Improving View Rendering Quality and Coding Efficiency by Suppressing Compression Artifacts in Depth-Image Coding

PoLin Lai*,†, Antonio Ortega*, Camilo C. Dorea†, Peng Yin† , and Cristina Gomila†

*Signal and Image Processing Institute, Univ. of Southern California, Los Angeles, CA 90089
†Thomson Corporate Research, 2 Independence Way, Princeton, NJ 08540

ABSTRACT

We consider the effect of depth-image compression artifacts on the quality of virtual views rendered using neighboring views. Such view rendering processes are utilized in new video applications such as 3D television (3DTV) and free viewpoint video (FVV). We first analyze how compression artifacts in compressed depth-images result in distortions in rendered views. We show that the rendering position error is a monotonic function of the coding error. For the scenario in which cameras are arranged with parallel optical axes , we further demonstrate specific properties of rendering position error. Exploiting special characteristics of depth-images, namely smooth regions separated by sharp edges, we investigate a possible solution to suppress compression artifacts by encoding depth-images with a recently published sparsity-based in-loop de-artifacting filter. Simulation results show that applying such techniques not only provides significantly higher coding efficiency for depth-image coding, but, more importantly, also improves the quality of rendered views in terms of PSNR and subjective quality.

Keywords: Depth compression, multiview video plus depth, de-artifact filter, view rendering

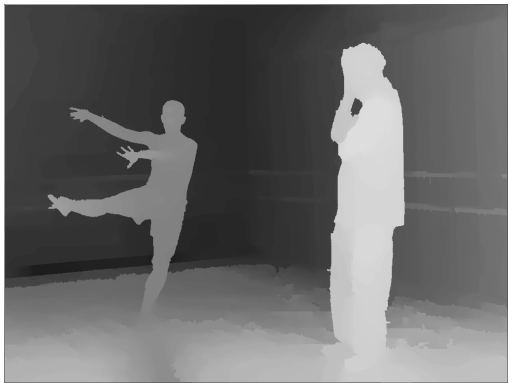
1. INTRODUCTION

With advances in computer vision, multiview video coding, and other related fields, recently, new video applications such as 3D television (3DTV) and free viewpoint video (FVV) have drawn wide attention. 3DTV aims to provide viewers depth perception of the scene by simultaneously rendering multiple images for different viewing angles using special displays. Instead, the goal of FVV is to allow viewers to select an arbitrary viewing angle of the scene. Currently, the most widely used video representation which enables 3DTV and FVV functionalities is the *multiview video plus depth* format (MVD). It consists of multiple video sequences (referred to as views), captured by synchronized cameras from different viewpoints, and the corresponding per pixel depth maps (referred to as depth-images). The pixel values in these gray-level depth-images represent depth Z in the range between Z_{near} and Z_{far}. At pixel location (x, y) in the depth image D we will have

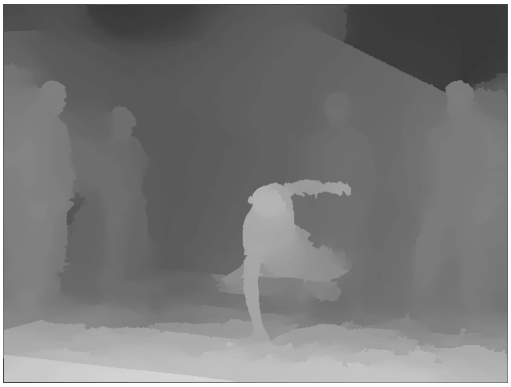
$$D(x,y) = 255 \cdot \frac{Z(x,y)^{-1} - (Z_{far})^{-1}}{(Z_{near})^{-1} - (Z_{far})^{-1}}, \tag{1}$$

Objects with smaller depths (closer to the camera) will appear brighter in the depth-image and farther away objects will be darker. Fig. 1 shows some examples of depth-images. Virtual views can be rendered from captured views using view-warping algorithms with the corresponding depth-images and camera parameters. In the following, we will refer to this process as view rendering.

The MVD format contains significant amounts of data and compression of MVD is essential in order to realize new 3DTV and FVV applications. For the video data, efficient multiview video coding schemes can be achieved by combining new coding tools for multiview scenarios[1] with existing techniques for conventional video coding. As for depth-images, they are not viewed by users directly, and instead they are used to render virtual views. Thus, the effect of lossy depth-image compression on view rendering should be studied. In addition to considering conventional rate-distortion criteria, the performance of depth-image coding should also be evaluated based on the quality of rendered views. For this purpose, there are two PSNR/MSE (mean-squared error) metrics employed in the literature. Let us denote D and $\hat{D}$ the original depth-image and the compressed one, respectively. Let $V_R(D)$, $V_R(\hat{D})$ denote the rendered views using D and $\hat{D}$, respectively. The first metric, denoted PSNR$(V_R(\hat{D}), V_R(D))$, calculates the PSNR between $V_R(\hat{D})$ and $V_R(D)$ [2]. This metric *implicitly assumes* that using uncompressed



