Adaptive Block Truncation Filter for MVC Depth Image Enhancement

Xuyuan Xu[1], Lai-Man Po[1], Chun-Ho Cheung[2], Litong Feng[1], Kwok-Wai Cheung[3], Chi-Wang Ting[1], Ka-Ho Ng[1]

1. Department of Electronic Engineering, City University of Hong Kong, Hong Kong SAR, China
2. Department of Information Systems, City University of Hong Kong, Hong Kong SAR, China
3. Department of Computer Science, Chu Hai College of Higher Education, Hong Kong SAR, China

ABSTRACT

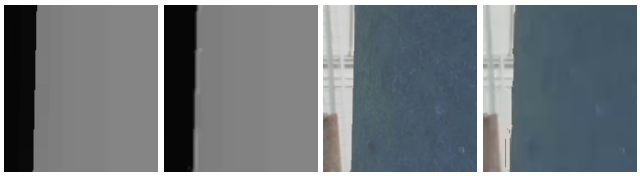
In Multiview Video plus Depth (MVD) format, virtual views are generated from decoded texture videos with decoded depth images through Depth Image based Rendering (DIBR). 3DV-ATM is a reference model for H.264/AVC based Multiview Video Coding (MVC) and aims at achieving high coding efficiency for 3D video in MVD format. Depth images are first downsampled then coded by 3DV-ATM. However, sharp object boundary characteristic of depth images does not well match with the transform coding of 3DV-ATM. Depth boundaries are often blurred with ringing artifacts in the decoded depth images that result in noticeable artifacts in synthesized views. This paper presents a low complexity adaptive block truncation filter to recover the sharp object boundaries of depth images using adaptive block repositioning and expansion for increasing the depth values refinement accuracy. This new approach is very efficient and can avoid false depth boundary refinement when block boundaries lie around the depth edge regions and ensure sufficient information within the processing block for depth layers classification. Experimental results show that sharp depth edges can be recovered using the proposed filter and boundary artifacts in the synthesized views can be removed. The proposed method can provide improvement up to 3.25dB in the depth map enhancement and bitrate reduction of 3.06% in the synthesized views.

Index Terms— *Multiview Video plus Depth (MVD), Depth Image based Rendering (DIBR), 3DV-ATM, Multiview Video Coding (MVC), depth image compression, depth image filter.*

1. INTRODUCTION

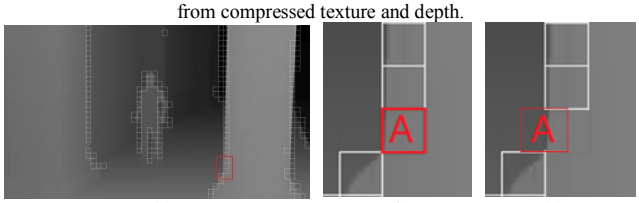
In the last two decades, 3D video (3DV) technology advances significantly. It evolves from the simple stereoscopic system to more realistic multiview video systems, such as autostereoscopic 3DV displays and free-view TV [1, 2]. However, realization of multiview systems using large number of texture video views is not efficient. Depth-enhanced video format such as multiview video plus depth (MVD) format has attracted much attention due to its high efficiency and compatibility with conventional multiview video systems. Instead of coding numerous views, two or three texture views with their corresponding depth images are used in Multiview Video Coding (MVC) standard. Virtual views can be generated by specifying the preference viewing angles through the Depth Image based Rendering (DIBR) [3]. There are two reference test models for 3DV codecs, 3DV-ATM [4] and 3DV-HTM [5], which are based on H.264/AVC and H.265/HEVC, respectively.

In 3DV-ATM, depth images are downsampled by half in both horizontal and vertical directions using the MPEG 13-tap downsampling filter [6]. After that, texture videos with reduced resolution depth videos are encoded using 3DV-ATM coding scheme. In decoding, bilinear filter is used to upsample the decoded depth images into original resolution. Coding techniques in 3DV-ATM not only include conventional intra-view prediction and inter-view prediction for texture and depth videos, but also contain new coding techniques favorable for higher compression ratio, for example, view synthesis prediction, in-loop joint inter-view depth filter, depth range based weighted prediction for depth coding, etc. The characteristic of depth image, however, is different from that of texture image. Depth image is a piecewise smooth image that has large homogeneous regions within objects and sharp depth changes at object boundaries, as shown in Fig. 1(a). H.264/AVC based coding method in 3DV-ATM, however, does not handle well at sharp object boundaries, which are always blurred by subsampling or transform coding even with ringing artifact (depth inconsistencies around boundaries) in the decoded depth images as shown in Fig. 1(b). This results in annoying artifacts in the synthesized views, as shown in Fig. 1(d).



(a) (b) (c) (d)

Fig. 1 (a) Original depth image (b) Compressed depth image with QP=32 (c) Synthesized view from uncompressed texture and depth (d) Synthesized view from compressed texture and depth.



(a) (b) (c)

Fig. 2 (a) Decoded depth with edge blocks (b) Cropped rectangular region of the decoded depth (c) The refined block A position.

