distortion performances of the proposed methods with and without new Lagrange multiplier.

Finally, the RD performance is compared to that of H.264/AVC as shown in Fig. 3.6. The BD-PSNR and BD-bitrate [3] are calculated using various multiview sequences, and listed in Table 3.2. On average, 0.6 dB BD-PSNR improvement or 70% bitrate saving is observed by 'Video Ref', and 0.25 dB and 35% by 'AR Model'.



(a) Lovebird2



(b) Doorflowers

Figure 3.6: Comparison of the rate-distortion curves between the proposed methods and H.264/AVC.

From the RD curves in Figs. 3.4-3.6, it can be observed that the PSNR saturates as bitrate increases. This is because the PSNR is calculated compared to the ground truth, and there is distortion even without coding video and depth maps due to various factors such as occlusion, depth estimation error, etc., as discussed in Section 2.2. It would be possible to calculate PSNR compared to the synthesized view without coding video and depth maps. However, we use the ground truth

Table 3.2: BD-PSNR (dB) and BD-bitrate (%)results of the proposed methods, 'Video Ref' and 'AR Model'.

Sequence	Video Ref		AR Model	
	BD-PSNR	BD-bitrate	BD-PSNR	BD-bitrate
Dog	0.56	61.84	0.32	45.53
Lovebird 1	0.28	71.86	0.01	-21.80
Lovebird 2	0.65	81.98	0.34	44.85
Door Flowers	1.64	94.29	0.64	72.16
Newspaper	0.22	78.86	0.10	37.88
Pantomime	0.54	49.84	0.14	19.38
Ballet	0.57	51.98	0.30	27.12
Breakdancers	0.14	64.77	0.12	54.94
Average	0.58	69.43	0.25	35.01

whenever it is available to calculate PSNR so that we can have better sense how much the result is close to the original view.

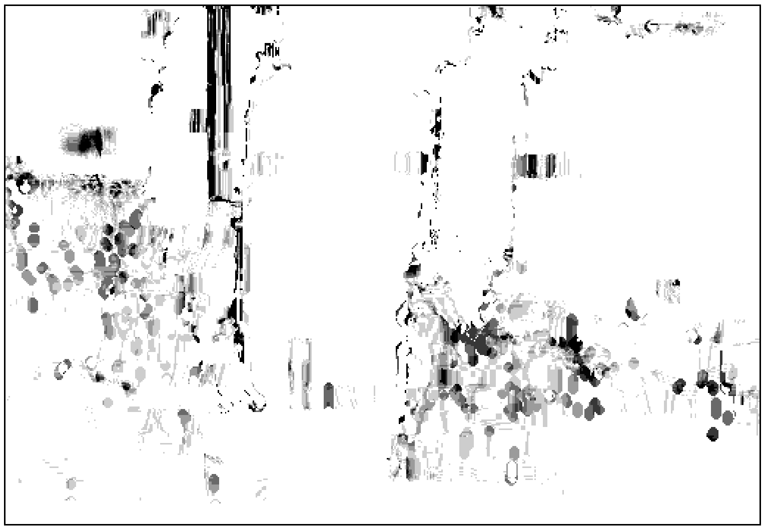
3.5.2 New skip mode selection scheme

We now evaluate the new skip mode selection scheme. By using this new approach, subjective quality can be improved because flickering artifacts are reduced. Flickering artifacts occur in the synthesized views due to false temporal variation in depth map. With the proposed method, false temporal variation in depth map can be suppressed by use of skip mode, and as a result the flickering artifact can be reduced.

To see the temporal variation in the static background region without and with the proposed method, the bottom right quarter of the synthesized Ballet sequence is taken from two temporally consecutive frames, and the difference image is shown in Fig. 3.7. It can be easily noticed that the temporal variation has been significantly reduced by the proposed method, leading to flickering artifact reduction. This also can lead to significant bitrate savings, since we are not coding the noise in the erroneous depth map blocks and just copying them from the previous frame. Table 3.3 lists the bitrate saving achieved using sequences with various amounts of temporal variation in depth map. The proposed measurement for the amount of flickering artifact due to depth map error, F, is also listed in Table 3.3. The coding efficiency improvement does not always match F, since various factors affect the coding efficiency improvement other than F, such as variance of depth map block, baseline distance, contrast between foreground and background objects, etc. However, it can be noticed that as F increases, better coding efficiency can be achieved by not coding false temporal variation due to depth map error. This shows that as false temporal variation increases the coding gain also increases, with maximum bitrate saving of 66 % and on average, 25 %.

Table 3.3: BD-PSNR/bitrate results of the proposed new skip mode selection scheme.

Temporal Variation	Sequence	F	BD-PSNR (dB)	BD-bitrate (%)
Little	Champagne Tower	0.2	1.1	26.4
	Newspaper	0.2	1.2	28.5
	Cafe	0.3	0.1	6.5
	Pantomime	0.7	0.5	19.2
	Lovebird 1	0.8	0.3	6.9
Moderate	Balloons	2.0	1.7	39.7
	Mobile	2.6	0.3	2.6
	Beergarden	6.8	0.5	8.0
	Kendo	7.6	1.2	28.5
	Book Arrival	8.8	1.1	26.7
Large	Street	20.7	0.4	11.8
	Hall 1	26.0	1.6	66.0
	Car Park	40.3	2.4	52.4
	Hall 2	46.4	0.5	20.5
Average			0.9	24.5


(a) H.264/AVC

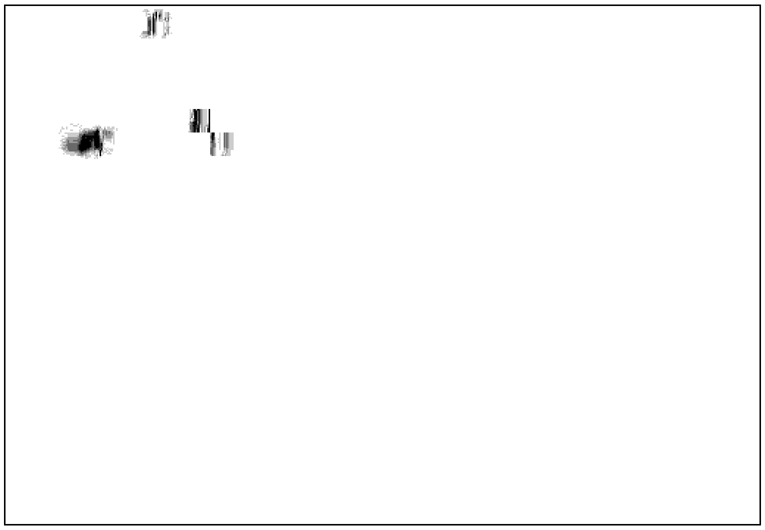
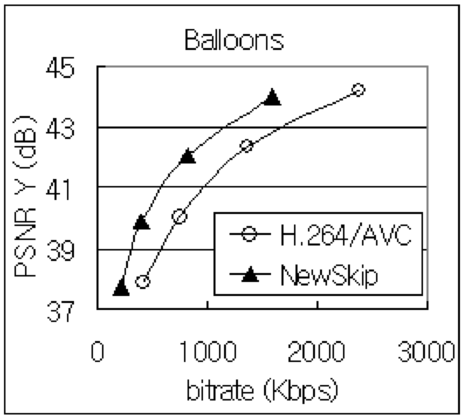
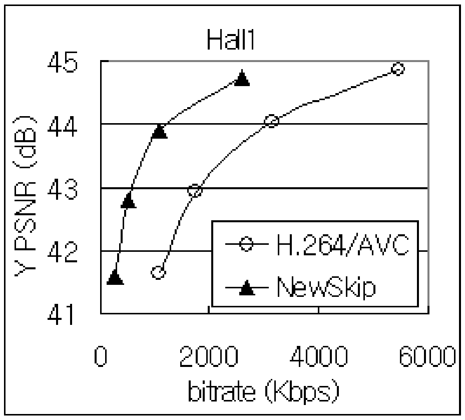

(b) Proposed methods

Figure 3.7: Example of flickering artifact reduction: (a) H.264/AVC and (b) proposed method.

One possible drawback of the proposed scheme is that false depth map value can propagate when using the skip mode if the reference depth map block which is copied by the skip mode to the next frame contains wrong depth value. This can be observed from the RD curves in Fig. 3.8, where large bitrate saving can be noticed with negligible PSNR drop.



(a) Balloons



(b) Hall 1

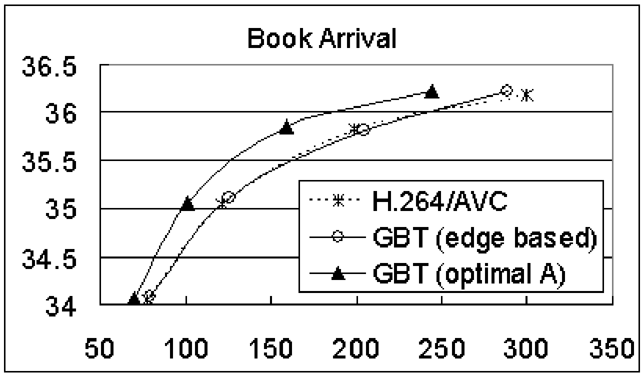
Figure 3.8: RD curves using the new skip mode selection scheme compared to H.264/AVC: (a) Balloons (b) Hall 1.

While large bitrate saving can be achieved by not coding the detail of the area with false temporal variation, part of the reason for little PSNR drop can be propagation of wrong depth information. This problem can be solved by using the proposed RD optimized mode selection scheme in Section 3.1. Because the proposed RD optimized mode selection scheme considers the rendered view distortion, this will prevent propagation of wrong depth map value by using skip mode, if the distortion is too large compared to bitrate saving. We leave study on combination of the proposed tools as future study.

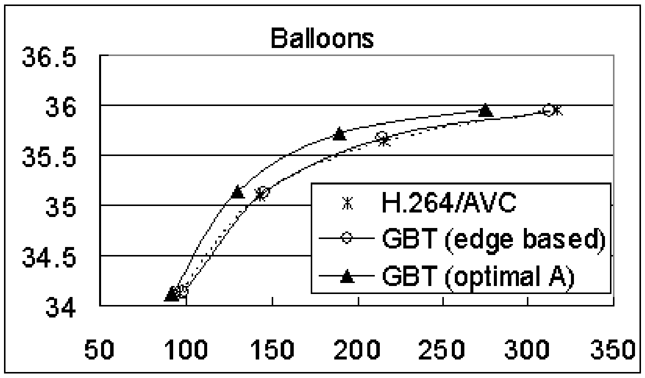
3.5.3 Graph based transform

In this subsection, the performance of GBT is evaluated using various test sequences. The implementation is based on H.264/AVC reference software JM17.1. The transform mode is signaled for each 4×4 block to indicate the best transform between the H.264/AVC integer transform which is a modification of DCT and the proposed GBT. GBT kernel is generated in two ways. First, normal edge detection scheme is applied to detect edges in a block. In this case, for the blocks coded using GBT, the edge map is losslessly encoded and sent to the decoder. Secondly, instead of finding edge map, we find optimal adjacency matrix using the method described in Section 3.4.1, where the adjacency matrix is losslessly encoded and sent to the decoder. We compare these two methods to the case where the depth maps are encoded using H.264/AVC.

Fig. 3.9 shows the RD curve comparison between the proposed methods and H.264/AVC, where the bitrates for GBT cases include transform selection bits and edge map or adjacency matrix information bits in addition to depth map coding bits. QP values of 24, 28, 32, and 36 are used to encode depth maps. PSNR is calculated by comparing the ground truth video and the synthesized video using the decoded depth maps.



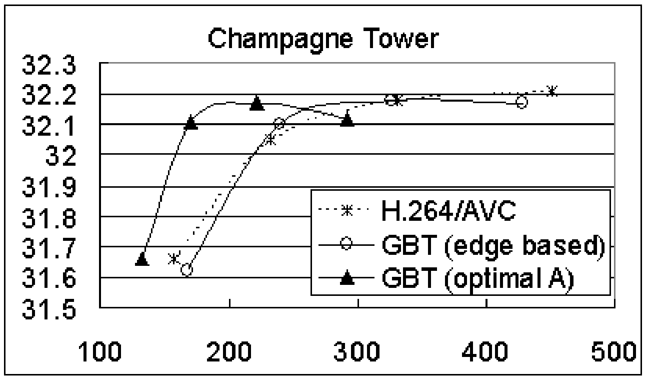
(a) Book Arrival



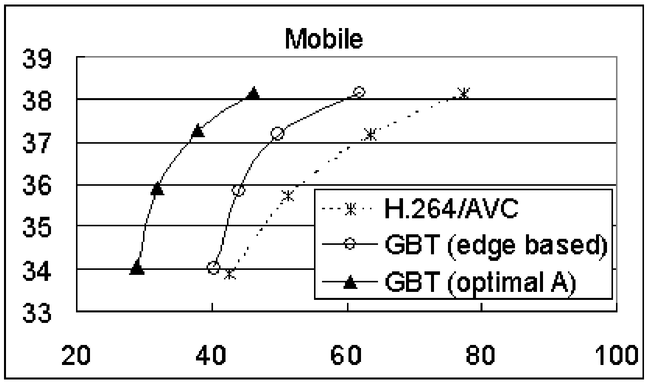
(b) Balloons

Figure 3.9: RD curve comparison between GBT and H.264/AVC, where GBT is formed using edge detection or by finding optimal adjacency matrix: (a) Book Arrival (b) Balloons.

From the RD curves in Fig. 3.9, it can be noticed the GBT generated using optimal adjacency matrix performs better than H.264/AVC, while the GBT generated using edge detection scheme does not perform better than H.264/AVC most of the cases. It is observed that significant amount of bitrate saving can be achieved



(c) Champagne Tower



(d) Mobile

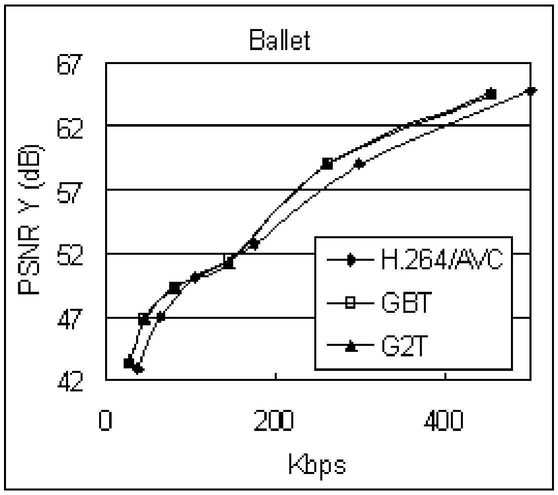
Figure 3.9: RD curve comparison between GBT and H.264/AVC, where GBT is formed using edge detection or by finding optimal adjacency matrix: (c) Champagne Tower (d) Mobile.

by GBT, while there is little PSNR improvement. The bitrate saving increases as overall bitrate increases. This is because more number of blocks are chosen to be coded using GBT in high bitrate, since the portion of additional bits for adjacency matrix in total bitrate reduces. It can be also noticed that different performance gain is achieved in each test sequence. The performance of GBT depends on the amount of edges in a frame, and the strength of edge. If there is large amount of noise around object boundary, GBT may not provide much gain over DCT. Among various test sequences, Champagne Tower contains large amount of edges in a frame with relatively strong edge strength. In case of Mobile sequence, there are strong edges along the object boundary with relatively less amount of noise compared to other sequences. Therefore, large gain can be achieved in these two sequences. All the results for 11 test sequences are given in Table 3.4, where the coding efficiency is represented using BD-PSNR and BD-bitrate.

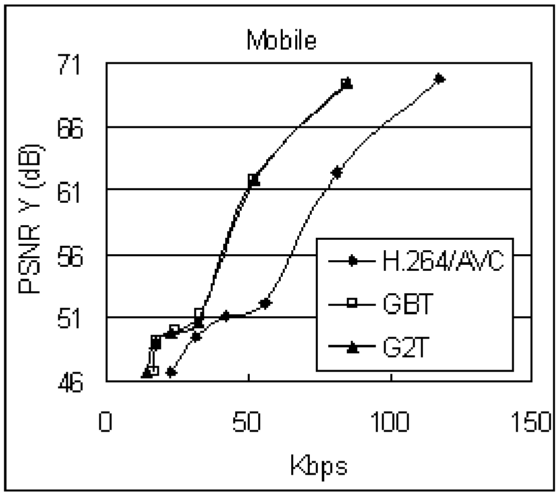
Table 3.4: BD-PSNR/bitrate results of GBT compared to H.264/AVC.

Sequence	BD-PSNR	BD-bitrate
Lovebird 1	0.0	-7.1
Cafe	0.1	-39.6
Newspaper	0.1	-7.7
Book Arrival	0.3	-17.7
Balloons	0.2	-8.3
Champagne Tower	0.2	-76.2
Kendo	0.1	-6.3
Pantomime	0.1	-5.2
Mobile	3.6	-38.7
Car Park	0.1	-13.0
Street	0.2	-9.3
Average	0.4	-20.8

Secondly, the performance of GBT and G2T is compared to that of DCT in terms of the depth map distortion. The RD curves are generated using Ballet and Mobile sequence as can be seen in Fig. 3.10. Note that the PSNR is generated by comparing the original and decoded depth maps. It can be noticed that GBT and G2T show almost same performance, which is significantly better than that of DCT.



(a) Ballet



(b) Mobile

Figure 3.10: RD curves of GBT and G2T compared to DCT. x-axis: bitrate to code a depth map; y-axis: PSNR of depth map.

Next, the performance of GBT and combination of GBT and TDS are evaluated using the test images, Teddy and Dolls, among Middlebury stereo datasets [46, 47], where the ground truth disparity maps for the left and the right view are compressed, and the decoded disparity maps are used for synthesis of the middle view between the left and right views. The original intensity image is used for the view synthesis. Even though disparity maps are used for the experiments, similar performance is expected when the proposed method is applied to depth map coding considering the relationship between them [66]. The view synthesis is performed by warping the left and right views to the target view position, then a blending process is applied if more than one pixels are mapped to the same position, which is weighted averaging using the distance from the reference view to the target view. For example, the weight is half if the middle view is synthesized between the left and right views. When there is no pixel mapped to a position, hole filling process is applied for this position by copying the horizontally nearest neighboring pixel value.

To compare the performance of the proposed method, which is the combination of TDS and GBT (TDS+GBT), various conventional methods are applied such as DCT, GBT, and TDS to compress the disparity maps. A block size of 4×4 is used for the transforms, followed by uniform quantization and CABAC as entropy coding. We used the integer transform in H.264/AVC as DCT, and the reference software JM 17.1 is used for the experiments. The same software is used to implement GBT and the proposed method. Note that for exploration purpose neither intra nor inter prediction is performed in our experiments. The inter-view prediction is not applied, either.

Fig. 3.11 shows the RD curves generated using various coding methods such as DCT, GBT, TDS, and the proposed method, TDS+GBT. Fixed QP values

Table 3.5: BD-PSNR (dB) and BD-bitrate (BDBR, %) results of GBT, TDS, and the proposed methods, GBT+TDS, compared to DCT.

Sequence	Teddy		Dolls	
	BD-PSNR	BDBR	BD-PSNR	BDBR
GBT	0.7	-24	0.3	-7
TDS	1.0	-31	0.9	-19
TDS+GBT	1.6	-34	1.2	-32

of 24, 28, 32, and 36 are used to code left and right disparity maps, and the bitrate is represented in the unit of bits per second assuming 30 frames per second (fps). While both GBT and TDS performs better than DCT, TDS+GBT provides additional coding gain for both test sequences, 0.6 dB and 0.3 dB for Teddy and Dolls, respectively, in terms of BD-PSNR [3] as shown in Table 3.5. The shape of the RD curves of TDS+GBT looks less straight compared to DCT or GBT, which is inherited from TDS. One reason for this is during TDS procedure the same λ value is applied for all QP values for TDS and TDS+GBT, 0.05 for Teddy and 0.5 for Dolls, respectively. It is expected that better performance can be achieved by adapting λ to QP value as the Lagrange multiplier can be adapted to QP value for optimization [19].

From the RD curves, it can be noticed that the bitrate is reduced, while the PSNR is improved. The bitrate reduction is achieved by efficiently combining TDS and GBT processes. While the PSNR improvement in GBT is achieved by preserving well the object boundaries in disparity maps, that of TDS is achieved by providing expanded smooth object boundaries for the foreground area. We leave as a further study topic understanding exactly what causes this and how to achieve optimal performance.

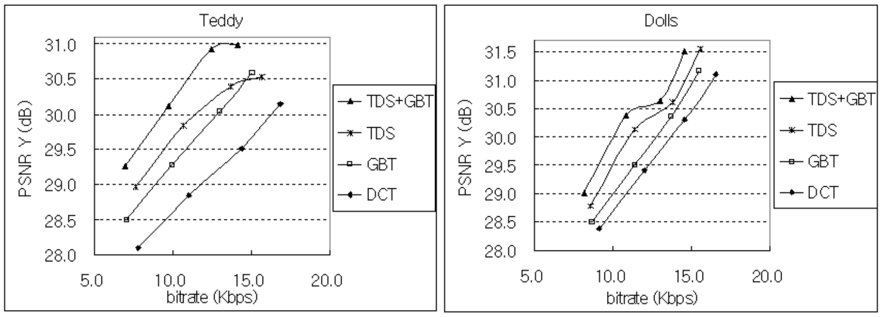


Figure 3.11: Rate-distortion curves of DCT, GBT, TDS, and the proposed method, TDS+GBT. x-axis: total bitrate to code two disparity maps; y-axis: PSNR of luminance component between the rendered view and the ground truth.

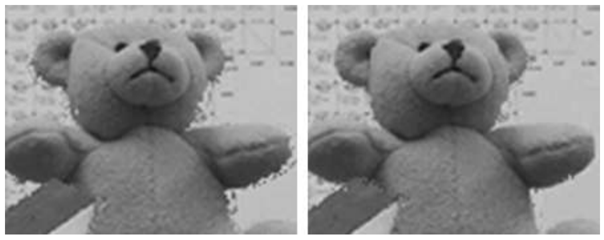


Figure 3.12: Subjective quality comparison of synthesized view of Teddy using DCT (left) and the proposed method, TDS+GBT (right).

Fig. 3.12 shows the subjective quality comparison between the synthesized view using DCT coded disparity map (left) and the one with the proposed method, TDS+GBT (right). In the DCT result it can be easily noticed that there is distortion along the foreground object boundary. Using the TDS+GBT, the distortion along the object boundary area is reduced. Both results are generated using QP value of 24, in which the bitrate of TDS is 16% lower than that of DCT.

3.6 Conclusion

In this chapter various coding algorithms are proposed to efficiently compress depth maps. First, we propose new distortion estimation methods, which derive the relationship between the depth map error to geometry error, and geometry error to the distortion in the rendered view. These estimation methods are applied to the RD optimized mode selection scheme along with the new Lagrange multiplier derived using the autoregressive model. The experimental results show that the proposed methods can provide accurate estimation of the rendered view distortion due to depth map error, so that coding efficiency improvement of 0.6 dB or 70 % bitrate savings on average can be achieved when it is applied to the depth map coding mode decision scheme.

In addition, the new skip mode selection scheme improves the subjective quality by suppressing the flickering artifact, which occurred by temporal inconsistency in depth map. This also results in significant bitrate savings by not coding noise in depth map. Experimental results shows 0.9 dB PSNR gain or 25 % bitrate savings on average.

Also the graph based transform (GBT) is proposed for depth map coding, which not only reduces bitrate but also improves the rendered view quality by preserving edges from transform coding. It is observed 0.4 dB PSNR coding gain or 20 % bitrate savings can be achieved when applied to depth map coding.

All these algorithms are developed considering depth map specific characteristics. To achieve the optimal performance it will be necessary to study how to combine these tools together. We leave this as a future study topic.

Chapter 4

New 3-D Video Format

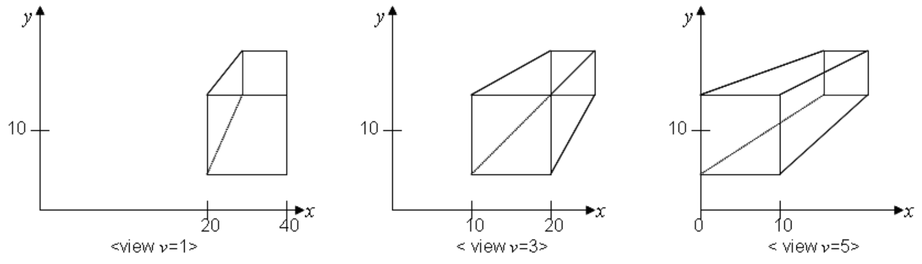
4.1 Depth transition data

As discussed in Chapter 2, depth map coding distortion can lead to significant subjective quality degradation, and it is found that the distortion in the rendered view due to depth map quantization error has non-linear and localized features. In this section we propose a new 3-D video format to improve the subjective quality of the rendered view. Since the cause for the artifact is depth map coding error, we can solve this problem by providing additional information which can provide more precision to the depth data.

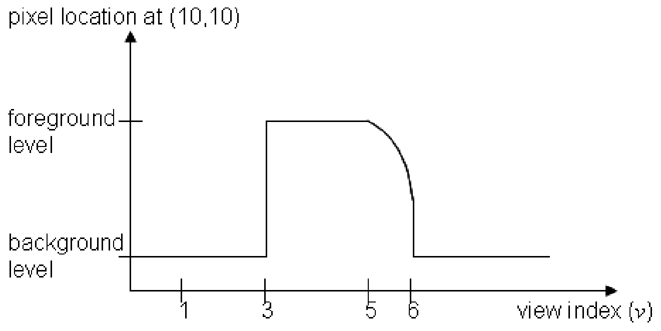
One possible approach to compensate the rendered view distortion would be to provide additional information for each intermediate rendered view. A simple example of this would be to synthesize views at the encoder and transmit a residue between the synthesized view and the original captured video. However, this solution is not feasible to synthesize an arbitrary view position if the ground truth video for this position is not available. Moreover, even though the ground truth video is available, this solution is not attractive because the required overhead will increase with the desired number of possible interpolated views. Instead, our goal is to provide auxiliary data that complements depth information and can be used *to improve rendering of multiple intermediate views*. With this approach the same auxiliary data can be sent, giving users maximum flexibility in determining which view should be interpolated.

To achieve these goals, we propose transmitting *depth transition data* (DTD) for specific pixel locations. Consider a pixel location in two different views. Because of the different camera position the depth value at the same image coordinate can be different in each view. DTD is the *view position* in between the two existing views at which the depth value for a given pixel location will change from that of the same pixel location in the right view to that of the same pixel location in the left view. Note that this implies that we do not need to send DTD if a given pixel has the same depth in both left and right views. Fig. 4.1 shows an example of this, where a cube object is captured with three horizontally different camera positions as shown in Fig. 4.1 (a). As the view index increases, the cube object moves to the left in the image frame. Therefore, for a given pixel location, we can trace how the depth value for that pixel location changes as a function of the chosen intermediate camera position as shown in Fig. 4.1 (b).

Note that conventional depth map information is provided for each reference view and is available for every pixel position, while DTD is associated to view pairs and is only transmitted for certain pixel locations. Since both standard depth information and DTD are transmitted, DTD allows us to spend additional bits in specific locations in order to improve view interpolation. Thus, we transmit DTD only for the subjectively important portions of the video, exploiting the highly localized nature of regions that are more sensitive to view interpolation errors (as shown in Figs. 2.9 and 2.10). The coding precision of DTD can also be easily adjusted depending on the desired density of intermediate views to be generated



(a)



(b)

Figure 4.1: Depth transition example (a) cube object captured in horizontally different camera positions (b) depth value transition curve at a specific image coordinates

at the decoder (i.e., coarser quantization can be used if the number of interpolated views is small).

Beside the DTD precision to represent the intermediate view location, another precision of DTD is how precisely represent depth information. For example, if it is represented like an 8 bit depth map, DTD can trace depth transition of every single depth level change. If it is represented as 1 bit data, DTD can trace depth change between two depth layers, i.e., foreground and background. Therefore, it is possible to represent any depth level precision using DTD. The other precision related to DTD is how many number of transition between two views can be recorded. It is possible there are more than one transition at a pixel position between two views. This will be more likely when baseline distance is large. For simplicity in our research the precision of DTD is limited to represent one intermediate view position between two reference views with tracing transition between two depth layers, i.e., foreground and background, and to record only the first transition happened between two reference views. In the next section a more detailed description is provided to explain how to generate DTD.

4.2 Generation of depth transition data

DTD is generated for each pixel location by tracing its depth map value change. For simplification only transition from foreground to background or vice versa is recorded instead of exact depth map value change. However, this can be easily extended to more number of depth layers.

First, it is necessary to set a criterion for depth layer determination for the reference views. There have been various researches on object separation, e.g. using video motion information, segmentation based method, etc. In this work, as a preliminary approach the depth map is used for the separation. For example, the depth map value range can be divided according to the number of depth layers represented by DTD. If DTD represents two depth layers, i.e., foreground and background, a middle point of the depth map value range, i.e. the average of maximum and minimum depth map values can be used as a threshold value, and for each pixel location, if its depth map value is smaller than the threshold, it will belong to the background, and vice versa. The threshold value can be adjusted for better separation of depth layers. Note that more advanced algorithms will help to improve the performance of our proposed method.

Next, if the depth maps for the intermediate views between reference views are available at the encoder side, they can be used to generate the depth transition data by thresholding the depth map values and generating the binary map using the same scheme applied to the reference views. Then, it is easy and precise to trace the transition in this case.

However, the depth maps are not always available for the target view at an arbitrary view position. Therefore, we derive how to estimate the camera position where the depth transition happens using the camera parameters. Refer to Table 3.1 for the notations used in this section. As shown in Section 3.1.1, camera coordinates (x, y, z) can be mapped into the world coordinates (X, Y, Z) using (3.1), and the image coordinates (x_{im}, y_{im}) can be expressed from the camera coordinates as in (3.2). Then, we can derive the point mapping from the reference view to the target view using camera parameters. Specifically, if we map a point in the p-th view of which the camera parameters are $\mathbf{A}_p$, $\mathbf{R}_p$, and $\mathbf{T}_p$, to the p'-th view with parameters of $\mathbf{A}_{p'}$, $\mathbf{R}_{p'}$, and $\mathbf{T}_{p'}$, the camera coordinates in the p'-th view can be represented as in (3.3). Then, the image coordinates in the p'-th view can be represented as in (3.4).

Now, based on the derivation of point mapping, we show how to calculate the camera position at which depth transition happens. We assume that the cameras are arranged in a horizontally parallel position, which implies that $\mathbf{R}_{p'} = \mathbf{R}_p^{-1} = \mathbf{R}_{p'}\mathbf{R}_p^{-1} = \mathbf{I}$ (identity matrix). To calculate $\mathbf{A}_{p'}\mathbf{A}_p^{-1}$, we define the intrinsic matrix $\mathbf{A}$ as

$$\mathbf{A} = \begin{pmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{4.1}$$

where f_x and f_y are the focal length divided by the effective pixel size in horizontal and vertical direction, respectively, and (o_x, o_y) is the coordinates in pixel of the image center (the principal point). Then, $\mathbf{A}^{-1}$ can be calculated as

$$\mathbf{A}^{-1} = \begin{pmatrix} 1/f_x & 0 & -o_x/f_x \\ 0 & 1/f_y & -o_y/f_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{4.2}$$

Therefore, if we assume the same focal length for both cameras at p-th and p'-th views, (3.4) will become

$$\begin{aligned} \begin{pmatrix} x'_{im} \\ y'_{im} \\ 1 \end{pmatrix} &= \begin{pmatrix} \frac{x'}{z'} \\ \frac{y'}{z'} \\ \frac{z'}{z'} \end{pmatrix} = \mathbf{A}_{p'}\mathbf{A}_p^{-1} \begin{pmatrix} x_{im} \\ y_{im} \\ 1 \end{pmatrix} + \frac{1}{z'}\mathbf{A}_{p'} \left\{ \mathbf{T}_p - \mathbf{T}_{p'} \right\} \\ &= \begin{pmatrix} x_{im} + o_{x,p} - o_{x,p'} \\ y_{im} + o_{y,p} - o_{y,p'} \\ 1 \end{pmatrix} + \frac{1}{z'}\mathbf{A}_{p'} \left\{ \mathbf{T}_p - \mathbf{T}_{p'} \right\}. \end{aligned} \tag{4.3}$$

Under the parallel camera setting assumption, there will be no disparity change other than in horizontal or x-direction. Therefore, the disparity Δx_{im} can be expressed as

$$\Delta x_{im} = x'_{im} - x_{im} = o_{x,p} - o_{x,p'} + \frac{1}{z'} \cdot f_x \cdot t_x, \tag{4.4}$$

where t_x indicates the camera distance in horizontal direction. The relationship between the actual depth value and 8-bit depth map value is given in (2.1). By plugging this into (4.4), we can get

$$\begin{aligned}\Delta x_{im} &= x'_{im} - x_{im} \\ &= o_{x,p} - o_{x,p'} + \left(\frac{L_p\left(x_{im}, y_{im}\right)}{255} \cdot \left(\frac{1}{Z_{near}} - \frac{1}{Z_{far}} \right) + \frac{1}{Z_{far}} \right) \cdot f_x \cdot t_x.\end{aligned} \tag{4.5}$$

Therefore, if we know the camera distance, t_x, we can calculate the disparity, Δx_{im}, and vice-versa.

To find the exact view position where transition happens, we set the disparity as the horizontal distance from the given pixel location to where depth transition happens. The horizontal distance is the number of pixels from the given pixel to the first pixel of which the depth map value difference with respect to the original pixel exceeds a preset threshold value. Then using this distance as the disparity, Δx_{im}, we can estimate the view position at which depth transition occurs as:

$$t_x = \frac{\Delta x_{im} + o_{x,p'} - o_{x,p}}{f_x} \cdot \frac{255}{a \cdot L_p\left(x_{im}, y_{im}\right) + 255b}, \tag{4.6}$$

where $a = \frac{1}{Z_{near}} - \frac{1}{Z_{far}}$ and $b = \frac{1}{Z_{far}}$. Then, t_x can be quantized to the desired precision and transmitted as auxiliary data.

4.3 Rate-distortion analysis of depth transition data

The performance of DTD depends on various factors. If the rendered view quality using depth map without DTD is very high, there would be not so much room for improvement when DTD is applied. Therefore, the performance of DTD is closely related to factors affecting the rendered view quality such as baseline distance, intensity contrast between objects at different depths, depth range in a scene, foreground object size, etc. For example if the baseline distance is large and the object is close to the camera, there will be large amount of occlusion which will result in low rendered view quality. Also, if there is high contrast between the intensity values of neighboring objects at different depths, a small amount of depth map error will cause significant distortion in the rendered view. The performance also depends on bitrate to code DTD, as compared to overall bitrate. In this section we analyze the performance of DTD considering these factors to find an optimal way to utilize DTD.

DTD is used in combination with a depth map of a neighboring view. The RD performance of DTD can be compared to that achieved when using the depth map only case. The depth map is quantized using quantization step size, Q, and DTD is losslessly coded. The RD performance can be compared by computing the difference between the Lagrangian costs achieved without and with DTD, L_{DM} and $L_{\mathrm{DM+DTD}}$, respectively, as

$$\begin{aligned} L_{\text{DM}} &= D(Q) + \lambda R(Q) \\ L_{\text{DM+DTD}} &= D(Q) - \delta + \lambda\left(R(Q) + R_{\text{DTD}}\right) \\ L_{\text{DM}} - L_{\text{DM+DTD}} &= \delta - \lambda R_{\text{DTD}}, \end{aligned} \tag{4.7}$$

where $D(Q)$ is the rendered view distortion with the depth map quantized using Q, $R(Q)$ is the bitrate to code the quantized depth map, δ is the amount of distortion reduced by applying DTD, and R_{DTD} is the bitrate to code DTD.

Note that this comparison can be made for various size of units according to user's intention, such as a block, macroblock, slice, frame, group of pictures, and the whole sequence. The variables used hereinafter stand for a representative value for the chosen size of unit, e.g. an average over the unit.

Now, we model δ as the size of corrected area multiplied by the square of corrected intensity value, which can be represented as

$$\delta = \Delta P \cdot S \cdot l_{\text{FB}}^2 \tag{4.8}$$

where ΔP is the amount of transition caused by depth map error, S is the boundary size of the foreground object, and l_{FB}^2 is the square of difference between foreground and background intensity level, $l(\text{F})$ and $l(\text{B})$, respectively, as

$$l_{\text{FB}}^2 = |l(\text{F}) - l(\text{B})|^2 . \tag{4.9}$$

ΔP is the amount of transition occurred by depth map distortion, D_{depth}, and can be calculated using the camera parameters under the parallel camera setting assumption as

$$\begin{aligned}\Delta P &= \frac{1}{255}\left(\frac{1}{Z_{\text{near}}}-\frac{1}{Z_{\text{far}}}\right) f_x t_x D_{\text{depth}} \\ &= kD_{\text{depth}}, \end{aligned} \tag{4.10}$$

where Z_{near} and Z_{far} is the nearest and farthest depth value in the scene, respectively, f_x is the focal length divided by the effective pixel size in horizontal direction, and t_x is the camera distance in horizontal direction. Note that D_{depth} is absolute difference and can be represented as a function of quantization step size, Q, under the uniform distribution assumption as

$$\begin{aligned} D_{\text{depth}} &= \frac{1}{Q}\int_{-\frac{Q}{2}}^{\frac{Q}{2}} |x|\, dx \\ &= \frac{Q}{4}. \end{aligned} \tag{4.11}$$

Since different sequences can have different statistical properties and result in different amounts of distortion due to quantization, we represent this as

$$D_{\text{depth}} = c_1 Q, \tag{4.12}$$

where c_1 is a scaling factor, which can be set as $1/4$ as in (4.11), or can be found using a linear regression for better fitting for the characteristics of the given sequence. Now, the boundary size S can be represented as a number of pixels as

$$S = \alpha N \tag{4.13}$$

where N is the total number of pixel to be coded, and α is the portion the boundary pixel occupies ($0 \leq \alpha \leq 1$). Therefore, δ can be represented as

$$\delta = k \cdot c_1 \cdot Q \cdot \alpha \cdot N \cdot l_{\mathrm{FB}}^2. \tag{4.14}$$

Next, we model the bitrate to code DTD, R_{DTD}. This can be modeled as a multiplication between the area where DTD is provided and the per-pixel bitrate of DTD. Since DTD is provided where the transition happens between two neighboring views, the size of area where DTD is provided is directly proportional to the disparity size, which is a function of camera parameters such as baseline distance, and depth value. And, it is also proportional to the size of the foreground object given in (4.13). The disparity between two views at depth Z can be calculated as in (4.4). By considering both foreground and background movement with assumption of $o_{x,p} = o_{x,p'}$, the translated foreground area can be calculated as

$$\begin{aligned}\Delta x_{\mathrm{im}} &= \left(\frac{1}{Z_{\mathrm{near}}} - \frac{1}{Z_{\mathrm{far}}}\right) \cdot f_x \cdot t_x \\ &= 255k. \end{aligned} \tag{4.15}$$

Then, R_{DTD} becomes

$$\begin{aligned} R_{\mathrm{DTD}} &= \Delta x_{\mathrm{im}} \cdot \alpha \cdot N \cdot r_{\mathrm{DTD}} \\ &= 255k \cdot \alpha \cdot N \cdot r_{\mathrm{DTD}}, \end{aligned} \tag{4.16}$$

where r_{DTD} is the per-pixel bitrate to code DTD, which depends on entropy coding method such as arithmetic coding, Huffman coding, etc.

Now, if these results are put into (4.7), it becomes

$$\delta - \lambda R_{\text{DTD}} = k\alpha N(c_1 Q l_{\text{FB}}^2 - 255\lambda r_{\text{DTD}}). \tag{4.17}$$

This reveals that if the intensity contrast between the foreground and background is large enough, we can achieve coding efficiency improvement by providing DTD. This gain scales up linearly with k and the boundary size, αN.

If we consider the λ factor according to our previous model in (3.26),

$$\lambda(Q) = \frac{\sigma_{\text{video}}^2}{\sigma_{\text{DM}}^2}\frac{c_1}{c_2}k \cdot \ln\rho_1 \cdot \rho_1^{kc_1Q} \cdot Q^3, \tag{4.18}$$

then, (4.17) becomes

$$\delta - \lambda R_{\text{DTD}} = k\alpha N Q(c_1 l_{\text{FB}}^2 + 255\frac{\sigma_{\text{video}}^2}{\sigma_{\text{DM}}^2}\frac{c_1}{c_2}k\ln\rho_1\rho_1^{kc_1Q}Q^2 r_{\text{DTD}}). \tag{4.19}$$

Now, if we consider the maximum video variance, which can be achieved when there are equal number of foreground and background pixels with a difference between them equal to l_{FB}, σ_{video}^2 can be expressed in terms of l_{FB}^2 as

$$\sigma_{\text{video}}^2 = \frac{1}{4}l_{\text{FB}}^2. \tag{4.20}$$

In addition, in this case the first order correlation coefficient can be calculated by calculating the covariance as

$$\begin{aligned}
\text{cov}(x_m, x_n) &= \frac{1}{N}\sum_i (x_{m,i} - \mu)(x_{n,i} - \mu) \\
&= \frac{1}{N}\left\{(1-\alpha)N\frac{(l(\text{F}) - \text{l(B)})^2}{4} + \alpha N(-\frac{(l(\text{F}) - \text{l(B)})^2}{4}\right\} \\
&= (1-2\alpha)\sigma^2_{\text{video}}, \qquad (4.21)
\end{aligned}$$

and then,

$$\rho_1 = \frac{\text{cov}(x_m, x_n)}{\sigma^2_{\text{video}}} = 1 - 2\alpha. \qquad (4.22)$$

Then, (4.19) becomes

$$\delta - \lambda R_{\text{DTD}} = k\alpha N Q c_1 l^2_{\text{FB}}(1 + \frac{255}{4}\frac{1}{\sigma^2_{\text{DM}}}\frac{1}{c_2}k\text{ln}\rho_1\rho_1^{kc_1Q}Q^2 r_{\text{DTD}}), \qquad (4.23)$$

which leads to positive gain when

$$\frac{255}{4}\frac{1}{\sigma^2_{\text{DM}}}\frac{1}{c_2}k\text{ln}\rho_1\rho_1^{kc_1Q}Q^2 r_{\text{DTD}} > -1. \qquad (4.24)$$

Now we denote (4.23) as a gain function, G(Q,k). The gain functions are plotted by varying these parameters, Q and k in addition to l_{FB}. Other values are set using some practical values as $c_1 = 1/4, c_2 = 1, \sigma^2_{\text{DM}} = \sigma^2_{\text{video}}/2, \alpha = 1/16, \rho_1 = 1-2\alpha, r_{\text{DTD}} = 1/5$, and $N = 1280 \times 720$. Fig. 4.2 shows 2-D mesh plot of the gain function in terms of Q and k.

Fig. 4.3 shows the curve of G(Q) when l_{FB} is 50 and 80, with different values of k, which reveals the relationship between the quantization step size and the gain. As Q increases the coding gain can increase as there is larger room to reduce the

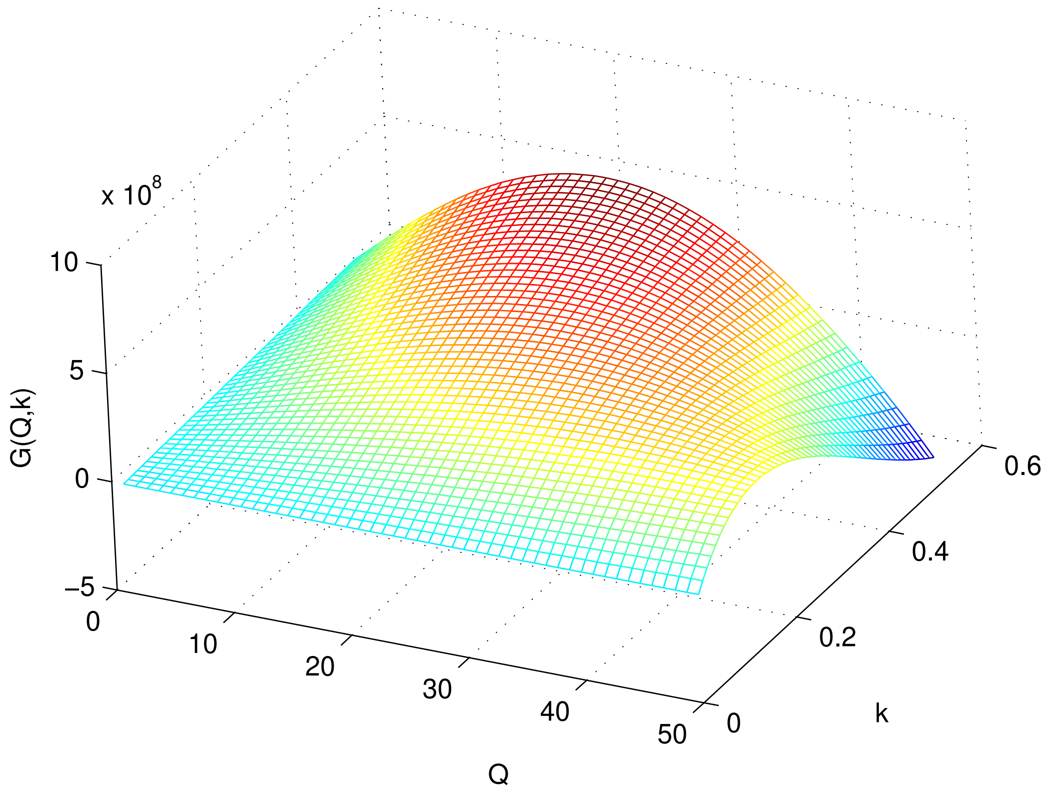
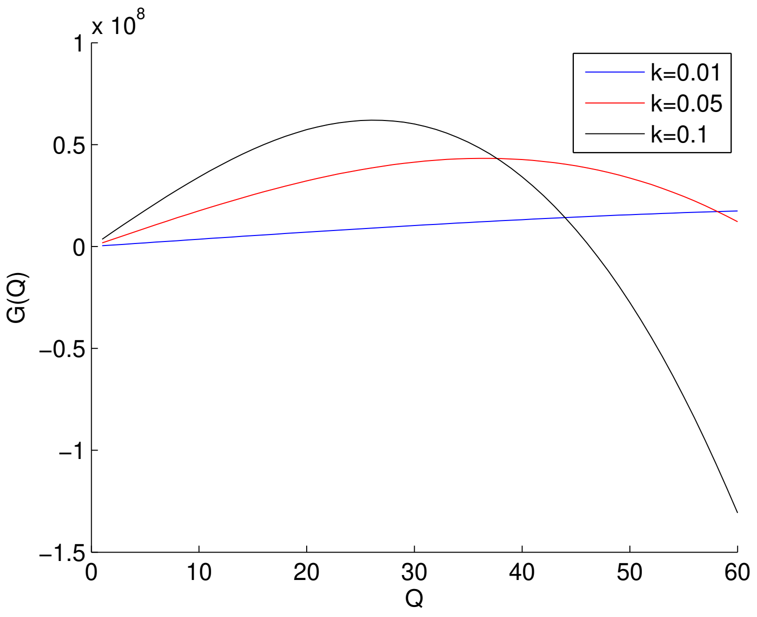


Figure 4.2: Mesh plot of the DTD coding gain function in terms of k and quantization step size, $\mathrm{G}(k, Q)$, when $l_{\mathrm{FB}} = 80$.

distortion using DTD. However, beyond certain point, coding gain decreases due to the bitrate to code DTD, since it is losslessly compressed and not scalable to Q, while depth map bitrate reduces as Q increases. Fig. 4.3 also shows how the contrast between the foreground and background affects the performance. Large contrast, l_{FB} leads to large magnitude of gain, and at the same time leads to a larger range of Q for which gain can be achieved. Fig. 4.3 also shows different value of k makes the maximum coding gain be achieved at different Q.

Fig. 4.4 shows the curve of $\mathrm{G}(k)$ with different values of Q. This shows that when Q is not too large, the coding gain scales up with k. This is because the



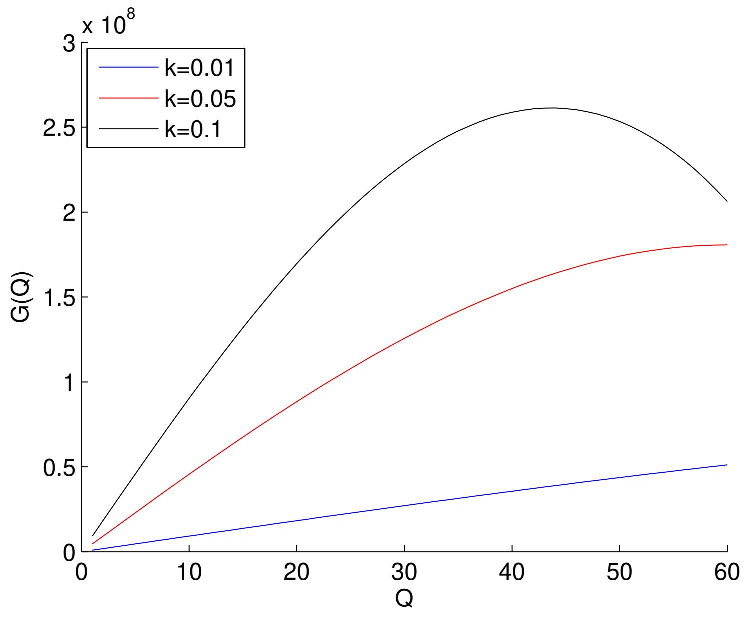


Figure 4.3: Plot of the DTD coding gain function in terms of quantization step size, G(Q), with different values of k and l_{FB}; top: $l_{\mathrm{FB}} = 50$, bottom: $l_{\mathrm{FB}} = 80$.

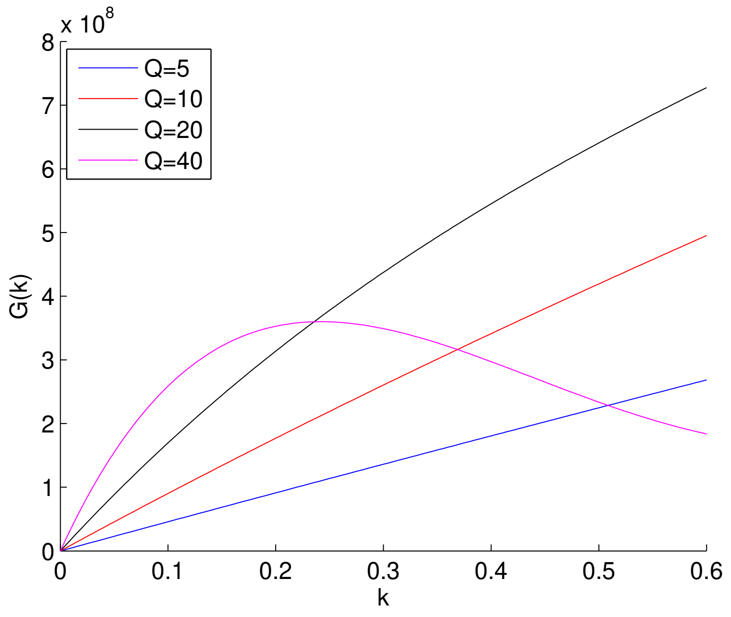


Figure 4.4: Plot of the DTD coding gain function in terms of k, G(k), with different values of Q.

amount of occlusion scales up with k, which means the amount of distortion reduction using DTD also scales up with k. However, as the bitrate to code DTD scales up with k, too, the gain decreases when both k and Q are large.

From these analyses it is possible to choose optimal parameters to get the maximum coding gain. For example it would be possible to adjust camera settings to adjust k by changing baseline distance, focal length, and/or distance to the objects in the scene. However, in many cases it is possible that the multiview sequence with depth map is just given with fixed camera parameters. Then, it will be possible to choose the depth map quantization step size using the RD analysis above given certain camera parameters which determine k.

Fig. 4.5 depicts the experimental results using the Mobile sequence. In the RD curves, the PSNR is calculated using the synthesized view compared to the original view, where the view synthesis is performed using decoded depth maps.

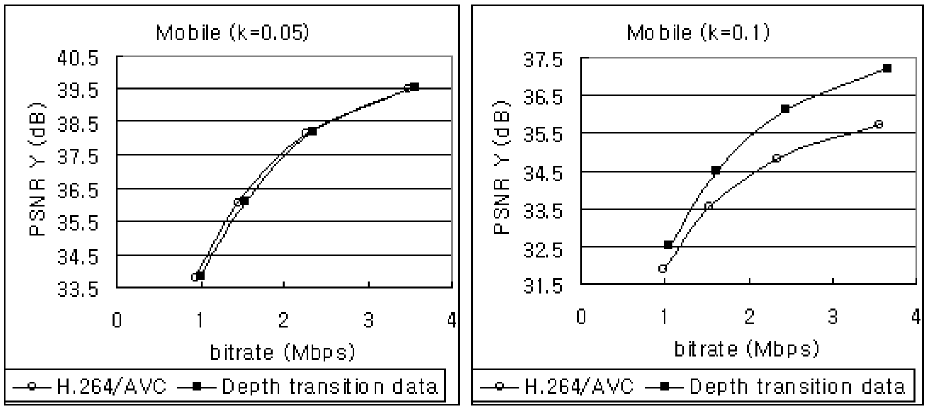


Figure 4.5: RD curves generated using different k values for Mobile sequence.

The bitrate is the total bitrate including DTD bitrate when DTD is applied during the view synthesis as described in Section 4.4.2. Among various parameters those that can be easily controlled at the encoder side are the baseline distance and the quantization step size. Therefore, the RD curves are generated using variation of these parameters. When $k = 0.1$, it can be noticed that the coding gain (the gap between two RD curves) increases as quantization step size decreases. This gain vanishes when k is reduced to 0.05. Thus, the result matches with the theoretical analysis.

4.4 Application of depth transition data

4.4.1 Correcting erosion artifact

We now describe a procedure to correct erosion artifacts described in Section 2.3 using the proposed DTD. To render a view at an arbitrary view point, we can use DTD to determine whether each pixel belongs to the foreground or background.

For example, at a specific pixel location, if it belongs to the foreground in the left reference and to the background in the right reference, we know that the depth transition happened between these two views. If DTD indicates that this transition happens before the target view to be rendered, the pixel location would belong to the background in the target view. If DTD indicates the transition happens after the target view, the pixel location would belong to the foreground in the target view. When the transition happens in the other way, i.e., from background to foreground, this decision will be reversed. We assume that DTD records the first transition happened between the two reference views.

Once the foreground/background map for the target view is generated using DTD, the erosion correction can be performed for a given local area, e.g., 8×8 block in the rendered view with erosion artifact. Each area can overlap and/or have adaptive size for better performance. First, for the given block a background average is calculated using the pixels belonging to the background. Then, each foreground pixel is compared to the background average. If a pixel is close to the background average, it is classified as an outlier or eroded pixel. Then, foreground average is calculated using the foreground pixels without outliers. Finally, the eroded pixel values are replaced with the foreground average. To replace the eroded pixel value, it would be also possible to use the nearest foreground pixel value which is not an outlier, or utilize the pixel values in the reference video. We will further investigate how to improve this procedure as a future work. Fig. 4.6 shows the flow chart of the proposed erosion correction process.

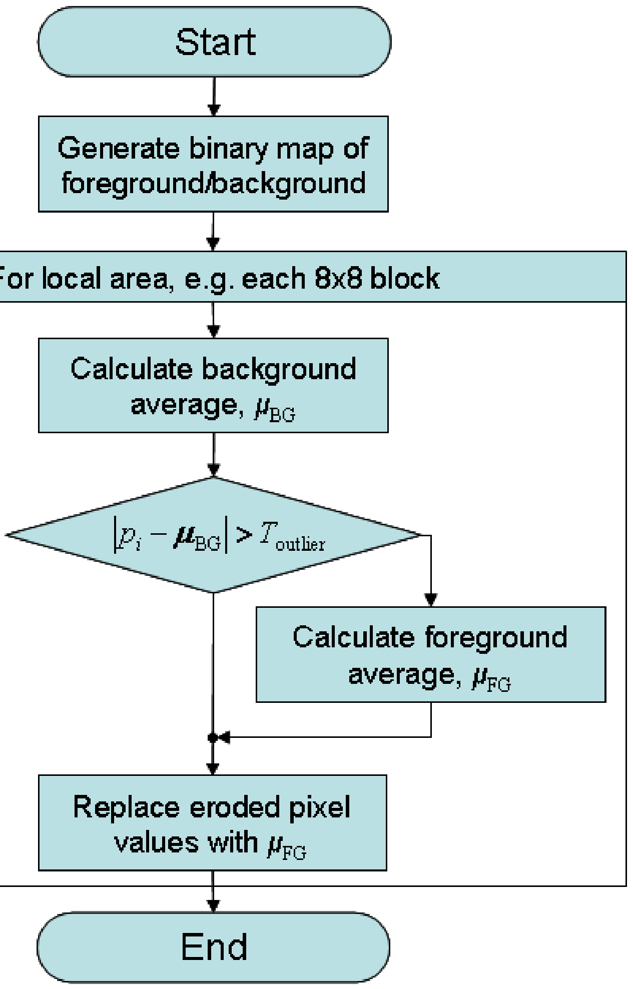

Start
Generate binary map of foreground/background
For local area, e.g. each 8x8 block
Calculate background average, μ_{BG}
$|p_i - \boldsymbol{\mu}_{BG}| > T_{outlier}$
Calculate foreground average, μ_{FG}
Replace eroded pixel values with μ_{FG}
End

Figure 4.6: Flow chart for the proposed erosion artifact correction method using the depth transition data as in Section 4.4.1

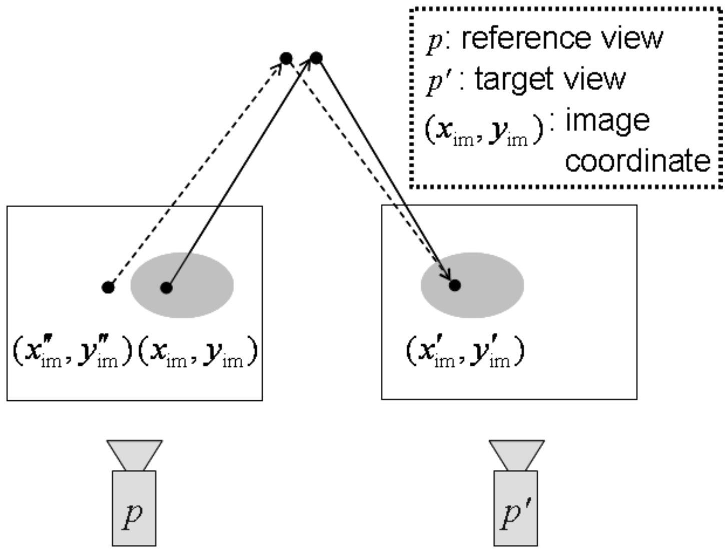

p: reference view
p': target view
$(x_{\text{im}}, y_{\text{im}})$: image coordinate
$(x''_{\text{im}}, y''_{\text{im}})$
$(x_{\text{im}}, y_{\text{im}})$
$(x'_{\text{im}}, y'_{\text{im}})$
p
p'

Figure 4.7: Illustration of view synthesis process with possible depth error.

4.4.2 View synthesis using depth transition data

In the previous section, DTD is used to correct erosion artifacts that occur in the rendered view, by using a post-processing after view rendering is completed. Since this is done as a post-processing, DTD is not fully utilized to correct the distortion. For example, if the erosion is large, the inpainted region would not look natural. Also it cannot fix distortions other than the erosion along object boundaries. This can be improved by directly applying DTD during the view synthesis process rather than correcting the distortion after the rendering process.

Fig. 4.7 illustrates the view synthesis process for a specific pixel location. When a reference pixel is warped into the target view, it is possible that a wrong pixel will be mapped to the target due to the depth map distortion (e.g., $(x''_{\text{im}}, y''_{\text{im}})$ is used for interpolation instead of $(x_{\text{im}}, y_{\text{im}})$ as in Fig. 4.7).

The non-linear nature of the rendered view distortion discussed in Section 2.4 can be understood from the example in Fig. 4.7: significant errors are produced if the chosen reference pixel belongs to a different object layer, e.g., a background pixel is used as reference for a pixel belonging to a foreground object. Thus similar depth errors may lead to very different errors in interpolation.

DTD information allows us to signal explicitly that a specific pixel belongs to one of the two possible depth layers (corresponding to left and right images). Thus, in the example of Fig. 4.7 DTD will indicate that the intermediate pixel belongs to a foreground object.

Without knowing the true depth map value at the decoder side due to quantization, it is impossible to verify whether the reference is correct or not. However, by using DTD we can verify whether it belongs to the same object layer as the target pixel.

As described in Section 4.4.1, an object layer map can be generated from DTD for any intermediate view, indicating which object layer each pixel belongs to in the target view. The object layer can be formed according to the depth level of different objects. In the simulation, two object layers are used, which indicate foreground and background level. Note that this approach operates independently of view synthesis algorithms trying to generate depth layers in the reference views without providing additional information, since the proposed method generates the layer map in the target view by wisely spending additional bits to provide this to many intermediate views.

A view synthesis is performed by warping reference pixels to the target view. When there is more than one reference pixel mapped to the same position in the target view, the synthesis can be done by a blending process. The blending process can be a weighted averaging using the distance between the reference and the target views as a weight. Or it is also possible to use the pixel which is nearer to the camera. If there is only one reference pixel mapped to the target view pixel position, its value can be directly copied to the target. If no reference pixel is mapped to a target pixel position, that location is regarded as a hole, and can be filled using an in-painting scheme.

Now, we describe how DTD is applied for each step of view synthesis process. Fig. 4.8 provides the flowchart for this procedure.

When a pixel in the target view is synthesized, first the availability of DTD for that pixel location is checked. If available, by using DTD it is determined which object layer the pixel belongs to. Then, the pixels in the reference views are warped to the target view.

When a pixel is warped, we can know which object layer it belongs to in the reference view using its corresponding depth value. Therefore, it can be checked whether this warped pixel from the reference view (reference pixel) belongs to the same object layer as that of the target view pixel. If the reference pixels belong to the same object layer as the target, they are regarded as valid and used for the view synthesis. Conversely, if the reference pixel does not belong to the same object layer as the target pixel, this implies it is not safe to use this reference pixel for the view synthesis.

Note that there can be some pixel positions in the target view, for which no reference pixel is mapped from the reference views. In this case, it is allowed to use the reference pixels in different object layer only when its corresponding depth value is close to that of the other reference view. This is the case where the transition has happened but with a small amount of depth change, which can be set as a threshold value to indicate the tolerance range with a unit of depth map value. In our simulation, a fixed threshold value is applied for all the sequences to simplify the simulation; however, it would be possible to adapt the threshold value according to global and local sequence characteristics such as camera parameters, contrast in local area, etc., to improve the performance. Note that in this case DTD does not

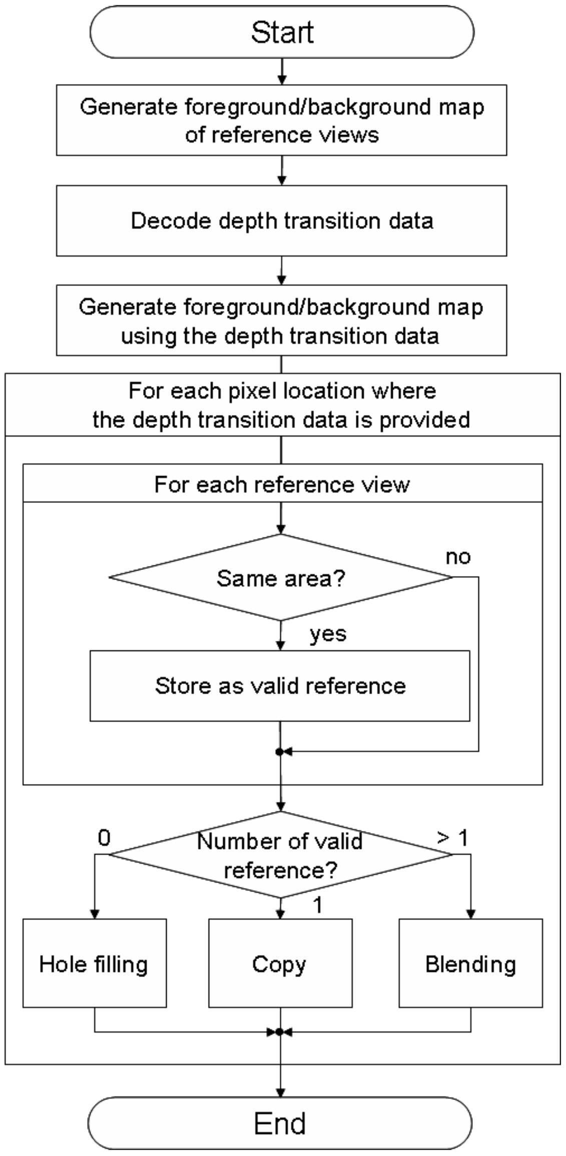

Start
Generate foreground/background map of reference views
Decode depth transition data
Generate foreground/background map using the depth transition data
For each pixel location where the depth transition data is provided
For each reference view
Same area?
no
yes
Store as valid reference
Number of valid reference?
0
> 1
1
Hole filling
Copy
Blending
End

Figure 4.8: Flowchart of the view synthesis procedure using the depth transition data of Section 4.4.2.

need to be provided, therefore, we do not need to code and transmit it, but the encoder and decoder should use the same threshold value.

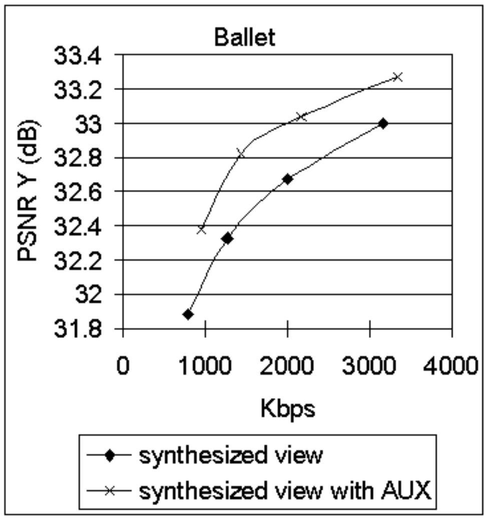
4.5 Experimental results

4.5.1 Erosion artifact correction

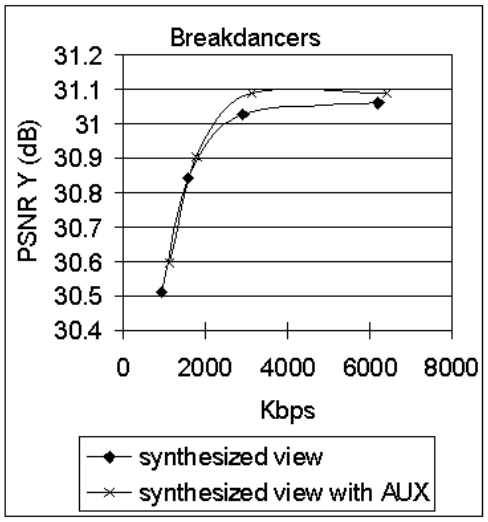
To evaluate the proposed DTD and its application to erosion artifact correction as described in Section 4.4.1, experiments are performed using Ballet and Breakdancers test sequences [83]. Fig. 4.9 shows the RD curve comparison where the 5th view is synthesized using 4th and 6th views as references. The reference video and depth maps are coded using H.264 with same QP values of 24, 28, 32, and 36 to generate the curve, and Y PSNR is calculated by comparing the synthesized view to the original 5th view. For 'synthesized view with AUX', the same synthesis is performed followed by the erosion correction using the depth transition data described in Section 4.4.1. For the erosion artifact correction, block size of 8×8 is used without overlapping.

The bitrate to code the depth transition data is approximately estimated by coding the foreground/background binary map of the target view, and added to the bitrate to code two sets of reference video and depth map. It may be possible to reduce the bitrate required to encode DTD by exploiting the reference depth map information. This will be further investigated and implemented as a future work.

From the curves, we can notice that there are moderate bitrate increases due to DTD, while there are noticeable PSNR gains for the Ballet sequence, and smaller



(a) Ballet



(b) Breakdancers

Figure 4.9: Rate-distortion curves with and without depth transition data applied for erosion artifact correction in Section 4.4.1.

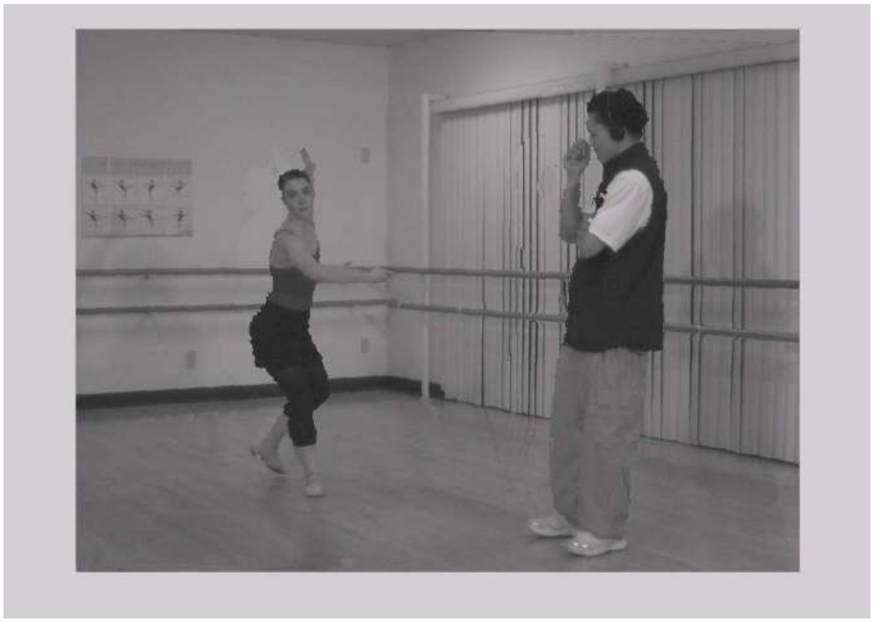


Figure 4.10: Synthesized Ballet sequence with depth map compressed using H.264/AVC without DTD applied to correct erosion artifact.

gains for the Breakdancers sequence. The reason for the smaller gain in the Breakdancers sequence is that some objects in the scene are close to the background in depth, so that not many portions of the scene belong to the foreground. In addition, the difference between the foreground and background pixel values is not as significant as compared to the Ballet sequence, which reduces the PSNR gain.

Figs. 4.10 and 4.11 show the subjective quality comparison with and without the proposed method. In Fig. 4.10 the erosion artifact is clearly visible which degrades the subjective quality greatly, and in Fig. 4.11 it can be noticed that the eroded area is fixed using the proposed method, thus noticeable subjective quality improvement is achieved.

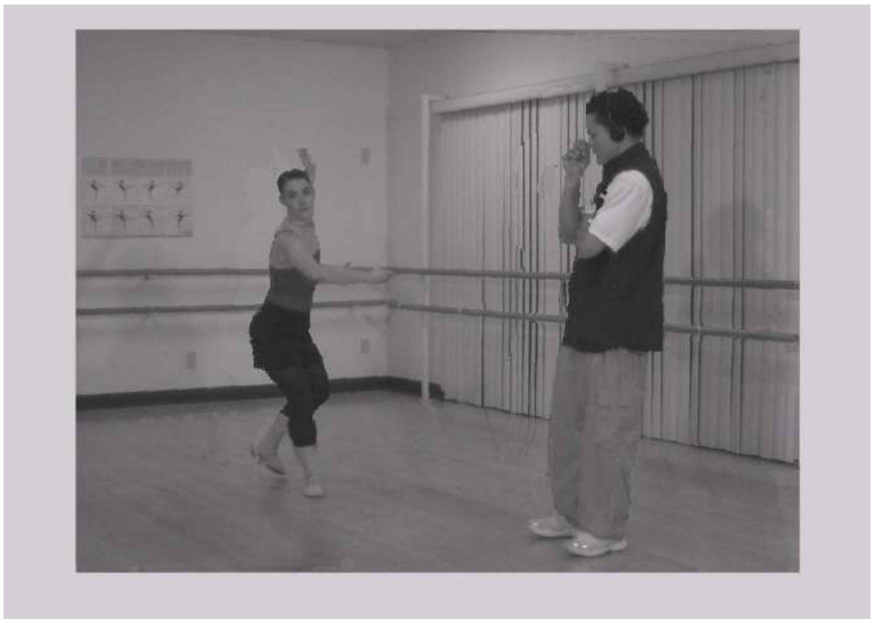


Figure 4.11: Synthesized Ballet sequence with the erosion artifact correction using the depth transition data as in Section 4.4.1

4.5.2 View synthesis using depth transition data of Section 4.4.2

The proposed method is evaluated with experiments using various test sequences. In this section, the test procedure is described and the results are discussed. Table 4.1 lists the test sequences used in the experiments with the view indices of left and right reference views and the target view. $d_{x,\mathrm{im}}$ in (4.4) is also calculated with setting $o_{x,p} = o_{x,p'}$ and $z = Z_{\mathrm{near}}$ for each sequence, and included in Table 4.1 to compare different camera settings among various sequences.

Table 4.1: View index of test sequences used for view synthesis using DTD of Section 4.4.2.

Sequence	Left	Target	Right	$d_{x,\mathrm{im}}$
Cafe	1	3	5	180.0
Mobile	3	5	7	43.5
Champagne Tower	37	39	41	146.1
Balloons	1	3	5	50.0
Newspaper	2	4	6	100.0
Pantomime	37	39	41	76.2
Lovebird1	4	6	8	67.1
Kendo	1	3	5	50.0

DTD is calculated using the left and right reference views first by generating their object layer maps with two levels (foreground and background) followed by calculating the transition position using (4.6) for the region where transition happened between the reference views. The foreground/background determination is performed based on the depth map value using k-means clustering. The fixed threshold value of the tolerance range described in Section 4.4.2 is set to 30 for all the sequences. For the area where transition happens more than once between the reference views, only the first transition position is recorded. Then, it is quantized to represent the mid-position between the reference views, and losslessly compressed using H.264 with CABAC encoding.

The proposed method to apply DTD to view synthesis is implemented using the MPEG view synthesis reference software (VSRS) 3.0 [58]. Fig. 4.12 shows the RD curves with and without DTD, where both depth map and video are compressed using H.264 with QP values of 24, 28, 32, and 36. Inter-view prediction is used to code both video and depth maps to reduce the bitrate. Two reference views are coded and the middle point view between them is synthesized. The PSNR of the rendered view is calculated compared to the ground truth. The x-axis is the bitrate to code reference video and depth maps. In case of "Depth transition data", the bitrate to code DTD is added. The y-axis is the PSNR of Y component.

In Fig. 4.12, it can be observed maximum PSNR improvement of 2 dB in Champagne Tower, and 1.5, 0.6, and 0.5 dB in Mobile, Cafe, and Balloons, respectively. Also there are smaller PSNR improvements in the Newspaper and Pantomime sequences. No gain is observed in the Lovebird1 and Kendo sequences.

For Cafe and Champagne Tower, large value of $d_{x,\mathrm{im}}$ contributes to the improvement, and large size of foreground objects in Champagne Tower increases the performance improvement. For the Mobile sequence, high contrast across the object boundary contributes to the improvement. In case of Lovebird1, the foreground object is small and the contrast is not strong, so no improvement is observed. For the Kendo sequence, the depth map is noisy, so it does not provide clear object boundary. In this case, it would be possible to improve the performance by generating DTD using other information such as reference video. We leave this as a future study.

Counter intuitively, in the RD curves of Pantomime and Champagne tower, PSNR decrement can be observed even though the bitrate is increased. This is

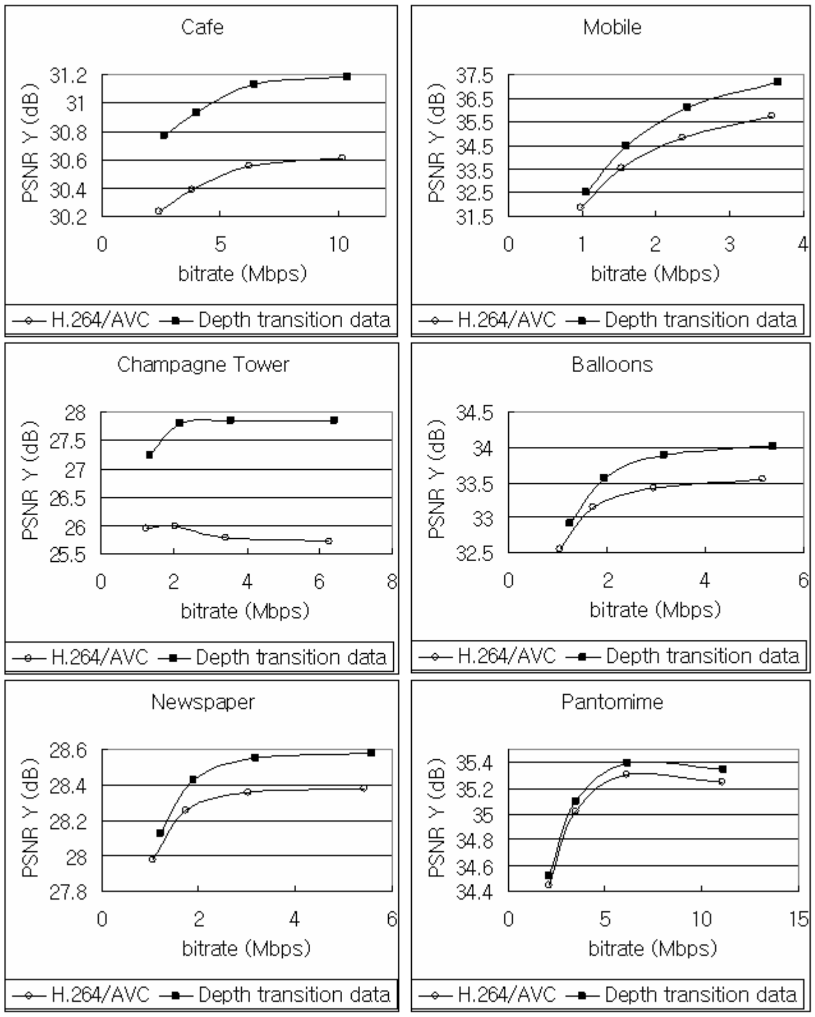


Figure 4.12: Rate-distortion curves with and without depth transition data applied to view synthesis process as in Section 4.4.2.

because the original depth map contains noise. Even though the bitrate is increased by coding more accurately the depth map (including noise), this does not help to increase PSNR. If the rendered view without compression is used as an anchor to calculate PSNR, this phenomenon will not occur. However, we used the ground truth video as the anchor, so that PSNR can represent comparison between the original video and the rendered view including various distortions described in Section 2.2.

In all the sequences, the bitrate is increased a little for the DTD case due to the bitrate to code DTD. Table 4.2 shows the relative percentage of the total bitrate used for DTD. Since it is coded losslessly, the percentage of rate used for DTD increases as QP increases.

Table 4.2: Bitrate occupancy of video, depth map, and DTD in total bitrate for two reference views.

Sequence	QP	Bitrate distribution (%)		
		Video	Depth map	DTD
Cafe	24	58	37	2
	28	53	39	4
	32	51	37	6
	36	49	33	9
Mobile	24	84	14	2
	28	80	16	3
	32	77	18	5
	36	72	20	8
Champagne Tower	24	75	23	2
	28	69	27	4
	32	66	28	6
	36	63	28	9

Figs. 4.13 to 4.15 show the subjective quality comparison between the rendered view without and with DTD. In Cafe, it can be noticed the wrong pixel mapping in man's hair area is corrected using the proposed method. Again, in Mobile

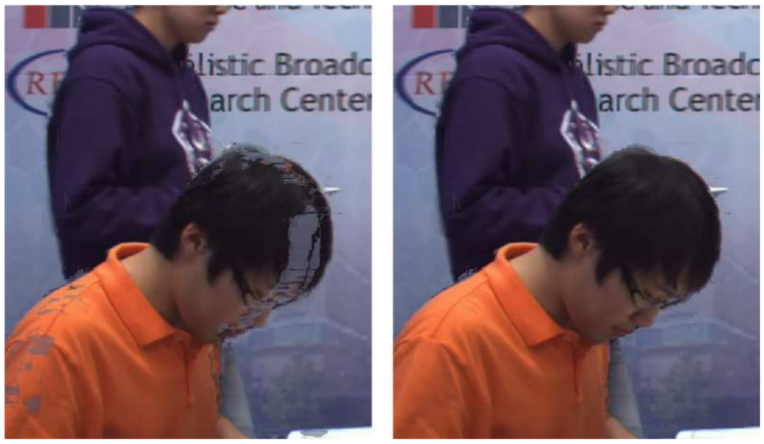


Figure 4.13: Subjective quality comparison: conventional method (left) and the proposed method with depth transition data applied to view synthesis process as in Section 4.4.2 (right); Cafe (QP 24).

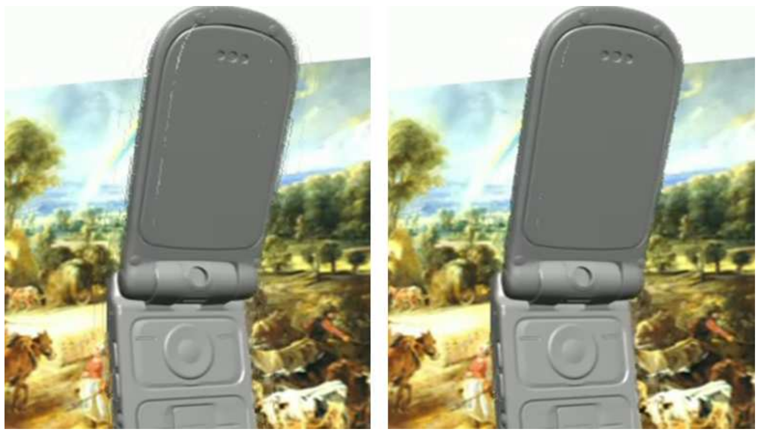


Figure 4.14: Subjective quality comparison: conventional method (left) and the proposed method with depth transition data applied to view synthesis process as in Section 4.4.2 (right); Mobile (QP 24).

and Champagne Tower, wrong reference pixel mapping in the foreground object is corrected, and clear object boundary can be observed.

Table 4.3 shows PSNR (dB) and N-PSNR (dB) results of various test sequences. N-PSNR is defined in Section 2.5, where threshold value of 10 is used. PSNR and

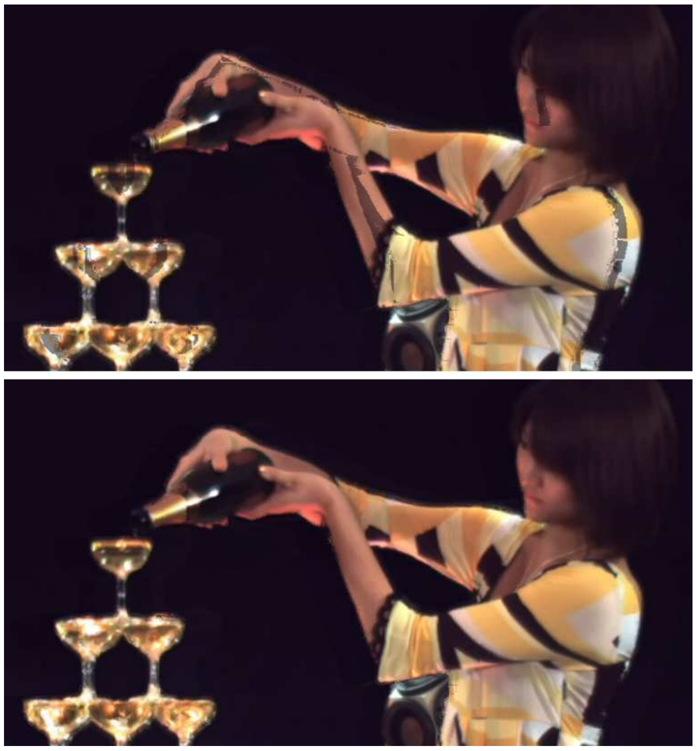


Figure 4.15: Subjective quality comparison: conventional method (top) and the proposed method with depth transition data applied to view synthesis process as in Section 4.4.2 (bottom); Champagne Tower (QP 24).

Table 4.3: PSNR (dB) and N-PSNR (dB) results with DTD, generated by comparing the luminance (Y) component of the ground truth and the rendered view using various test sequences, where both video and depth map are compressed using H.264/AVC with QP set to 24 and 36. Threshold value of 10 is used for N-PSNR. Difference is generated compared to PSNR and N-PSNR without DTD.

Sequence	QP	with DTD		Difference	
		PSNR	N-PSNR	PSNR	N-PSNR
Cafe	24	31.1	31.8	0.6	0.7
	36	30.7	31.5	0.5	0.6
Mobile	24	37.2	39.9	1.4	2.4
	36	32.5	34.4	0.7	1.0
Newspaper	24	28.6	29.7	0.2	0.3
	36	28.1	29.2	0.2	0.2
Balloons	24	34.0	35.8	0.5	0.7
	36	32.9	34.7	0.4	0.6
Champagne Tower	24	27.8	28.2	2.1	2.3
	36	27.2	27.6	1.3	1.4
Pantomime	24	35.3	36.8	0.1	0.1
	36	34.5	36.0	0.1	0.1
Average	24	32.3	33.7	0.8	1.1
	36	31.0	32.2	0.5	0.6

N-PSNR are generated by comparing the luminance (Y) component of the ground truth and the rendered view, where both video and depth map are compressed using H.264/AVC with QP set to 24 and 36, and DTD is applied during the view synthesis procedure. Difference is generated compared to PSNR and N-PSNR without DTD, which is listed in Table 2.1. From this table it can be noticed that DTD can improve the objective quality for the area which is more noticeable.

Table 4.4 shows the results of noticeable distortion in terms of s and n-PSNR defined in Section 2.5, where the threshold value of 10 is used, and difference is calculated compared to s and n-PSNR without DTD, which is listed in Table 2.2.

Table 4.4: Results of noticeable distortion in terms of s (%) and n-PSNR (dB) generated by comparing the luminance (Y) component of the ground truth and the rendered view with DTD using various test sequences, where both video and depth map are compressed using H.264/AVC with QP set to 24 and 36. Threshold value of 10 is used for s and n-PSNR. Difference is generated compared to s and n-PSNR generated without DTD.

Sequence	QP	with DTD		Difference	
		s	n-PSNR	s	n-PSNR
Cafe	24	4.0	17.8	-0.5	0.1
	36	4.6	18.1	-0.5	0.2
Mobile	24	1.4	21.3	-0.3	1.6
	36	7.7	23.2	-0.2	0.8
Newspaper	24	13.2	20.9	-0.2	0.2
	36	16.2	21.3	-0.3	0.1
Balloons	24	3.3	21.0	0.0	0.6
	36	4.5	21.3	0.0	0.6
Champagne Tower	24	9.0	17.7	-1.2	1.7
	36	10.1	17.6	-0.8	1.1
Pantomime	24	3.4	22.1	0.0	0.1
	36	4.1	22.1	0.0	0.1
Average	24	5.7	20.1	-0.4	0.7
	36	7.9	20.6	-0.3	0.5

This reveals that different factor, s or n-PSNR, or combination of these causes coding efficiency improvement in different sequences.

4.6 Conclusion

In this chapter, the depth transition data is proposed as a new 3-D video data format to achieve better coding efficiency with improved rendered view quality by exploiting localized and non-linear characteristics of the rendered view distortion. While the conventional DIBR based scheme needs to provide the depth map information to every reference views, the advantage of the proposed depth transition data is that one such data set can be applied to multiple views at an arbitrary position. The experimental results show that the subjective quality can be significantly improved with maximum PSNR gain of 2 dB.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

When depth map is coded, it is more appropriate to consider the effect of coding on the rendered view quality instead of the distortion in compressed depth map itself. Therefore, in Chapter 2, we first analyzed the distortion that occurs in the rendered view.

In Chapter 3, new depth map coding tools were proposed, which take the rendered view quality into account. First, new distortion estimation methods were proposed, which derive the relationship between the depth map error to geometry error, and geometry error to the distortion in the rendered view. It was found that there is linear relationship between depth map error and geometry error, which depends on global camera setting parameters. To estimate the distortion in the rendered view, an approach to use the reference video frame was proposed, which can reflect the local video characteristics. We also proposed a simpler estimation method using autoregressive model. These estimation methods were applied to the RD optimized mode selection scheme along with the new Lagrange multiplier derived using the autoregressive model. The experimental results show that the proposed methods can provide accurate estimation of the rendered view distortion due to depth map error, so that coding efficiency can be achieved when it is applied to the depth map coding mode decision scheme. In addition, the new skip mode selection scheme improves the subjective quality by suppressing the flickering artifact, due to temporal inconsistencies in depth map. Also the graph based transform (GBT) was applied to depth map coding, which not only reduces bitrate but also improves the rendered view quality by preserving edges from transform coding.

In Chapter 4, we have developed the new 3-D video format by providing the depth transition data in addition to the existing video plus depth data. The depth transition data indicates where the background/foreground transition happens, therefore is applicable to correct the depth map distortion. While the conventional DIBR based scheme needs to provide the depth map information to every reference views, the advantage of the proposed depth transition data is that one such data set can be applied to multiple views at an arbitrary position. By using the proposed depth transition data, the subjective quality of the rendered view is improved significantly with maximum PSNR gain of 2 dB.

5.2 Future Work

As a future work, it can be investigated how to combine the proposed methods together to achieve optimal performance in terms of the subjective quality of the rendered view and bitrate saving. Then, it can be further investigated how to efficiently represent and encode the depth transition data, and how to apply it to improve the rendered view quality.

The following recommendations are suggested for future work:

- *Efficient representation of depth information.*

A depth map is used to synthesize an intermediate view, and we have shown that the depth transition data can help to improve the rendered view quality. Since they both serve the rendering process, it will be possible to consider one unified data format for such purpose. We will investigate how to combine these two types of data together, and how to efficiently compress it. One example can be performing prediction of DTD using the decoded depth maps.

- *Improving rendered view quality using depth transition data.*

  It is important to provide precise DTD to achieve improvement in rendered view quality. If DTD is generated from depth maps of neighboring views, it may not be possible to provide exact transition position due to occlusion artifact. Also when depth map is noisy, the errors in depth map would lead to errors in DTD. In this case better quality of DTD can be achieved if video information is utilized. Also when there are more than one transition between two views, indicating multiple transition position can help improve rendered view quality. Providing multiple levels of depth change rather than binary level (e.g., foreground and background) can also be considered for future study.

- *Optimization considering combination of the proposed methods.*

  According to our analysis of rendered view distortion, there are various factors which affect the rendered view quality. We have developed methods that can serve different aspect of the problem. Therefore, it is important to find a way to arrange these methods together to achieve optimal performance. In the future it can be investigated how to optimize these for improved subjective quality of the rendered view under bitrate constraints.

References

[1] J. Berent and P. L. Dragotti. Plenoptic manifolds. *IEEE Signal Proc. Magazine*, 24(6):34–44, Nov. 2007.

[2] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proc. of 27th Conf. Computer Graphics and Interactive Techniques, SIGGRAPH 2000*, New Orleans, LA, USA, Jul. 2000.

[3] G. Bjøntegaard. Calculation of average PSNR differences between RD-curves. Document VCEG-M33, ITU-T SG16 Q.6, Apr. 2001.

[4] F. Blais. Review of 20 years of range sensor development. *Journal of Electronic Imaging*, 13(1):231–240, Jan. 2004.

[5] E. Candès and D. Donoho. *Curvelets - a surprisingly effective nonadaptive representation for objects with edges.* Vanderbilt University Press, 1999.

[6] S. C. Chan, H.-Y. Shum, and K.-T. Ng. Image-based rendering and synthesis. *IEEE Signal Proc. Magazine*, 24(6):22–33, Nov. 2007.

[7] C.-L. Chang, M. Makar, S. S. Tsai, and B. Girod. Direction-adaptive partitioned block transform for color image coding. *IEEE Trans. Image Proc.*, 19(7):1740–1755, Jul. 2010.

[8] G. Cheung, J. Ishida, A. Kubota, and A. Ortega. Transform domain sparcification of depth maps using iterative quadratic programming. In *Proc. of IEEE Int. Conf. Image Proc., ICIP 2011*, Brussels, Belgium, Sep. 2011.

[9] G. Cheung, A. Kubota, and A. Ortega. Sparse representation of depth maps for efficient transform coding. In *IEEE Picture Coding Symposium*, Nagoya, Japan, Dec. 2010.

[10] I. Daribo, C. Tillier, and B. Pesquet-Popescu. Adaptive wavelet coding of the depth map for stereoscopic view synthesis. In *Proc. of 2008 IEEE 10th Workshop on Multimedia Signal Processing*, pages 413–417, Queensland, Australia, Oct. 2008.

[11] M. N. Do and M. Vetterli. The contourlet transform: An efficient directional multiresolution image representation. *IEEE Trans. Image Proc.*, 14(12):2091–2106, Dec. 2005.

[12] E. Ekmekcioglu, M. Mrak, S. Worrall, and A. Kondoz. Utilisation of edge adaptive upsampling in compression of depth map videos for enhanced free-viewpoint rendering. In *Proc. of IEEE Int. Conf. Image Proc., ICIP 2009*, Cairo, Egypt, Nov. 2009.

[13] C. Fehn, R. de la Barré, and S. Pastoor. Interactive 3-DTV – concepts and key technologies. *Proc. IEEE*, 94(3):524–538, Mar. 2006.

[14] J. Fu and B. Zeng. Directional discrete cosine transforms: A theoretical analysis. In *Proc. of IEEE Int. Conf. Acoustics, Speech and Signal Proc., ICASSP 2007*, volume I, pages 1105–1108, Honolulu, HI, USA, Apr. 2007.

[15] L. Goldmann, F. de Simone, and T. Ebrahimi. A comprehensive database and subjective evaluation methodology for quality of experience in stereoscopic video. In *Proc. of IS&T/SPIE Electronic Imaging, 3D Image Processing and Applications*, San Jose, CA, USA, Jan. 2010.

[16] D. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. In *Elsevier: Appplied and Computational Harmonic Analysis*, volume 30, pages 129–150, Apr. 2010.

[17] C. T. E. R. Hewage, S. T. Worall, S. Dogan, and A. M. Kondoz. Prediction of stereoscopic video quality using objective quality models of 2-D video. *Electronics Letters*, 44(16):963–965, Jul. 2008.

[18] W.-S. Kim, A. Ortega, P. Lai, D. Tian, and C. Gomila. Depth map distortion analysis for view rendering and depth coding. In *Proc. of IEEE Int. Conf. Image Proc., ICIP 2009*, Cairo, Egypt, Nov. 2009.

[19] W.-S. Kim, A. Ortega, P. Lai, D. Tian, and C. Gomila. Depth map coding with distortion estimation of rendered view. In *Proc. of IS&T/SPIE Electronic Imaging, VIPC 2010*, San Jose, CA, USA, Jan. 2010.

[20] W.-S. Kim, A. Ortega, J. Lee, and H. Wey. 3-D video coding using depth transition data. In *Proc. of 28th Picture Coding Symposium, PCS 2010*, Nagoya, Japan, Dec. 2010.

[21] W.-S. Kim, A. Ortega, J. Lee, and H. Wey. 3-D video quality improvement using depth transition data. In *Proc. of IEEE Int. Workshop on Hot Topics in 3D, Hot 3D 2011*, Barcelona, Spain, Jul. 2011.

[22] W.-S. Kim, P. Pahalawatta, Z. Li, and A. M. Tourapis. New video quality metrics in the H.264 reference software. Document JVT-AB031, ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, Jul. 2008.

[23] R. Krishnamurthy, B.-B. Chai, H. Tao, and S. Sethuraman. Compression and transmission of depth maps for image-based rendering. In *Proc. of IEEE Int. Conf. Image Proc., ICIP 2001*, Thessaloniki, Greece, Oct. 2001.

[24] P. Lai, A. Ortega, C. Dorea, P. Yin, and C. Gomila. Improving view rendering quality and coding efficiency by suppressing compression artifacts in depth-image coding. In *Proc. of SPIE Visual Communic. and Image Proc., VCIP 2009*, San Jose, CA, USA, Jan. 2009.

[25] G. Leon, H. Kalva, and B. Furht. 3D video quality evaluation with depth quality variations. In *Proc. of 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video, 3DTV-CON 2008*, Istanbul, Turkey, May 2008.

[26] Y. Liu, S. Ma, Q. Huang, D. Zhao, W. Gao, and N. Zhang. Compression-induced rendering distortion analysis for texture/depth rate allocation in 3D video compression. In *Proc. of Data Compression Conf., DCC 2009*, Snowbird, UT, USA, Mar. 2009.

[27] M. Maitre and M. N. Do. Joint encoding of the depth image based representation using shape-adaptive wavelets. In *Proc. of IEEE Int. Conf. Image Proc., ICIP 2008*, pages 1768–1771, San Diego, CA, USA, Oct. 2008.

[28] P. Merkle, Y. Morvan, A. Smolic, D. Farin, K. Müller, P. H. N. de With, and T. Wiegand. The effects of multiview depth video compression on multiview rendering. *Singal Proc.: Image Comm.*, 24:73–88, Jan. 2009.

[29] Y. Morvan, D. Farin, and P. H. N. de With. Platelet-based coding of depth maps for the transmission of multiview images. In *Proc. of SPIE: Stereoscopic Displays and Applications (2006)*, volume 6055, San Jose, CA, USA, Jan. 2006.

[30] Y. Morvan, D. Farin, and P. H. N. de With. Multiview depth-image compression using an extended H.264 encoder. *Advanced Concepts for Intelligent Vision Systems*, 4678-2007:675–686, Aug. 2007.

[31] K. Müller, A. Smolic, K. Dix, P. Merkle, and T. Wiegand. Coding and intermediate view synthesis of multiview video plus depth. In *Proc. of IEEE Int. Conf. Image Proc., ICIP 2009*, Cairo, Egypt, Nov. 2009.

[32] S. K. Narang and A. Ortega. Local two-channel critically sampled filter-banks on graphs. In *Proc. of IEEE Int. Conf. Image Proc., ICIP 2010*, Hong Kong, Sep. 2010.

[33] H. T. Nguyen and M. N. Do. Error analysis for image-based rendering with depth information. *IEEE Trans. Image Proc.*, 18(4):703–716, Apr. 2009.

[34] H. Oh and Y.-S. Ho. H.264-based depth map sequence coding using motion information of corresponding texture video. *Advanced Concepts for Intelligent Vision Systems*, 4319-2006:898–907, Dec. 2006.

[35] A. Ortega and K. Ramchandran. Rate-distortion techniques in image and video compression. *IEEE Signal Proc. Magazine*, 15(6):23–50, Nov. 1998.

[36] E. Le Pennec and S. Mallat. Sparse geometric image representations with bandelets. *IEEE Trans. Image Proc.*, 14(4):423–438, Apr. 2005.

[37] W. Philips. Comparison of techniques for intra-frame coding of arbitrarily shaped video object boundary blocks. *IEEE Trans. Circuits Syst. Video Technol.*, 9(7):1009–1012, Oct. 1999.

[38] M. Pinson and S. Wolf. A new standardized method for objectively measuring video quality. *IEEE Trans. Broadcasting*, 50(3):312–322, Sep. 2004.

[39] D. D.-Y. Po and M. N. Do. The contourlet transform: An efficient directional multiresolution image representation. *IEEE Trans. Image Proc.*, 15(6):1610–1620, Jun. 2006.

[40] P. Ramanathan and B. Girod. Rate-distortion analysis for light field coding and streaming. *Singal Proc.: Image Comm.*, 21(6):462–475, Jul. 2006.

[41] G. R. Ramesh and K. Rajgopal. Binary image compression using the radon transform. In *Proc. of XVI Annual Convention and Exhibition of the IEEE in India, ACE 90*, pages 178–182, Bangalore, India, Jan. 1990.

[42] J. Ribas-Corbera and S. Lei. Rate control in DCT video coding for low-delay communications. *IEEE Trans. Circuits Syst. Video Technol.*, 9(1):172–185, Feb. 1999.

[43] H. Rutishauser. The jacobi method for real symmetric matrices. *Numerische Mathematik*, 9(1), Nov. 1966.

[44] A. Sanchez, G. Shen, and A. Ortega. Edge-preserving depth-map coding using tree-based wavelets. In *Prof. of Asilomar Conference on Signals and Systems*, Pacific Grove, CA, USA, Nov. 2009.

[45] M. Sarkis and K. Diepold. Depth map compression via compressed sensing. In *Proc. of IEEE Int. Conf. Image Proc., ICIP 2009*, Cairo, Egypt, Nov. 2009.

[46] D. Scharstein and C. Pal. Learning conditional random fields for stereo. In *Proc. of IEEE Conf. Computer Vision and Pattern Recognition, CVPR 2007*, Minneapolis, MN, USA, Jun. 2007.

[47] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *Proc. of IEEE Conf. Computer Vision and Pattern Recognition, CVPR 2003*, volume 1, pages 195–202, Madison, WI, USA, Jun. 2003.

[48] K. Seshadrinathan, R. Soundararajan, A. C. Bovik, and L. K. Cormack. Study of subjective and objective quality assessment of video. *IEEE Trans. Image Proc.*, 19(6):1427–1441, Jun. 2010.

[49] G. Shen, W.-S. Kim, S. K. Narang, A. Ortega, J. Lee, and H. Wey. Edge-adaptive transforms for efficient depth-map coding. In *Proc. of 28th Picture Coding Symposium, PCS 2010*, Nagoya, Japan, Dec. 2010.

[50] G. Shen, W.-S. Kim, A. Ortega, J. Lee, and H. Wey. Edge-aware intra prediction for depth-map coding. In *Proc. of IEEE Int. Conf. Image Proc., ICIP 2010*, Hong Kong, Sep. 2010.

[51] T. Sikora, S. Bauer, and B. Makai. Efficiency of shape-adaptive 2-D transforms for coding of arbitrarily shaped image segments. *IEEE Trans. Circuits Syst. Video Technol.*, 5(3):254–258, Jun. 1995.

[52] A. Smolic, K. Müller, N. Stefanoski, J. Ostermann, A. Gotchev, G. B. Akar, G. Triantafyllidis, and A. Koz. Coding algorithms for 3DTV–a survey. *IEEE Trans. Circuits Syst. Video Technol.*, 17(11):1606–1621, Nov. 2007.

[53] M. Soumekh. Binary image reconstruction from four projections. In *Proc. of IEEE Int. Conf. Acoustics, Speech and Signal Proc., ICASSP 1988*, pages 1280–1283, New York, NY, USA, Apr. 1988.

[54] G. J. Sullivan, P. N. Topiwala, and A. Luthra. The H.264/AVC advanced video coding standard: overview and introduction to the fidelity range extensions. In *Proc. of SPIE: Applications of Digital Image Processing XXVII*, volume 5558, pages 454–474, Denver, CO, USA, Aug. 2004.

[55] G. J. Sullivan and T. Wiegand. Rate-distortion optimization for video compression. *IEEE Signal Proc. Magazine*, 15(6):74–90, Nov. 1998.

[56] T. K. Tan, G. J. Sullivan, and T. Wedi. Recommended simulation conditions for coding efficiency experiments. Document VCEG-AH10, ITU-T SG16 Q.6, Jan. 2008.

[57] M. Tanimoto, T. Fujii, and K. Suzuki. Experiment of view synthesis using multi-view depth. Document M14889, ISO/IEC JTC1/SC29/WG11, Oct. 2007.

[58] M. Tanimoto, T. Fujii, and K. Suzuki. View synthesis algorithm in view synthesis reference software 2.0 (VSRS2.0). Document M16090, ISO/IEC JTC1/SC29/WG11, Feb. 2009.

[59] M. Tanimoto, T. Fujii, M. P. Tehrani, K. Suzuki, and M. Wildeboer. Depth estimation reference software (DERS) 3.0. Document M16390, ISO/IEC JTC1/SC29/WG11, Apr. 2009.

[60] A. Tikanmaki, A. Gotchev, A. Smolic, and K. Muller. Quality assessment of 3D video in rate allocation experiments. In *Proc. of IEEE Int. Symposium on Cosumer Electronics, ISCE 2008*, Algarve, Portugal, Apr. 2008.

[61] E. Trucco and A. Verri. *Introductory techniques for 3-D computer vision.* Prentice Hall, 1998.

[62] D. Tzovaras, N. Grammalidis, and M.G. Strintzis. Disparity field and depth map coding for multiview image sequence compression. In *Proc. of IEEE Int. Conf. Image Proc., ICIP 1996*, Lausanne, Switzerland, Oct. 1996.

[63] V. Velisavljevic, B. Beferull-Lozano, M. Vetterli, and P.L. Dragotti. Directionlets: Anisotropic multidirectional representation with separable filtering. *IEEE Trans. Image Proc.*, 15(7):1916–1933, Jul. 2006.

[64] Video. Description of exploration experiment in 3D video. Document N9596, ISO/IEC JTC1/SC29/WG11, Jan. 2008.

[65] Video. Description of exploration experiments in 3D video coding. Document N11274, ISO/IEC JTC1/SC29/WG11, Apr. 2010.

[66] Video. Report on experimental framework for 3D video coding. Document N11631, ISO/IEC JTC1/SC29/WG11, Oct. 2010.

[67] Video and Requirements. Applications and requirements on 3D video coding. Document N11829, ISO/IEC JTC1/SC29/WG11, Jan. 2011.

[68] Z. Wang and A. C. Bovik. *Modern image quality assessment.* Morgan & Claypool Publishers, 2006.

[69] Z. Wang and A. C. Bovik. Mean squared error: Love it or leave it? - a new look at signal fidelity measures. *IEEE Signal Proc. Magazine*, 26(1):98–117, Jan. 2009.

[70] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Proc.*, 13(4):600–612, Apr. 2004.

[71] Z. Wang, E.P. Simoncelli, and A.C. Bovik. Multi-scale structural similarity for image quality assessment. In *Proc. of Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, USA, Nov. 2003.

[72] T. Wiegand and B. Girod. Lagrange multiplier selection in hybrid video coder control. In *Proc. of IEEE Int. Conf. Image Proc., ICIP 2009*, pages 542–545, Cairo, Egypt, Oct. 2001.

[73] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan. Rate-constrained coder control and comparison of video coding standards. *IEEE Trans. Circuits Syst. Video Technol.*, 13(7):688–703, Jul. 2003.

[74] M. Wien. Variable block-size transforms for H.264/AVC. *IEEE Trans. Circuits Syst. Video Technol.*, 13(7):604–613, Jul. 2003.

[75] R. Willett and R. Nowak. Platelets: A multiscale approach for recovering edges and surfaces in photon-limited medical imaging. *IEEE Trans. Medical Imaging*, 22(3):332–350, Mar. 2003.

[76] K. Yamamoto, M. Kitahara, H. Kimata, T. Yendo, T. Fujii, M. Tanimoto, S. Shimizu, K. Kamikura, and Y. Yashima. Multiview video coding using view interpolation and color correction. *IEEE Trans. Circuits Syst. Video Technol.*, 17(11):1436–1449, Nov. 2007.

[77] Y. Ye and M. Karczewicz. Improved H.264 intra coding based on bi-directional intra prediction, directional transform, and adaptive coefficient scanning. In *Proc. of IEEE Int. Conf. Image Proc., ICIP 2008*, pages 2116–2119, San Diego, CA, USA, Oct. 2008.

[78] B. Zeng and J. Fu. Directional discrete cosine transforms for image coding. In *Proc. of IEEE Int. Conf. Multimedia and Expo, ICME 2006*, pages 721–724, Toronto, Canada, Jul. 2006.

[79] B. Zeng and J. Fu. Directional discrete cosine transforms - a new framework for image coding. *IEEE Trans. Circuits Syst. Video Technol.*, 18(3):305–313, Mar. 2008.

[80] C. Zhang and T. Chen. Light field sampling. *Synthesis lectures on image, video, and multimedia processing*, 6, 2006.

[81] C. Zhang, K. Ugur, J. Lainema, and M. Gabbouj. Video coding using spatially varying transform. In *Proc. of 3rd Pacific Rim Symposium on Advances in Image and Video Technology, PSIVT 2007*, pages 796–806, Tokyo, Japan, Jan. 2009.

[82] C. Zhang, K. Ugur, J. Lainema, and M. Gabbouj. Video coding using variable block-size spatially varying transforms. In *Proc. of IEEE Int. Conf. Acoustics, Speech and Signal Proc., ICASSP 2009*, pages 905–908, Taipei, Taiwan, Apr. 2009.

[83] L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2004)*, 23(3):660–608, Aug. 2004.