(a) Ballet, depth-image V4 frame 0



(b) Breakdancers, depth-image V4 frame 0

Figure 1. Examples of depth-image

depth-images leads to the best rendering quality and thus $V_R(D)$ is used as a reference to evaluate different $\hat{D}$: higher PSNR indicates $V_R(\hat{D})$ is closer $V_R(D)$. However, when the original depth-image D is estimated (e.g. via disparity estimation) instead of obtained using active systems such as range camera, there could be some distortion in D, and thus $V_R(D)$ may not serve as a good reference for rendering quality. For example in Fig. 1, we can see some false contours in the floor area. Processing such region could lead to improved rendering quality, while it will decrease PSNR$(V_R(\hat{D}), V_R(D))$ as the depth-image is "altered". On the other hand, the second PSNR metric is calculated between $V_R(\hat{D})$ and the *ground truth* V_C, i.e., the corresponding view captured at the same location and orientation chosen for rendering[3] (denoted as PSNR$(V_R(\hat{D}), V_C)$.) For example in a MVD system, one can use compressed video and depth-images from View 0 and View 2 to render View 1 and evaluate PSNR against the captured View 1. This metric directly reflects the fidelity of the rendered view. In this paper, we will use this metric to measure the rendering quality.

Using PSNR$(V_R(\hat{D}), V_R(D))$, the effect of MVD compression has been investigated by changing the bitrate ratio between multiview video and the depth-images[4]. The results indicate that, on rendered views, the quality of compressed depth-images has much stronger impact than the quality of compressed video. The compression artifacts in depth-images result in significant distortion around depth discontinuities, which occur at boundaries of objects with different depths. To better preserve these boundaries, the authors suggested processing the coded depth-images, and/or designing other coding techniques. Y. Morvan *et al.*[5] proposed an alternative intra coding approach, called "platelet-based coding", to encode depth-images. Considering the fact that depth-images are typically made up with smooth regions separated by depth discontinuities, the authors employed a quadtree decomposition to partition depth-images into blocks with variable sizes and represent them with piecewise linear functions (platelets). It was reported that, in certain low-bitrate scenarios, depth-images encoded with platelet-based method provide higher PSNR for the rendered views (again using PSNR$(V_R(\hat{D}), V_R(D))$) as compared to encoded with H.264 Intra (i.e. PSNR$(V_R(\hat{D}_{platelet}), V_R(D)) >$ PSNR$(V_R(\hat{D}_{H.264I}), V_R(D))$). However, the coding efficiency of the platelet-based scheme on depth-images, is considerably lower than the coding efficiency of H.264 Intra[6]. In this paper, we employ video coding tools (e.g. H.264 with IBBP structure) with the sparsity-based de-artifacting filtering approach in [7, 8] to encode depth-images. It is an in-loop filtering scheme such that de-artifacting is performed after encoding each frame, and the processed frames (with improved PSNR) serve as references for predictive coding. We will demonstrate that this approach improves coding efficiency while also increasing the quality of rendered views.

The remainder of the paper is organized as follows: In Section 2, we first analyze how compression artifacts in coded depth-images result in distortions in rendered views and demonstrate that the rendering position error is a monotonic function of the coding error. The sparsity-based in-loop de-artifacting filter approach is described in Section 3. Simulations results on coding efficiency and quality of rendered views (measured against the ground truth, i.e., PSNR$(V_R(\hat{D}), V_C)$), are provided in Section 4. Finally, conclusions are drawn in Section 5

2. EFFECT OF DEPTH-CODING ARTIFACTS ON VIEW RENDERING

Fig. 1 shows depth-image examples. As compared to natural images, they consist of relatively *smooth regions separated by depth discontinuities*, which occur at boundaries of objects with different depths. It is well known that video coding schemes (intra/inter) introduce artifacts especially around sharp edges, due to the quantization of DCT coefficients of the prediction residue. In Fig. 2(d), we observe such typical artifacts in compressed depth-images. In this section, we analyze how compression artifacts exhibited in coded depth-images affect the quality of rendered views.

Consider an example in which we want to use the view captured by a camera at position p, and the corresponding depth-image $D_p(x,y)$, to render a virtual view as if it were captured by a camera at position p'. The per-pixel depth values $Z_p(x,y)$ can be calculated from $D_p(x,y)$, using (1):

$$\frac{1}{Z_p(x,y)} = \frac{D_p(x,y)}{255} \cdot \left(\frac{1}{Z_{near}} - \frac{1}{Z_{far}} \right) + \frac{1}{Z_{far}} \tag{2}$$

After obtaining $Z_p(x,y)$, a view-warping from p to p' based on geometric camera parameters[9] can be carried out as a two-step process*. First, the pixel position (x,y) at the frame coordinate system of camera at p, is projected into a world coordinate $[u,v,w]^{\mathrm{T}}$ (column vector, 3×1). With intrinsic parameter matrix $\mathbf{A}_p$ (3×3) of the camera at p, rotation matrix $\mathbf{R}_p$ (3×3) and translation vector $\mathbf{T}_p$ (3×1) which relate the camera coordinate system to the global coordinate system†, this projection can be represented as:

$$[u,v,w]^{\mathrm{T}} = \mathbf{R}_p \cdot \mathbf{A}_p^{-1} \cdot [x,y,1]^{\mathrm{T}} \cdot Z_p(x,y) + \mathbf{T}_p \tag{3}$$

Then, $[u,v,w]^{\mathrm{T}}$ is mapped to coordinate $[x',y',z']^{\mathrm{T}}$ of the virtual camera at p' with $\mathbf{A}_{p'}$ $\mathbf{R}_{p'}$ and $\mathbf{T}_{p'}$:

$$[x',y',z']^{\mathrm{T}} = \mathbf{A}_{p'} \cdot \mathbf{R}_{p'}^{-1} \cdot \left\{ [u,v,w]^{\mathrm{T}} - \mathbf{T}_{p'} \right\} \tag{4}$$

The corresponding pixel location in the rendered image of camera p' is $(\frac{x'}{z'}, \frac{y'}{z'})$.

Now assume that the depth-image is compressed, resulting in $\hat{D}_p(x,y) = D_p(x,y) + \Delta D_p(x,y)$, where $\Delta D_p(x,y)$ is an error term due to compression. This will lead to a distortion in the calculated depth with $\hat{Z}_p(x,y) = Z_p(x,y) + \Delta Z_p(x,y)$, i.e.,

$$\frac{1}{Z_p(x,y) + \Delta Z_p(x,y)} = \frac{D_p(x,y) + \Delta D_p(x,y)}{255} \left(\frac{1}{Z_{near}} - \frac{1}{Z_{far}} \right) + \frac{1}{Z_{far}} \tag{5}$$

By combining (3) and (4), we can derive that, due to $\Delta Z_p(x,y)$, the resulting warped coordinate in p' will be affected by an additive error $[\Delta x', \Delta y', \Delta z']^{\mathrm{T}}$ as :

$$[x',y',z']^{\mathrm{T}} = \mathbf{A}_{p'} \cdot \mathbf{R}_{p'}^{-1} \cdot \left\{ \mathbf{R}_p \cdot \mathbf{A}_p^{-1} \cdot [x,y,1]^{\mathrm{T}} \cdot Z_p(x,y) + \mathbf{T}_p - \mathbf{T}_{p'} \right\} \tag{6}$$

$$[x'+\Delta x', y'+\Delta y', z'+\Delta z']^{\mathrm{T}} = \mathbf{A}_{p'} \cdot \mathbf{R}_{p'}^{-1} \cdot \left\{ \mathbf{R}_p \cdot \mathbf{A}_p^{-1} \cdot [x,y,1]^{\mathrm{T}} \cdot (Z_p(x,y) + \Delta Z_p(x,y)) + \mathbf{T}_p - \mathbf{T}_{p'} \right\} \tag{7}$$

Thus in the rendered frame corresponding to p', the pixel at (x,y) in p will no longer be mapped to $(\frac{x'}{z'}, \frac{y'}{z'})$ and instead will be mapped to a wrong location $\left(\frac{x'+\Delta x'}{z'+\Delta z'}, \frac{y'+\Delta y'}{z'+\Delta z'} \right)$. We define the rendering position error ($\mathbf{\Delta P}$) for a given pixel as follows:

*While a particular view-warping algorithm may not follow exactly the procedure described here, the equations (3) (4) are the basic principles that most of the view-warping methods based on.

†These parameters are all provided along with the MVD data set used in this paper.

$$\mathbf{\Delta P} = \left(\frac{x' + \Delta x'}{z' + \Delta z'}, \frac{y' + \Delta y'}{z' + \Delta z'}\right) - (\frac{x'}{z'}, \frac{y'}{z'}) \tag{8}$$

To analyze how $\mathbf{\Delta P}$ is affected by the compression error ΔD, let us define the following terms:

$$\begin{aligned} \text{Define:} \qquad \mathbf{M} &= \mathbf{A}_{p'} \cdot \mathbf{R}_{p'}^{-1} \cdot \mathbf{R}_p \cdot \mathbf{A}_p^{-1} \cdot [x, y, 1]^{\mathrm{T}} \\ \mathbf{N} &= \mathbf{A}_{p'} \cdot \mathbf{R}_{p'}^{-1} \cdot (\mathbf{T}_p - \mathbf{T}_{p'}) \\ S &= \frac{1}{255}\left(\frac{1}{Z_{near}} - \frac{1}{Z_{far}}\right) \end{aligned}$$

$\mathbf{M} = [M_1, M_2, M_3]^{\mathrm{T}}$ and $\mathbf{N} = [N_1, N_2, N_3]^{\mathrm{T}}$ are 3×1 vectors. Then (5) and (7) can be represented as:

$$Z_p + \Delta Z_p = \left(S(D + \Delta D) + \frac{1}{z_{far}}\right)^{-1} \tag{9}$$

$$[x' + \Delta x', y' + \Delta y', z' + \Delta z']^{\mathrm{T}} = \mathbf{M} \cdot (Z_p + \Delta Z_p) + \mathbf{N} \tag{10}$$

Taking derivative of the x component of $\mathbf{\Delta P}$ with respect to ΔD, we get (y component can be analyzed in the same manner):

$$\frac{\partial}{\partial(\Delta D)}\left(\frac{x' + \Delta x'}{z' + \Delta z'} - \frac{x'}{z'}\right) = \frac{\partial}{\partial(Z_p + \Delta Z_p)}\left(\frac{x' + \Delta x'}{z' + \Delta z'} - \frac{x'}{z'}\right) \cdot \frac{\partial(Z_p + \Delta Z_p)}{\partial(\Delta D)} \tag{11}$$

With (9) and (10), the derivative of (11) can be calculated from the following two terms:

$$\begin{aligned} \frac{\partial}{\partial(Z_p + \Delta Z_p)}\left(\frac{x' + \Delta x'}{z' + \Delta z'}\right) &= \frac{M_1(z' + \Delta z') - (x' + \Delta x')M_3}{(z' + \Delta z')^2} \\ &= \frac{M_1\left(M_3(Z_p + \Delta Z_p) + N_3\right) - \left(M_1(Z_p + \Delta Z_p) + N_1\right)M_3}{\left(\quad M_3(Z_p + \Delta Z_p) + N_3 \quad\right)^2} \\ &= \frac{M_1 N_3 - N_1 M_3}{\left(M_3 \cdot \left(S(D + \Delta D) + \frac{1}{z_{far}}\right)^{-1} + N_3\right)^2} \end{aligned} \tag{12}$$

$$\frac{\partial(Z_p + \Delta Z_p)}{\partial(\Delta D)} = \frac{\partial}{\partial(\Delta D)}\left(S(D + \Delta D) + \frac{1}{z_{far}}\right)^{-1} = -S \cdot \left(S(D + \Delta D) + \frac{1}{z_{far}}\right)^{-2} \tag{13}$$

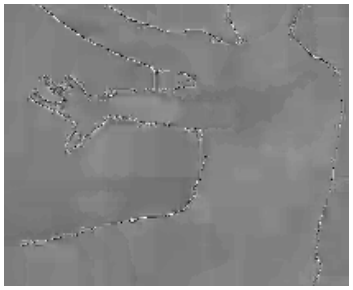
Finally, by multiplying (12) and (13), we get the derivative $\frac{\partial(\mathbf{\Delta P}_x)}{\partial(\Delta D)}$:

$$\frac{\partial}{\partial(\Delta D)}\left(\frac{x' + \Delta x'}{z' + \Delta z'} - \frac{x'}{z'}\right) = \frac{-S \cdot (M_1 N_3 - N_1 M_3)}{\left(M_3 + N_3 \cdot \left(S(D + \Delta D) + \frac{1}{z_{far}}\right)\right)^2} \tag{14}$$

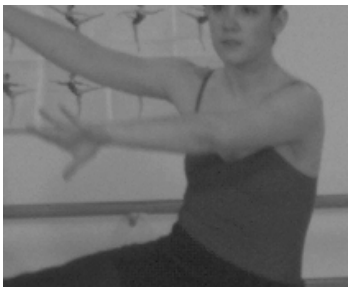
From (14), some important properties of how ΔD affects $\mathbf{\Delta P}$ can be derived:

- The sign of the derivative of $\mathbf{\Delta P}$ is the same regardless of the value of ΔD. This indicates that $\mathbf{\Delta P}$ is a *monotonic function of* ΔD.

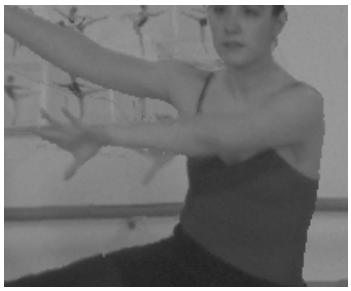
(b) Original depth-image D_{V0}



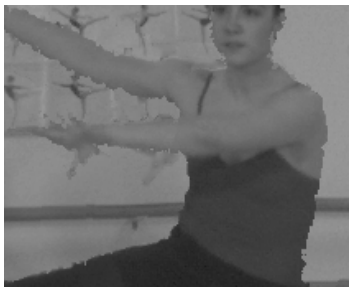
(d) Coding error $\Delta D_{V0} = \hat{D}_{V0} - D_{V0}$ (×3+128)



(a) Captured V1



(c) Synthesized V1 with original depth D_{V0} and D_{V2}



(e) Synthesized V1 with coded depth $\hat{D}_{V0}$ and $\hat{D}_{V2}$

Figure 2. Effect of coding artifact on rendered views: Ballet, frame 7, depth coded with H.264 INTER, QP32

- The derivative of $\mathbf{\Delta P}$ is not symmetric with respect to ΔD, i.e. putting $\Delta D = +\delta$ and $\Delta D = -\delta$ in (14) does not produce derivatives with the same magnitude. Thus for a particular pixel, a positive error $\Delta D = +\delta$ in the depth will *not* produce the same rendering position error, as a negative error $\Delta D = -\delta$ of same magnitude.
- From the term S, we can see that, other things being equal, a scene with depth range $[Z_{near}, Z_{far}]$ leading to larger $\left(\frac{1}{Z_{near}} - \frac{1}{Z_{far}}\right)$ will be more sensitive to the depth coding error.

From this analysis, if we separately consider positive and negative coding errors, *the deviation of a projected pixel from its correct position will increase monotonically with the absolute value of* ΔD. In depth-images, regions that are prone to result in large coding errors, such as sharp edges and boundary of fast moving objects, will lead to stronger artifacts in the rendered views. Fig. 2(e) demonstrates an example when the view is rendered with a compressed depth-image, thus exhibiting rendering errors along the boundary of the dancer (as compared with Fig. 2(c) where the depth is not compressed). Note that, from the analysis above, the relationship between depth coding error ΔD and the rendering position error $\Delta \mathbf{P}$ is, *in general, non-linear* (Derivative (14) is not a constant with respect to ΔD). In the following sub-section, we will analyze this relationship under the most commonly used setting in a multiview system, in which the cameras are arranged/oriented on the same vertical plane with their optical axes being parallel (referred to as a parallel camera arrangement, see Fig. 3).

2.1. Effect of depth coding under parallel camera arrangement

For multiview systems with a parallel camera arrangement, we can construct the world coordinate system as in Fig. 3 such that the view-warping in (6) can be simplified with $\mathbf{R}_p = \mathbf{R}_{p'} = \mathbf{I}$, $\mathbf{A}_p = \mathbf{A}_{p'}$ (assuming same intrinsic camera parameters), and $\mathbf{T}_p - \mathbf{T}_{p'} = [\delta x, 0, 0]^{\mathrm{T}}$, therefore:

$$[x', y', z']^{\mathrm{T}} \quad = \quad [x, y, 1]^{\mathrm{T}} \cdot Z_p(x, y) + \mathbf{A}_{p'} \cdot [\delta x, 0, 0]^{\mathrm{T}} \tag{15}$$

$$\text{which leads to} \quad \left(\frac{x'}{z'}, \frac{y'}{z'}\right) = \left(\frac{xZ_p(x,y) + a \cdot \delta x}{Z_p(x,y)}, y\right) = \left(x + \frac{a \cdot \delta x}{Z_p(x,y)}, y\right) \tag{16}$$

In (16), $a = \mathbf{A}_{p'}(1,1)$, the first element of the matrix $\mathbf{A}_{P'}$. Note that (16) indicates that for a parallel camera arrangement, the projected pixel in p' is simply shifted from (x, y) with a disparity vector that is proportional to camera distance δx and inversely proportional to its depth $Z_P(x, y)$. When the depth-image has been encoded and we use $\hat{Z}_p(x, y) = Z_p(x, y) + \Delta Z_p(x, y)$ for view rendering, the new coordinates can be represented as:

$$[x' + \Delta x, y' + \Delta y, z' + \Delta z]^{\mathrm{T}} = [x, y, 1]^{\mathrm{T}} \cdot (Z_p(x,y) + \Delta Z_p(x,y)) + \mathbf{A}_{p'} \cdot [\delta x, 0, 0]^{\mathrm{T}} \tag{17}$$

$$\left(\frac{x' + \Delta x}{z' + \Delta z}, \frac{y' + \Delta y}{z' + \Delta z}\right) = \left(x + \frac{a \cdot \delta x}{Z_p(x,y) + \Delta Z_p(x,y)}, y\right) \tag{18}$$

The rendering position error can then be calculated using (5) and (2) as:

$$\begin{aligned} \frac{x' + \Delta x}{z' + \Delta z} - \frac{x'}{z'} &= \frac{a \cdot \delta x}{Z_p(x,y) + \Delta Z_p(x,y)} - \frac{a \cdot \delta x}{Z_p(x,y)} \\ &= a \cdot \delta x \cdot \frac{\Delta D_p(x,y)}{255} \left(\frac{1}{Z_{near}} - \frac{1}{Z_{far}}\right) \end{aligned} \tag{19}$$

Some interesting properties of the rendering error for parallel camera arrangement can be observed from (19). These can be summarized as follows:

- The rendering position error is *proportional to the depth coding error ΔD (linear).*
- The greater the distance δx between cameras is, the larger the rendering position error will be. In other words, systems with dense camera setting will be more robust to the effect of depth coding.

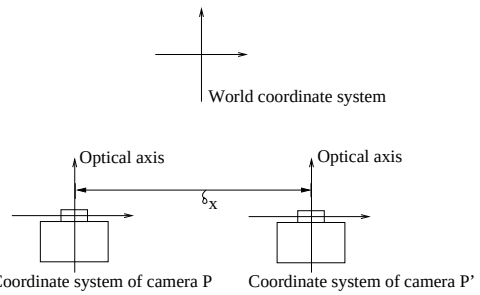


Figure 3. Parallel camera arrangement, with δx between cameras

3. SUPPRESSING COMPRESSION ARTIFACTS IN DEPTH-IMAGE CODING

From the analysis in the previous section, the quality of rendered views will be improved if we can reduce the compression errors when encoding depth-images. Since the rendering position error increases monotonically as the coding error increases, regions more likely to result in significant errors due to compression, such as depth edges, should receive more attention. While various de-artifacting/denoising techniques could potentially be utilized to remove coding artifacts while perserving edges, we exploit special characteristics of depth-images in developing/selecting a proper method. The fact that depth-images are made of smooth regions separated by depth edges (discontinuities) matches very well the assumptions which underpin the design of sparsity-based non-linear in-loop filters [10, 11]. The underlying principle of these methods is that, based on a sparse image

model, compact representations of images can be obtained via transforms. After applying transforms to an encoded image, denoising is first performed with thresholding operations on transform coefficients. Furthermore, to improve denoising results, an overcomplete set of transforms is constructed with all possible translations of a particular transform and weighting combination is performed in a way favoring translated transforms with stronger sparseness. In the following, we describe the basic procedure using DCT [11]:

Consider an $N \times N$ block size DCT. An overcomplete set of transforms $\mathbf{H}_i$ is generated by using all possible translations of the $N \times N$ block DCT (hence $i \in 0, 1, ..., N \times N - 1$). For a coded image I, a transformed version is obtained as $C_{I_i} = \mathbf{H}_i I$ for each $\mathbf{H}_i$. Under the assumption that images tend to be sparse in transform domain, the first denoising step is performed by thresholding small coefficients in C_{I_i} to zero, resulting in $\hat{C}_{I_i}$. Then, **a set of denoised estimations of** I are constructed using $\mathbf{H}_i^{-1}\hat{C}_{I_i}$. The second denoising step, which also exploits a sparse image model, generates a final estimation $\hat{I}$ by a weighted averaging method:

$$\hat{I}(x,y) = \sum_{i=0}^{N\times N-1} W_i(x,y) \cdot \mathbf{H}_i^{-1}\hat{C}_{I_i} \text{ , where } W_i(x,y) = \frac{k(x,y)}{||\hat{C}_{I_i}(x,y)||_0} \tag{20}$$

In (20), for each pixel (x, y), the weights $W_i(x, y)$ are inversely proportional to the number of non-zero coefficients in the transform domain that contribute to pixel (x, y) ($||\hat{C}_{I_i}(x, y)||_0$). $k(x, y)$ is a scalar chosen so as to ensure that $\sum_{i=0}^{N\times N-1} W_i(x, y) = 1$. In this manner, transforms $\hat{C}_{I_i}(x, y)$ that have stronger sparseness are given larger weights in (20). The filtered image $\hat{I}$ achieves improved PSNR and will then be used as reference for encoding the following frames.

Starting from the method above, recent work [7,8] further improves the de-artifacting filtering results by proposing the following:

1. For video coding, block DCT is also utilized for coding prediction residues. After quantization on the DCT coefficients, the resulting block may exhibit "artificially large sparseness" which allows some artifacts to persist when computing the weights as in (20). To tackle this problem, in the weighed averaging procedure specified by (20), DCT translations $\mathbf{H}_i$ which are spatially aligned with the DCT blocks for residue coding, should be excluded. For instance, $\mathbf{H}_i$ within $\pm$ 1 pixel of alignment to residue coding DCT block in x/y directions are avoided [7,8]. It has been demonstrated that higher PSNR is achieved for the filtered images when excluding these DCT translations [7].

2. DCT is not very efficient in representing signals that are neither vertically nor horizontally oriented (for example, diagonal). Thus, in addition to DCTs on the regular sampling grid, DCTs operate on the quincunx sampling of a picture (i.e. grid rotated by 45 degrees) are also utilized in the de-noising/de-artifacting process [7,8]. Fig. 4 illustrates quincunx sampling on image pixels, resulting in two complementary cosets of samples (o and x) arranged in diagonal directions. For each coset, a translated DCT, $\mathbf{H}_k$, are applied on the quincunx sampling grid (rotated coordinate) of the image.

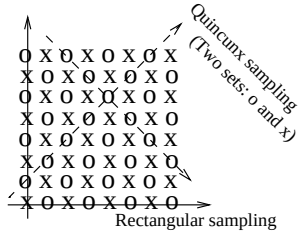


Figure 4. Quincunx sampling of a rectangular grid: o and x represent two complementary cosets

Combining with item 1 above, the new weighted averaging to obtain the estimation $\hat{I}$ becomes:

$$\hat{I}(x,y) = \sum_{i=0}^{M} W_i(x,y) \cdot \mathbf{H}_i^{-1} \hat{C}_{I_i} + \sum_{i=0}^{N\times N-1} W_j^q(x,y) \cdot \left(\mathbf{H}_j^{-1} \hat{C}_{I_j}^{q1} \oplus \mathbf{H}_j^{-1} \hat{C}_{I_j}^{q2} \right), \quad (21)$$

Compared to (20), $M < N \times N - 1$ as DCT translations $\mathbf{H}_i$ closely aligned with residue coding DCTs are excluded. $\hat{C}_{I_j}^{q1}$ and $\hat{C}_{I_j}^{q2}$ are denoised (thresholded) DCT coefficients applied to each quincunx coset ($q1$ and $q2$, o and x as in Fig. 4). Operator $\oplus$ in (21) represents the merging operation of both cosets (i.e., rotation and upsampling) to obtain pixels in the original rectangular sampling grid. Note that as shown in Fig. 4, each pixel can only be in one of the quincunx cosets. $W_j^q(x,y) = W_j^{q1}(x,y) \oplus W_j^{q2}(x,y)$ where the weights W_j^{q1} and W_j^{q1} are calculated as in (20). The exclusive-or $\oplus$ here reflects the fact that a pixel is either in $q1$ or in $q2$. Finally, these weight must satisfy $\sum_{i=0}^{M} W_i(x,y) + \sum_{j=0}^{N\times N-1} W_j^q(x,y) = 1$

3. To better adapt to different spatial characteristics of quantization noise due to varying types of coding/prediction modes as well as the temporal variations in signal characteristics, for each frame, pixels are classified into different classes and a threshold will be determined for each class to obtain $\hat{C}_{I_i}$ from C_{I_i} (thresholding on DCT coefficients). The classification is performed based on the coding mode such as INTRA, INTER, the presence or absence coding residue, and whether the pixel is on the boundary of an encoded block (refer to Dorea *et al* [8] for details.) For pixels in a given class, thresholds from a pre-defined set are tested and the one achieving highest PSNR will be selected. These threshold selections are transmitted to the decoder in order to perform the same de-artifacting filtering process.

In this paper, we utilize the method by Dorea *et al* [8], which includes all the techniques listed above, to perform de-artifacting filtering when encoding depth-images. Note that the method we investigated in this paper is one possible approach to treat depth coding artifacts. Other techniques, which consider the special properties of depth-images to remove coding artifacts and preserve edges, are also worth being evaluated.

4. SIMULATION RESULTS

We conduct simulations using the abovementioned de-artifacting filtering approach [8], which is embedded within H.264/AVC. Using High Profile with IBBP structure, we encoded 100 frames of depth-images of V0 V2 V4 for Ballet and Breakdancers sequences. Four different rate-points are obtained with QP=$\{22, 27, 32, 37\}$. The results are compared against H.264/AVC without de-artifacting filtering. Table 1 provides the average efficiency gains for depth-images of each view, calculated according to [12]. Fig. 5 shows the rate-distortion (RD) performance. It can be seen that, at a given QP, de-artifacting filtering achieves significantly higher PSNR. In Fig. 6, we can observe the improvement around edges in depth-images.

We then use the view-synthesis software provided by Nagoya University[13] to render V1 with compressed depth-images from V0/V2 and render V3 from V2/V4. (Note that to focus on the effect of depth coding, in these simulations, the video sequences V0/2/4 are not compressed and are used directly for rendering.) In Fig. 7 we provide PSNR of rendered views, calculated against the captured (ground truth) V1 and V3 (i.e. PSNR$(V_R(\hat{D}), V_C)$ as described in Section 1). It can be seen that for all tested cases and across different QPs, applying de-artifacting filtering achieves higher fidelity in the rendered views (closer to captured view V_C). Comparing Fig. 8(a) and (b), an improvement in rendering quality can be clearly observed, especially along the boundary of the dancer.

Table 1. Average PSNR gains and bitrate savings (Calculated according to [12])

Sequence	Avg. PSNR gain	Avg. bitrate saving	Sequence	Avg. PSNR gain	Avg. bitrate saving
Ballet V0	0.794 dB	13.64%	Breakdancers V0	0.401 dB	7.50%
Ballet V2	0.856 dB	13.67%	Breakdancers V2	0.425 dB	7.72%
Ballet V4	0.863 dB	13.52%	Breakdancers V4	0.489 dB	7.41%

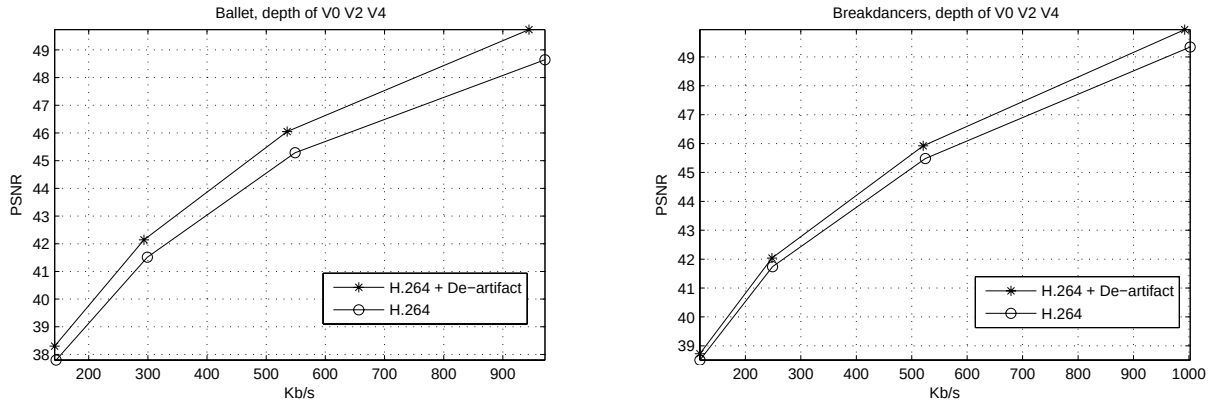


Figure 5. Rate-distortion performance of different depth coding schemes

(a) Compressed $\hat{D}_{V0}$, H.264

(b) Compressed $\hat{D}_{V0}$, H.264 with De-artifacting

Figure 6. Encoded depth-image, Ballet V0 frame 7

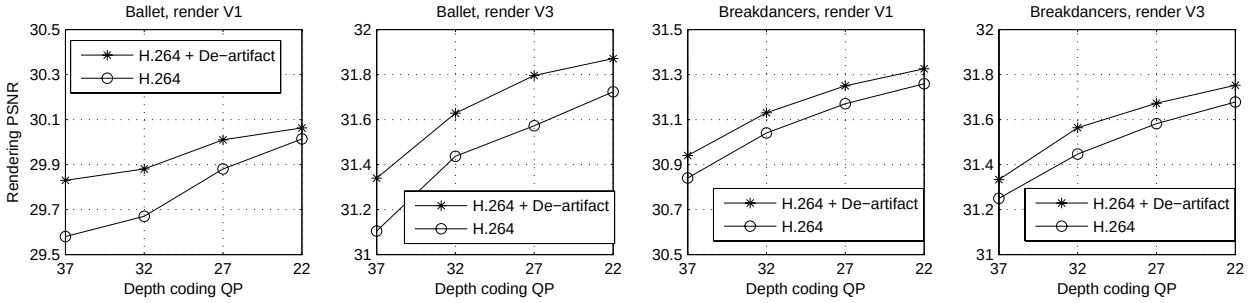


Figure 7. PSNR of rendered views

5. CONCLUSIONS

In this paper, we analyze the effect of compression artifacts in compressed depth-images resulting in distortions in rendered views. We show that the rendering position error is a monotonic function of the coding error. Under parallel camera arrangement, we demonstrate that the rendering position error is proportional to the depth coding error. For piecewise smooth signals like the depth-images, larger coding error occurs around depth edges. We utilize a sparsity-based de-artifacting in-loop filtering approach, which was developed with assumptions that fit well for the characteristics of depth-images, to suppress coding artifacts. Simulation results show that, applying

(a) Rendered from depth coded with H264 inter−P QP32

(b) Rendered from depth coded with H264 inter−P QP32 + De−artifact

Figure 8. Comparison between rendered views: Ballet rendered V1 from V0/V2, frame 7

such techniques on depth-image coding not only achieves higher coding efficiency, but also improves the fidelity of rendered views in terms of PSNR as well as subjective quality.

REFERENCES

1. IEEE, "Special Issue on Multi-view Video Coding and 3DTV," *IEEE Trans. Circuits Systems and Video Technologies (CSVT)* **17, no. 11**, Nov 2007.
2. R. Krishnamurthy, B. B. Chai, H. Tao, and S. Sethuraman, "Compression and transmission of depth maps for image-based rendering," in *Proc. IEEE 2001 International Conference on Image Processing (ICIP)*, pp. III.828–III.831, (Thessaloniki, Greece), Oct 2001.
3. Y. Morvan, D. Farin, and P. H. N. de With, "Joint depth/texture bit-allocation for multi-view video compression," in *Proc. 2007 Picture Coding Symposium (PCS)*, (Lisbon, Portugal), Nov 2007.
4. P. Merkle, A. Smolic, K. Muller, and T. Wiegand, "Multi-view video plus depth representation and coding," in *Proc. IEEE 2007 International Conference on Image Processing (ICIP)*, pp. I.201–I.204, (San Antonio, TX, USA), Sep 2007.
5. Y. Morvan, P. H. N. de With, and D. Farin, "Platelet-based coding of depth maps for the transmission of multiview images," in *Proc. SPIE 2006 Visual Communications and Image Processing (VCIP)*, (San Jose, CA, USA), Jan 2006.
6. P. Merkle, Y. Morvan, A. Smolic, D. Farin, K. Muller, P. de With, and T. Wiegand, "The effect of depth compression on multiview rendering quality," in *Proc. 3DTV Conference*, (Istanbul, Turkey), May 2008.
7. O. Divorra Escoda, P. Yin, and C. Gomila, "A multi-lattice direction-adaptive deartifacting filter for image and video coding," in *Proc. 2007 Picture Coding Symposium (PCS)*, (Lisbon, Portugal), Nov 2007.
8. C. C. Dorea, O. Divorra Escoda, P. Yin, and C. Gomila, "A direction-adaptive in-loop deartifacting filter for video coding," in *Proc. IEEE 2008 International Conference on Image Processing (ICIP)*, (San Diego, CA, USA), Oct 2008.
9. D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, Prentice Hall, 2003.
10. O. G. Guleryuz, "Weighted overcomplete denoising," in *Proc. Asilomar Conference on Signals and Systems*, (Pacific Grove, CA, USA), Nov 2003.
11. O. G. Guleryuz, "A nonlinear loop filter for quantization noise removal in hybrid video compression," in *Proc. IEEE 2005 International Conference on Image Processing (ICIP)*, pp. II.333–II.336, (Genoa, Italy), Sep 2005.
12. G. Bjontegaard, "Calculation of average PSNR differences between RD-curves,," *Tech. Rep., VCEG-M33, ITU-T Q.6/SG16* , Apr 2001.
13. *http://www.tanimoto.nuee.nagoya-u.ac.jp/english/index.html Tanimoto Lab, Dept. of Information Electronics, Nagoya University* .