To tackle this problem, many depth map enhancement filters have been proposed. They are generally classified as pixel-based or block-based depth map filters. In pixel-based depth map filter, bilateral filter [7] and clustering of depth layer [8] are the most popular approaches. In [7], a depth adaptive joint bilateral was proposed. It takes the advantages of smoothing regions inside object boundary and preserving sharp edges. But the proper selection of filter parameters has great impact on sharp edge recovery. In the depth clustering

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

filter and depth reconstruction filter [8], sharp edge can be retained through the depth layer classification. However, pixel-based depth map filter may create inconsistent depth values around the depth edge regions and generate ringing artifacts. In addition, this depth inconsistency causes texture distortion in synthesized views. In general, the block-based depth map filter can avoid inconsistent depth values and remove ringing artifacts. In [9, 10], a depth map post-processing filter is designed to minimize the compression artifacts by analyzing the histogram inside each block. Unfortunately, layer classification fails in the situation that the boundaries of the blocks lying around the depth edge regions due to the lack of information of all the depth layers.

In addition, the conventional block-based depth map filters divide the depth image into non-overlapped blocks. Only blocks containing large depth discontinuity edges, like blocks with white boundary as shown in Fig. 2(a), are filtered. However, some of these block boundaries located closer to edge regions are similar to the block A shown in Fig. 2(b). In this case, sharp edges are difficult to be recovered due to insufficient depth layer information within the block. For example, the block A of Fig. 2(b) contains foreground information and small part of the smooth edges. It is because the smooth edge regions only take very small percentages of the block A. They will be treated as noise and refined as the foreground regions. The lack of information of background regions makes it difficult to recover the original sharp edge. Sometimes it even degrades the quality of synthesized views in both objective and subjective evaluations that cause the overall performance to degrade after applying the block filter. If these blocks can be placed in the correct position, such as the center of block A is located at the vertical edge regions as shown in Fig. 2(c), the refinement of depth values has reliable reference since this center positioning provides sufficient depth values for each of the depth layers. Smooth edge regions of block A can be easily treated as noise and refined using the correct background or foreground depth layer information to recover its original depth values.

In this paper, a block-based filter with adaptive block positioning and expansion is proposed to sharpen the compressed smooth depth edges. Details of proposed filter are presented in section 2. Experimental results are shown in section 3, and a conclusion is drawn in section 4.

2. ADAPTIVE BLOCK TRUNCATION FILTER

Depth image exhibits a high degree of spatial correlation except at object boundaries [11,12]. Object boundaries are more sensitive to coding errors compared to smooth and flat regions [12]. The proposed filter aims to enhance depth edge regions and recover the original sharp edges from blurred edges caused by depth image subsampling or compression. The proposed filter includes three steps: (1) Edge Pixel Detection, (2) Edge Block Repositioning and Expansion, and (3) Depth Refinement and Layer based Smoothing. Details of each step are described in following subsections.

2.1. EDGE PIXEL DETECTION

Sharp depth edge pixels are detected by the horizontal and vertical gradients of depth image defined as below:

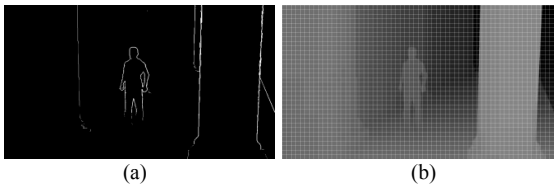
$$\nabla f_x = \frac{\partial f}{\partial x}, \tag{1}$$

$$\nabla f_y = \frac{\partial f}{\partial y}. \tag{2}$$

If horizontal or vertical gradients are larger than a threshold (D_T) (i.e. $|\nabla f_x| > D_T$ or $|\nabla f_y| > D_T$), depth pixel at position (x, y) is treated as sharp edge pixels, i.e. the detected edge pixels in white color as shown in Fig. 3(a). Since the view synthesis of ATM only has horizontal disparities, in our experiment, a 1-D kernel, (-1, 1), is used to calculate both horizontal and vertical gradients and D_T is set adaptively based on the hole's size created by the depth value difference between two adjacent pixels to the virtual view. The location of the virtual view is the nearer neighborhood of the input view. The threshold D_T is calculated as:

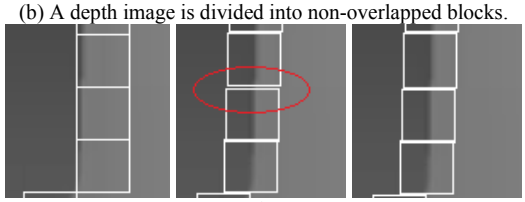
$$D_T = \frac{h}{t_c f} \frac{1}{\left(\frac{1}{255 z_n} - \frac{1}{255 z_f}\right)} \tag{3}$$

where t_c and f are the baseline distance and the focal length, respectively. z_n and z_f represent the nearest distance and the farthest distance in the scene. h is the hole size and set to 2. In 3DV-ATM, the camera parameters are encoded and transmitted to the decoder. Thus, using the adaptive threshold D_T, no extra data is needed to transfer to decoder. Also, no modification is required for the encoder.



(a) (b)

Fig. 3 (a) