

# BIT ALLOCATION FOR MULTIVIEW IMAGE COMPRESSION USING CUBIC SYNTHESIZED VIEW DISTORTION MODEL

Vladan Velisavljević<sup>1</sup>, Gene Cheung<sup>2</sup>, Jacob Chakareski<sup>3</sup>

<sup>1</sup>Deutsche Telekom Laboratories, Berlin, Germany,

<sup>2</sup>National Institute of Informatics, Tokyo, Japan,

<sup>3</sup>Ecole Polytechnique Fédérale de Lausanne, Switzerland

## ABSTRACT

“Texture-plus-depth” has become a popular coding format for multiview image compression, where a decoder can synthesize images at intermediate viewpoints using encoded texture and depth maps of closest captured view locations via depth-image-based rendering (DIBR). As in other resource-constrained scenarios, limited available bits must be optimally distributed among captured texture and depth maps to minimize the expected signal distortion at the decoder. A specific challenge of multiview image compression for DIBR is that the encoder must allocate bits without the knowledge of how many and which specific virtual views will be synthesized at the decoder for viewing. In this paper, we derive a cubic synthesized view distortion model to describe the visual quality of an interpolated view as a function of the view’s location. Given the model, one can easily find the virtual view location between two coded views where the maximum synthesized distortion occurs. Using a multiview image codec based on shape-adaptive wavelet transform, we show how optimal bit allocation can be performed to minimize the maximum view synthesis distortion at any intermediate viewpoint. Our experimental results show that the optimal bit allocation can outperform a common uniform bit allocation scheme by up to 1.0dB in coding efficiency performance, while simultaneously being competitive to a state-of-the-art H.264 codec.

**Index Terms**— multiview imaging, depth-image-based rendering, bit allocation, distortion modeling

## 1. INTRODUCTION

With the advent of sophisticated camera systems and the growing consumer demand for a more immersive visual experience, multiview imaging has become a popular paradigm, where a scene of interest is captured simultaneously by multiple cameras physically located in proximity. If the closely spaced cameras are capable of capturing both texture (typically RGB components) and depth images (per-pixel distance between physical objects being captured and capturing camera)<sup>1</sup>, then texture and depth maps can be compressed together into one format, commonly called *texture-plus-depth* representation [1]. This enables a decoder to interpolate an image at an intermediate viewpoint using texture and depth images of the two closest captured view locations via depth-image-based rendering (DIBR) [2]. The ability to generate images at any desired view-

point in a defined continuum, called *free viewpoint* [3], offers the user a new level of visual immersion compared to traditional single-camera / single-view imaging systems.

As in other resource-constrained scenarios, available coding bits out of a bit budget must be optimally distributed among captured texture and depth images in order to minimize the expected signal distortion of one or more constructed view(s) at the decoder. What is particularly challenging for multiview image compression with DIBR, however, is that the encoder must allocate bits in the absence of the knowledge of how many and which specific virtual views will be synthesized at the decoder for viewing. While [4] showed experimentally that the synthesized view distortion as a function of the view location obeys a convex shape, no bit allocation strategy has been formally devised for texture-plus-depth representation in a rigorous way.

In this paper, we derive a cubic synthesized view distortion model that mathematically describes the visual quality of an interpolated view as a function of the view’s location. Given the model, one can easily find the view location between two captured coded views where the *maximum* synthesized distortion occurs. Using a state-of-the-art multiview image codec based on the shape-adaptive wavelet transform (SA-WT) [5], we show how optimal bit allocation can be performed to minimize the maximum synthesized distortion at any intermediate viewpoint in a computationally efficient manner. For the case when there are only two captured viewpoints, we show experimentally that the optimal bit allocation can outperform a commonly deployed uniform bit allocation scheme by up to a 1.0dB in visual quality measured in Peak Signal-to-Noise Ratio (PSNR). At the same time, our compression framework demonstrates coding performance competitive to that of a conventional H.264 coder.

The outline of the rest of the paper is as follows. First, we provide a system level description of our coding framework in Section 2. Then, we present the proposed distortion model and the related problem formulation in Sections 3 and 4, respectively. Next, we examine the performance of our framework via simulation experiments in Section 5. Finally, concluding remarks are provided in Section 6.

## 2. SYSTEM DESCRIPTION

Here, we first review the design of the shape-adaptive wavelet transform, as adopted from [5], and motivate its application to multiview image coding. Then, we describe the virtual view rendering process and analyze its intrinsic properties. Finally, we show how by using these properties we can model the distortion of synthesized images at arbitrary view locations.

The work of J. Chakareski has been supported by the Swiss National Foundation under Ambizione grant PZ000P2-126416.

<sup>1</sup>Depth values can be either captured directly using time-of-flight cameras or estimated using depth estimation algorithms.

## 2.1. Shape-Adaptive Wavelet Transform

The SA-WT was originally proposed in [6] to efficiently deal with irregular shapes of objects in images. Wavelet filtering is adapted to object boundaries so that the pixels convolved with the wavelet filters and located in the outer region (out of the object boundaries) are filled with values related to the pixels from the inner region, so that a *symmetric boundary extension* is applied [6]. That is, the filled pixel values are symmetric with respect to the object boundary. As a result, the high-pass wavelet coefficients produced by filtering across the discontinuities along these boundaries have reduced magnitudes, which, in turn, leads to a lower distortion of the decoded image, for the same bit rate. However, this improvement of the rate-distortion (RD) performance is counter-acted by an additional bit rate spent for the edge map coding needed for the decoder to apply the identical inverse SA-WT. Thus, bits must be allocated between edge maps and wavelet coefficients, so that the RD performance of the SA-WT image coding is optimized.

To apply previous SA-WT concepts designed for single-view images to multiview image coding, we use a modified version of the SA-WT proposed in [5], where the shape boundaries used for the adaptation of the wavelet filtering do not necessarily have to be closed contours. In addition, we exploit the correlation between edge locations in texture and depth images, for the same viewpoint, in order to efficiently encode them, as explained next.

## 2.2. Multiview Image Coder

The input to our multiview image coder is a set of  $N$  texture and depth images,  $\{t_0, t_1, \dots, t_{N-1}\}$  and  $\{d_0, d_1, \dots, d_{N-1}\}$  (hence  $2N$  images in total), captured at known camera viewpoints denoted as  $\mathcal{V} = \{0, 1, \dots, N-1\}$ . The camera viewpoints can in general be located anywhere in the 3D coordinates, but, for simplicity we only consider the case of a baseline equidistant camera setup.

For coding, either the left-most or the right-most camera viewpoint is chosen as the first encoded viewpoint, denoted as 0. In particular, given allocated target bit rates  $R_{t_0}$  and  $R_{d_0}$ , the corresponding texture and depth images,  $t_0$  and  $d_0$ , are encoded using the SA-WT followed by an application of the Set Partitioning in Hierarchical Trees (SPIHT) coder [7] to the resulting wavelet coefficients. The associated edge maps for the SA-WT are losslessly encoded using the traditional Freeman method [8] at bit rates  $R_{c,t_0}$  and  $R_{c,d_0}$ , as shown in Fig. 1.

In order to enable scaling of the influence of the edge maps on the RD performance, we introduce parameters  $0 \leq C_{t_0}, C_{d_0} < 1$  that determine the portion of pixel coordinates in  $t_0$  and  $d_0$  selected as edge locations, respectively. In particular, the edge map associated with the depth image  $d_0$  is generated such that only the  $C_{d_0}$  portion of pixel coordinates comprising the largest magnitudes of the first-order difference of  $d_0$  computed across the horizontal and vertical directions is retained, as the most prominent edge locations. Then, recalling the fact that edges in the depth image commonly coincide with the edges in the texture image captured at the same viewpoint [5],<sup>2</sup> the edge map for  $t_0$  comprises the edge map generated for  $d_0$  and the additional  $C_{t_0}$  portion of the remaining pixel coordinates featuring the largest magnitudes of the first-order difference in  $t_0$ . An example of selected edge maps for  $t_0$  and  $d_0$  is shown in the top two images in Fig. 2.

<sup>2</sup>Notice that this is not a bidirectional relation in general because texture images are usually more complex than depth images and have a richer edge structure.

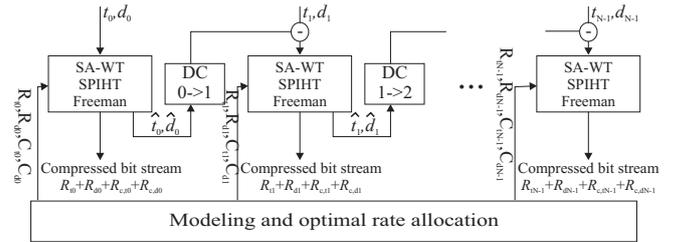


Fig. 1. Schematic illustration of the coder.

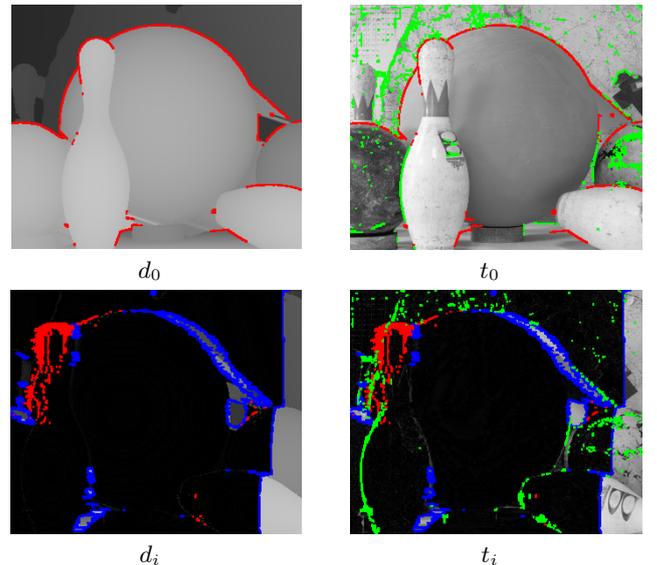


Fig. 2. Edge maps for depth and texture images at viewpoints 0 (top) and  $i > 0$  (bottom) for the Middlebury data set Bowling2. The blue color shows the edge locations associated with disparity compensation discontinuities. The red color shows the most prominent edge locations in depth images used also as edge locations in texture images. The green color shows the edge locations selected as most prominent in texture images. This example is obtained for  $C_{d_0} = 0.26\%$ ,  $C_{t_0} = 0.30\%$ ,  $C_{d_i} = 0.52\%$  and  $C_{t_i} = 0.30\%$ , whereas the resulting bit rates are  $R_{c,d_0} = 0.0093$ ,  $R_{c,t_0} = 0.0036$ ,  $R_{c,d_i} = 0.0021$  and  $R_{c,t_i} = 0.0026$  bpp.

The parameters  $R_{t_0}$ ,  $R_{d_0}$ ,  $C_{t_0}$ , and  $C_{d_0}$  are used as an input to the coder and they are optimized for the best RD coding performance (to be explained in Section 4). The bit streams of lengths  $R_{t_0}$ ,  $R_{d_0}$ ,  $R_{c,t_0}$ , and  $R_{c,d_0}$  are generated by the SPIHT and Freeman coders as the output. Note that only the additional  $C_{t_0}$  edge locations have to be encoded in the case of the edge map associated with  $t_0$ . That is because the edge locations in  $t_0$  that coincide with the edge map associated with  $d_0$  have been already encoded and, therefore, will be available at the decoder for reconstructing the edge map associated with  $t_0$ .

The texture and depth images at the other camera viewpoints,  $i = 1, 2, \dots, N-1$ , are differentially encoded such that the SA-WT and the SPIHT coders are applied to the respective differences between the captured images and their predictions obtained by disparity compensation (denoted as DC blocks in Fig. 1) using the quantized images at the previous viewpoint  $i-1$ . Similar to the case of view 0, the allocated target bit rates  $R_{t_i}$  and  $R_{d_i}$  are taken as input

parameters to the coding procedure. In addition, the corresponding edge maps for each  $t_i$  and  $d_i$  are computed similarly to those for  $t_0$  and  $d_0$  using the parameters  $C_{t_i}$  and  $C_{d_i}$ .

However, unlike the intra-coded  $t_0$  and  $d_0$ , the images obtained as differences between the actual viewpoint  $i$  images and their disparity compensated versions often have a large-step discontinuity between the compensated and non-compensated regions (which we call here *disparity compensation discontinuities*), either around occluded areas or along the image borders. The disparity compensation discontinuities coincide in the texture and depth images at the same viewpoint and their locations are known to the decoder from the geometry of the camera locations and the previously encoded views, without additional encoding. Thus, the edge map for  $d_i$  consists of the disparity compensation discontinuities and the  $C_{d_i}$  portion of the remaining most prominent edge locations in  $d_i$ . Similarly, the edge map for  $t_i$  comprises the entire edge map for  $d_i$  and the additional  $C_{t_i}$  portion of the remaining most prominent edge locations in  $t_i$ . Note that only the added edge locations corresponding to  $C_{d_i}$  pixels from  $d_i$  and to  $C_{t_i}$  pixels from  $t_i$  have to be encoded by the Freeman coder, while the other locations are already known to the decoder, as explained above. An example of the edge map for  $t_i$  and  $d_i$  is illustrated in the bottom two images in Fig. 2.

### 2.3. Virtual View Rendering

In a common virtual view rendering setup, by knowing the extrinsic camera parameters and the captured view locations, a virtual intermediate view can be synthesized by a back projection of the two nearest captured reference views to the 3D scene coordinates, followed by a projection to the virtual view camera location [4]. Additionally, in DIBR, each captured texture image is associated with a per-pixel depth image, which is used to determine the distance (depth) between the camera and the scene's 3D surface. To prevent overwriting the foreground with background information in the projection, a depth buffer is maintained when reconstructing a virtual view. Then, given the two pixel projections from the two respective reference views, we select for the corresponding pixel reference in the virtual view the projection that is closer to the virtual camera location, as measured from the 3D scene coordinates. In other words, this is the projection that exhibits a smaller depth value for its associated pixel location in the respective reference view.

We denote the two reference viewpoint locations as  $l$  and  $r$ , referring to *left* and *right* references, and the virtual viewpoint location as  $x$ . Due to a typically high correlation between captured neighboring views when the camera spacings are small, the projections  $t_{x,P_l}$  and  $t_{x,P_r}$  at the same virtual view location  $x$  obtained from the two reference texture images  $t_l$  and  $t_r$  overlap at most of the pixels, i.e.,

$$t_{x,P_l}(i) = t_{x,P_r}(i) \quad (1)$$

at such pixel coordinates  $i$ . For overlaying these two projections into one, a view-dependent texture blending technique is used, so that each pixel at coordinate  $i$  that is available from both projections is computed as a weighted linear sum  $t_x(i) = (1-x)t_{x,P_l}(i) + xt_{x,P_r}(i)$ . Here,  $0 \leq x \leq 1$  is proportional to the distance between the virtual view location  $x$  and the left reference view location  $l$ .

For the pixels that do not have available both projection versions due to occlusion or because their coordinates lie out of the reference image borders, the resulting pixel value is either equal to the single available projection or to zero, in the case where no projection is available. The zeros in the virtual view are filled-in during a post-processing step of in-painting or interpolation.

Note that other more complex blending techniques are also possible (e.g. [9] or [4] with coordinate rotation). However, we use the described technique because of simplicity and smoothness of the associated virtual view distortion with respect to  $x$ .

For simplicity and without loss of generality, we assume that both captured and virtual camera viewpoints are located on a baseline, reducing the camera coordinates to a 1D index. Then, the virtual viewpoint location  $x$  is constrained to lie on a line between the left and right reference views. We assume the notation where  $x = 0$  coincides with the left reference and  $x = 1$  with the right reference view. Note that the assumed dimensionality reduction also leads to a simpler view projection using only a 1D disparity compensation warping operator instead of a complex 3D coordinate projection.

Let  $i$  denote a pixel coordinate at the virtual viewpoint location  $x$ . Furthermore, denote as  $j_l(x, i)$  the pixel coordinate in the reference  $t_l$  that is warped to the coordinate  $i$  in the projected image  $t_{x,P_l}$ , i.e., if there is no occlusion,  $t_{x,P_l}(i) = t_l(j_l(x, i))$ . Similarly, denote as  $j_r(x, i)$  the pixel coordinate in the reference  $t_r$  that is warped to the same coordinate  $i$  in the projection  $t_{x,P_r}$ , or, in mathematical terms,  $t_{x,P_r}(i) = t_r(j_r(x, i))$ . Notice that if the two reference texture images are perfectly matched so that the pixel intensities at the same warped coordinate  $i$  are equal and if those pixels are not occluded, then (1) holds and, thus,

$$t_l(j_l(x, i)) = t_r(j_r(x, i)). \quad (2)$$

In the case when DIBR is used in disparity compensation, the warped coordinate  $i$  along the camera baseline is obtained by a linear shift of the original coordinate  $j_l(x, i)$  (or  $j_r(x, i)$ ) scaled by the distance  $x$  (or  $1-x$ ) between the virtual and reference view locations and by the depth value at the original coordinate, i.e.,

$$i = j_l(x, i) - x \cdot d_l(j_l(x, i)) \quad (3)$$

$$i = j_r(x, i) + (1-x) \cdot d_r(j_r(x, i)), \quad (4)$$

where  $d_l$  and  $d_r$  are the depth images captured at the left and right reference views.

Using the proposed camera location constraint and the assumed notation, we can rewrite the blending operator for rendering the texture images associated with virtual views as

$$t_x(i) = \begin{cases} (1-x)t_l(j_l) + xt_r(j_r), & t_l(j_l), t_r(j_r) \text{ avlb.} \\ t_l(j_l), & t_r(j_r) \text{ unavlb.} \\ t_r(j_r), & t_l(j_l) \text{ unavlb.} \\ 0, & t_l(j_l), t_r(j_r) \text{ unavlb.} \end{cases}, \quad (5)$$

where the arguments  $(x, i)$  in  $j_l(x, i)$  and  $j_r(x, i)$  are dropped for simplicity.

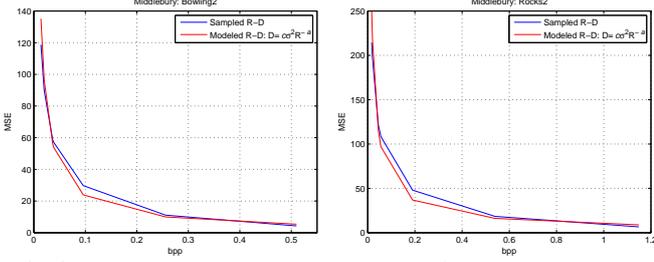
## 3. MODELING

### 3.1. Rate-distortion model for $\alpha$ -Lipschitz smooth images

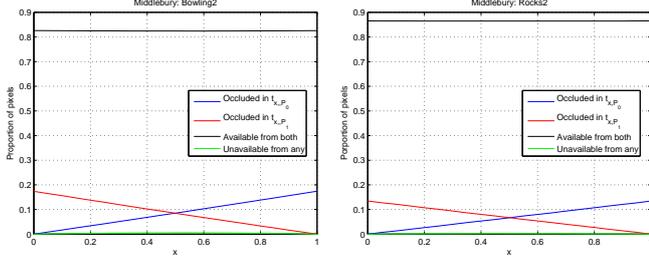
Following the analysis of Mallat [10], the RD relation for a general  $\alpha$ -Lipschitz smooth image compressed by an adaptive wavelet transform can be represented as

$$D(R) = c(\sigma^2)R^{-\alpha}. \quad (6)$$

Furthermore, we assume that the scaling  $c(\sigma^2)$  is linearly dependent on the variance  $\sigma^2$  of the high-pass wavelet coefficients, so that the relation in (6) is simplified to  $D = c\sigma^2 R^{-\alpha}$ . Fig. 3 shows a comparison of modeled and measured RD behavior for two data sets, where the parameters  $\alpha$  and  $c$  are estimated using a linear least-square estimator.



**Fig. 3.** Modeled and measured RD behavior for the Middlebury data sets *Bowling2* (left) and *Rocks2* (right). The estimated parameters are  $a = 0.89$ ,  $c = 3.18 \cdot 10^{-2}$  and  $\sigma^2 = 93.29$  for *Bowling2* and  $a = 0.79$ ,  $c = 5.67 \cdot 10^{-2}$  and  $\sigma^2 = 175.4$  for *Rocks2*.



**Fig. 4.** Proportion of occluded pixels for virtual viewpoint location in between two reference views for *Bowling2* (left) and *Rocks2* (right). The estimated parameter  $\alpha = 0.1739$  for *Bowling2* and  $\alpha = 0.1345$  for *Rocks2*.

In addition, we assume that the SA-WT affects only the variance  $\sigma^2$ , but not the smoothness degree  $a$  and the scaling constant  $c$ . Although this assumption is only an approximation, it matches well the RD behaviors observed in practice, as verified in our experiments.

### 3.2. Cubic model for virtual view distortion

To estimate the virtual view distortion in the case when the reference texture and depth images are quantized, we first analyze the portion of the pixels generated in each of the four cases of the blending operator in (5), at virtual view location  $x$ . Supported by experiments, we assume that: (a) each portion of occluded pixels in the projections  $t_{x, P_l}$  and  $t_{x, P_r}$  grows linearly with respect to the distance between the reference and virtual view; and (b) no pixel for  $x$  in between the left and right references is occluded in both projections  $t_{x, P_l}$  and  $t_{x, P_r}$ . Note that the actual pixel portions depend on the complexity of the 3D scene. Nonetheless, the above assumptions are realistic in practical implementations that feature small camera spacings along the baseline, as supported by the example shown in Fig. 4.

Denote by  $\alpha$  the growth rate of the portion of occluded pixels in the projection with respect to the distance between the reference and virtual view. Then, the portion of pixels that are occluded in  $t_{x, P_l}$  (that is, the pixels unavailable in the projection from the left reference, i.e., the third line in (5)) is equal to  $\alpha \cdot x$ . Similarly, the portion of pixels occluded in  $t_{x, P_r}$  (the second line in (5)) is given by  $\alpha \cdot (1 - x)$ . From the assumption (b) above, the portion of pixels that correspond to the fourth line of (5) is equal to zero and, thus, the portion of pixels available from both projections  $t_{x, P_l}$  and  $t_{x, P_r}$  remains constant across  $x$  and is equal to  $(1 - \alpha)$ . Fig. 4 illustrates these portion values for two data sets *Bowling2* (left) and *Rocks2* (right) from the Middlebury database [11].

Let us further assume that the covariance of the texture image pixels can be modeled as  $E\{t(i) \cdot t(i+k)\} = s_t^2 \cdot \rho^{|k|}$  [12, 13], where  $s_t^2 = E\{t(i)^2\}$  and  $0 \leq \rho \leq 1$  is the covariance of the texture image pixels for a unit shift  $k = 1$ . In addition, since  $\rho$  is very close

to 1, the covariance can be efficiently linearized by the expression  $s_t^2(1 - (1 - \rho) \cdot |k|)$ .

We denote the quantized versions of the reference texture and depth images as  $\hat{t}_l$ ,  $\hat{t}_r$ ,  $\hat{d}_l$ , and  $\hat{d}_r$ , respectively, with the associated distortions  $D_{t_l} = E\{(t_l - \hat{t}_l)^2\}$  and  $D_{t_r}$ ,  $D_{d_l}$ , and  $D_{d_r}$  defined similarly. Using (5) and the estimated portions of pixels, the view  $t_x$  rendered at viewpoint  $x$ , using the quantized reference images as an anchor, contains the following pixels

$$\hat{t}_x(i) = \begin{cases} (1-x)\hat{t}_l(\hat{j}_l) + x\hat{t}_r(\hat{j}_r), & \text{at } (1-\alpha) \# \text{ of pxls} \\ \hat{t}_l(\hat{j}_l), & \text{at } \alpha(1-x) \# \text{ of pxls} \\ \hat{t}_r(\hat{j}_r), & \text{at } \alpha x \# \text{ of pxls} \end{cases}, \quad (7)$$

where similarly to (5), the arguments  $(x, i)$  in the quantized indices  $\hat{j}_l(x, i)$  and  $\hat{j}_r(x, i)$  are dropped. Thus, the distortion of the rendered view is expressed as

$$D_s(x) = E\{(t_x(i) - \hat{t}_x(i))^2\} = (\alpha(1-x) + (1-\alpha)(1-x)^2)E_l + (\alpha x + (1-\alpha)x^2)E_r + 2(1-\alpha)x(1-x)E_{lr}, \quad (8)$$

where

$$E_l = E\{(t_l(j_l) - \hat{t}_l(\hat{j}_l))^2\}, \quad (9)$$

$$E_r = E\{(t_r(j_r) - \hat{t}_r(\hat{j}_r))^2\}, \quad (10)$$

$$E_{lr} = E\{(t_l(j_l) - \hat{t}_l(\hat{j}_l)) \cdot (t_r(j_r) - \hat{t}_r(\hat{j}_r))\}. \quad (11)$$

To compute (9), denote the quantization error of  $t_l$  as  $\epsilon_{t_l}(i) = t_l(i) - \hat{t}_l(i)$ , where  $\epsilon_{t_l}(i)$  is zero-mean, with variance  $E\{\epsilon_{t_l}^2\} = D_{t_l}$ , and independent of the quantized values in  $\hat{t}_l$ . Thus,

$$E_l = E\left\{(t_l(j_l) - t_l(\hat{j}_l) + \epsilon_{t_l}(\hat{j}_l))^2\right\}.$$

Approximating  $E\{t_l(j_l)^2\} = E\{t_l(\hat{j}_l)^2\} = s_t^2$  and  $E\{\epsilon_{t_l}(\hat{j}_l) \cdot [t_l(j_l) - t_l(\hat{j}_l)]\} = 0$ , we obtain that  $E_l = D_{t_l} + 2s_t^2(1-\rho)E\{|j_l - \hat{j}_l|\}$ . Since from (3) it follows that  $|j_l - \hat{j}_l| = x|d_l - \hat{d}_l|$ , we can write  $E_l = D_{t_l} + 2s_t^2(1-\rho)x E\{|d_l - \hat{d}_l|\}$ . Furthermore, since from Jensen's inequality it holds  $E\{|d_l - \hat{d}_l|\} \leq D_{d_l}^{1/2}$ , we can approximate  $E_l \approx D_{t_l} + 2s_t^2(1-\rho)x D_{d_l}^{1/2}$ . Similarly, for (10), we have  $E_r \approx D_{t_r} + 2s_t^2(1-\rho)(1-x) D_{d_r}^{1/2}$ .

For (11), by assuming independent zero-mean quantization errors  $\epsilon_{t_l}(i)$  and  $\epsilon_{t_r}(i)$ , we obtain  $E_{lr} = E\{(t_l(j_l) - t_l(\hat{j}_l)) \cdot (t_r(j_r) - t_r(\hat{j}_r))\}$ . Recalling from (2) that  $t_l(j_l) = t_r(j_r)$ , the four terms in  $E_{lr}$  are obtained as follows:

$$E\{t_l(j_l)t_r(j_r)\} = s_t^2,$$

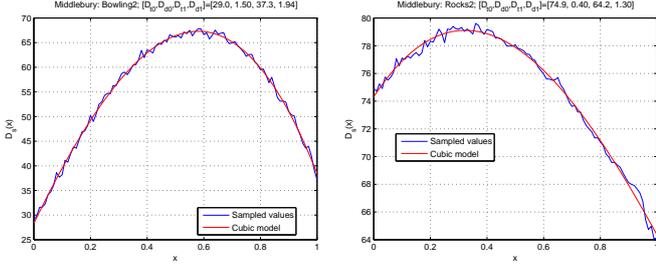
$$E\{t_l(j_l)t_r(\hat{j}_r)\} \approx s_t^2 \left(1 - (1-\rho)(1-x)D_{d_r}^{1/2}\right),$$

$$E\{t_l(\hat{j}_l)t_r(j_r)\} \approx s_t^2 \left(1 - (1-\rho)x D_{d_l}^{1/2}\right),$$

$$E\{t_l(\hat{j}_l)t_r(\hat{j}_r)\} \approx s_t^2 \left(1 - (1-\rho)(x D_{d_l}^{1/2} + (1-x) D_{d_r}^{1/2})\right).$$

In sum, these terms are canceled out and hence  $E_{lr} \approx 0$ .

Finally, by substituting  $E_l$ ,  $E_r$  and  $E_{lr}$  in (8),  $D_s(x)$  can be approximated as a cubic polynomial with respect to  $x$ . Fig. 5 shows two examples of a sampled virtual view distortion and the corresponding cubic model obtained for quantized data sets *Bowling2* (left) and *Rocks2* (right). We can observe a close match between the model and the actual sampled values.



**Fig. 5.** Two examples of sampled virtual view distortion and estimated cubic model using a linear least-square estimator for Bowling2 (left) and Rocks2 (right).

## 4. PROBLEM FORMULATION

### 4.1. Model Parameter Sampling

The model parameters required to optimize RD performance are classified in 3 groups: image coding RD modeling, edge map influence, and cubic modeling for virtual view distortion.

1. *RD model for images.* The two parameters  $c$  and  $a$  from (6) are estimated using a linear least-square estimator applied to each texture and depth image, separately. Following the statistical analysis recommendation from [14] that the number of samples should be at least a multiple of the number of parameters, we sample the RD values at 4 operating points, such that the range of sampled distortion values lies in the interval  $[\sigma^2/4, 1.5\sigma^2]$ . The estimated parameters are denoted as  $c_{t_i}$ ,  $c_{d_i}$ ,  $a_{t_i}$ , and  $a_{d_i}$ , for  $i = 0, \dots, N-1$ .

2. *Edge maps for SA-WT.* The SA-WT is applied to each texture and depth image,  $t_i$  and  $d_i$ , for different values of  $C_{d_i}$  and  $C_{t_i}$  ranging from 0 to 1% of the total number of pixels. For each case, the energy of the high-pass wavelet coefficients is measured and denoted as  $\sigma_{t_i}^2$  and  $\sigma_{d_i}^2$ , respectively. At the same time, the resulting edge maps are losslessly encoded using the Freeman method. In particular, as explained previously, each depth image edge map is differentially constructed with respect to the boundaries between the occluded and disclosed regions. In addition, the texture image edge maps are differentially constructed with respect to their counterparts for the corresponding depth images, as explained in Section 2.2. The obtained edge map bit rates are denoted as  $R_{c,t_i}$  and  $R_{c,d_i}$ . Note that the complexity of the SA-WT and the Freeman method is significantly lower than that of the entire coder and, thus, despite a possibly large number of samples that need to be acquired, the sampling does not exhibit a significant computational cost. Furthermore, when the camera spacings are small, the edge structure in the differentially encoded  $d_i$  and  $t_i$ , for  $i \geq 1$ , is significantly simpler than that in  $d_0$  and  $t_0$ . Therefore, the number of acquired samples for  $i \geq 1$  can be smaller than that for  $i = 0$ , which in turn allows for an additional reduction of the computational cost of sampling.

3. *Cubic model for virtual view distortion.* Since the parameters  $\alpha$  and  $s_t^2$  used in the modeling of the synthesized view distortion can be measured precisely from the original data at very low computational cost, we only estimate the covariance parameter  $\rho$ . Recall from the modeling in Section 3.2 that  $\rho$  influences  $D_s(x)$  only by scaling the depth image distortions  $D_{d_i}$  and  $D_{t_r}$ , but it is invariant to  $D_{t_i}$  and  $D_{t_r}$ . Thus, to estimate  $\rho$ , we use a linear least-square estimator where the synthesis distortion is sampled using the original (uncompressed) texture images (i.e.,  $D_{t_i}, D_{t_r} = 0$ ) and compressed depth images such that  $\sigma_d^2/4 \leq D_{d_i}, D_{d_r} \leq \sigma_d^2/2$ . Following [14] and allowing for a more robust estimation, we compute at least 4 samples of  $D_s$ .

### 4.2. Encoding Rate and Virtual View Distortion

The total encoding bit rate is equal to the sum of all bit rates allocated to the images  $t_i$  and  $d_i$ , for  $i = 0, \dots, N-1$ , and the bit rates spent for the edge map coding. That is,

$$R_{tot}(\mathcal{R}) = \sum_{i=0}^{N-1} [R_{t_i} + R_{d_i} + R_{c,t_i} + R_{c,d_i}], \quad (12)$$

where  $\mathcal{R} = \{R_{t_i}, R_{d_i}, R_{c,t_i}, R_{c,d_i}\}$ .

The virtual view distortion  $D_s(x)$ , for  $0 \leq x \leq N-1$ , is modeled as a piecewise cubic polynomial using (8) such that the nearest left and right captured views indexed by integers  $i$  and  $i+1$ , respectively, are chosen as references and the corresponding cubic polynomial on the  $i$ th segment  $i \leq x \leq i+1$  is given by  $D_s^{(i)}(x) = m_{i,3}(x-i)^3 + m_{i,2}(x-i)^2 + m_{i,1}(x-i) + m_{i,0}$ . From Section 3.2, the cubic parameters  $m_{i,j}$ ,  $j = 0, 1, 2, 3$ , are estimated as

$$\begin{aligned} m_{i,3} &= 2s_t^2(1-\rho)(1-\alpha)(D_{d_i}^{1/2} - D_{d_{i+1}}^{1/2}), \\ m_{i,2} &= (1-\alpha)(D_{t_i} + D_{t_{i+1}}) - \\ &\quad 2s_t^2(1-\rho)((2-\alpha)D_{d_i}^{1/2} - (1-2\alpha)D_{d_{i+1}}^{1/2}), \\ m_{i,1} &= \alpha D_{t_{i+1}} - (2-\alpha)D_{t_i} + 2s_t^2(1-\rho)(D_{d_i}^{1/2} + \alpha D_{d_{i+1}}^{1/2}), \\ m_{i,0} &= D_{t_i}. \end{aligned}$$

### 4.3. Rate-Distortion Optimization

The goal of the optimization is to find the optimal bit allocation for  $\mathcal{R} = \{R_{t_i}, R_{d_i}, R_{c,t_i}, R_{c,d_i}\}$ , such that the resulting maximum of the distortion  $D_s(x)$ ,  $0 \leq x \leq N-1$ , is minimized given that the total bit rate  $R_{tot}$  from (12) does not exceed a bit rate constraint  $R_{max}$ . We formalize this goal as

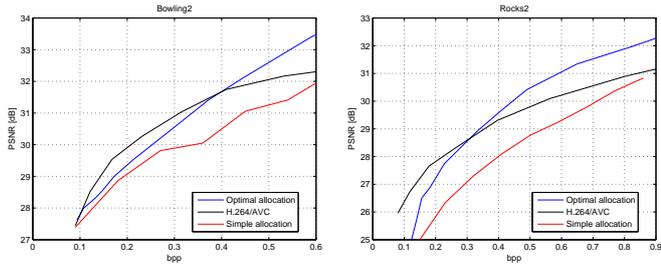
$$\mathcal{R}^* = \arg \min_{\mathcal{R}} \left\{ \max_{0 \leq x \leq N-1} D_s(x) \right\}, \text{ s.t. } R_{tot}(\mathcal{R}) \leq R_{max}. \quad (13)$$

To solve this min-max problem, we first compute the inner maximization in (13) by computing the maximum of each cubic polynomial  $D_s^{(i)}(x)$  for  $i \leq x \leq i+1$ . In particular, we find the roots  $x_i^\circ$ , if any, of the first derivative  $D_s^{(i)'}(x)$ , such that the second derivative  $D_s^{(i)''}(x_i^\circ) < 0$ . Since  $D_s^{(i)}(x)$  is cubic, there can be at most one such  $x_i^\circ$ . The maximum of each  $D_s^{(i)}(x)$  is then equal to  $D_{s,max}^{(i)} = \max\{D_s^{(i)}(i), D_s^{(i)}(x_i^\circ), D_s^{(i)}(i+1)\}$ , if  $i < x_i^\circ < i+1$ , or to  $D_{s,max}^{(i)} = \max\{D_s^{(i)}(i), D_s^{(i)}(i+1)\}$  otherwise. Finally, the maximum of the virtual view distortion  $D_{s,max}$  is obtained as the maximum of all  $D_{s,max}^{(i)}$ ,  $i = 0, \dots, N-1$ .

Having  $D_{s,max}$ , we convert the optimization problem in (13) into its Lagrangian unconstrained version, for a given Lagrange multiplier  $\lambda > 0$ , as follows

$$\mathcal{R}^\circ = \arg \min_{\mathcal{R}} D_{s,max} + \lambda R_{tot}. \quad (14)$$

It can be shown that the solution  $\mathcal{R}^\circ$  is equal to the solution  $\mathcal{R}^*$  of (13) for  $R_{max} = R_{tot}(\mathcal{R} = \mathcal{R}^\circ)$ . Notice that the search in (14) is mixed discrete-continuous because the bit rates  $R_{t_i}$  and  $R_{d_i}$  are modeled in the continuous domain, whereas  $R_{c,t_i}$  and  $R_{c,d_i}$  are sampled at discrete points. Thus, we separate this search into a two-step problem, such that for each sampled  $R_{c,t_i}$  and  $R_{c,d_i}$ , the optimal rates  $R_{t_i}$  and  $R_{d_i}$  are found by numerical minimization using the Matlab function `fmincon` with imposed nonnegative constraints. Then, the final solution  $\mathcal{R}^\circ$  is found by an exhaustive search over the set of sampled  $R_{c,t_i}$  and  $R_{c,d_i}$ .



**Fig. 6.** RD performance of the optimal rate allocation compression algorithm compared to the performance of a simple uniform allocation and H.264. The results are obtained for two data sets: *Bowling2* and *Rocks2*. The optimal allocation compression always outperforms the compression with the simple allocation and has a better RD performance than H.264 at mid- and high-range rates. However, at lower rates, the sophisticated motion compensation tools in H.264 have a key influence on the RD performance and thus lead to a better quality for the virtual views.

## 5. RESULTS

For experimentation, we encode texture and depth images of two Middlebury [11] data sets, *Bowling2* and *Rocks2*, using the SA-WT followed by SPIHT, as explained in Section 2. The data sets contain 7 texture and 2 depth images with resolution  $1110 \times 1330$  pixels for *Bowling2* and  $1110 \times 1276$  pixels for *Rocks2* captured at equidistant viewpoint locations with a baseline camera setup. The two viewpoints with available depth images are chosen as references, whereas the other texture images are used for verification of the virtual view quality.

The images are encoded at different bit rates determined by the optimal bit allocation algorithm. Then, the actual virtual view distortion is measured at the available intermediate points (i.e., the captured views without depth images) in between the two reference views and the maximal value is considered as an upper bound. Notice that this is only an approximation of the theoretical distortion upper bound; due to unavailability of images captured at arbitrary viewpoint locations, we use this approach as an alternative for the performance evaluation.

The resulting compression performance is compared to those of two other methods. In the first method, the images are encoded using the same compression method, but with a simple rate allocation such that an equal bit rate is allocated to each texture image and a half of that rate to each depth image. In the second method, the H.264/AVC [15] is applied across the images so that the temporal dimension is replaced with the view dimension. In both cases, the upper bound on the virtual view distortion is computed in the same way as for the optimal rate allocation. The related RD performances are shown in Fig. 6.

It can be seen that the optimal allocation algorithm always outperforms the uniform allocation, as expected, because of the applied optimization. Furthermore, the optimal algorithm performs better than the H.264 coder at mid-rates. However, at lower rates, the sophisticatedly designed motion compensation tools in H.264 have a key influence on the RD performance and, thus, result in a better quality of the virtual views. Since the goal of the present paper is to examine the use of our novel virtual view distortion model within a rather simple compression framework, we leave further improvements of the coder components of our system as future work.

## 6. CONCLUSION

We have derived a cubic distortion model for virtual view synthesis via depth-image-based rendering in multi-view imaging. The model accurately characterizes the signal distortion of a synthetic view, as a function of its location between the two reference views. We employ our model to optimize the bit allocation between captured depth and texture images in a multi-view imaging system, such that the maximum reconstructed distortion of a synthetic view is minimized for the given bit budget. We observed in our experiments that the proposed compression framework outperforms, with a significant gain, a heuristic rate allocation that uniformly distributes the available bit budget across all captured views. Encouraging performance advances are also observed over the state-of-the-art H.264 codec for medium and high encoding rates. We believe that further gains can be achieved if the other components of our compression framework are also optimized.

## 7. REFERENCES

- [1] P. Merkle, A. Smolic, K. Mueller, and T. Wiegand, "Multi-view video plus depth representation and coding," in *IEEE International Conference on Image Processing*, San Antonio, TX, October 2007.
- [2] C. Fehn, "Depth-image-based rendering (DIBR), compression and transmission for a new approach on 3D-TV," in *SPIE Stereoscopic Displays and Virtual Reality Systems XI*, San Jose, CA, January 2004.
- [3] T. Fujii and M. Tanimoto, "Free viewpoint TV system based on ray-space representation," in *Proceedings of SPIE*, 2002, vol. 4864, p. 175.
- [4] 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," *Signal Proc.: Image Comm.*, vol. 24, no. 1–2, pp. 73–88, 2009.
- [5] M. Maitre, Y. Shinagawa, and M.N. Do, "Wavelet-based joint estimation and encoding of depth-image-based representations for free-viewpoint rendering," in *IEEE Transactions on Image Processing*, June 2008, vol. 17, no.6, pp. 946–957.
- [6] S. Li and W. Li, "Shape-adaptive discrete wavelet transforms for arbitrarily shaped visual object coding," *IEEE Trans. on Circuits and Systems for Video Tech.*, vol. 10, pp. 725–743, 2000.
- [7] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. on Circuits and Systems for Video Tech.*, vol. 6, pp. 243–250, 1996.
- [8] H. Freeman, "On the encoding of arbitrary geometric configurations," *IRE Trans. of Electronic Computers*, vol. 10, no. 2, pp. 260–268, 1961.
- [9] G.-C. Chang and W.-N. Lie, "Multi-view image compression and intermediate view synthesis for stereoscopic applications," in *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, Geneva, Switzerland, May 2000.
- [10] S. Mallat, *A wavelet tour of signal processing*, Academic Press, 1997.
- [11] "2006 stereo datasets," <http://vision.middlebury.edu/stereo/data/scenes2006/>.
- [12] W.-S. Kim, A. Ortega, P. Lai, D. Tian, and C. Gomila, "Depth map coding with distortion estimation of rendered view," in *SPIE Visual Information Processing and Communication*, San Jose, CA, January 2010.
- [13] G. Cheung, V. Velisavljević, and A. Ortega, "On dependent bit allocation for multiview image coding & view synthesis," in *(submitted to) IEEE Transactions on Image Processing*, November 2010.
- [14] C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.
- [15] "The TML project web-page and archive," <http://kbc.cs.tu-berlin.de/~stewel/vceq/>.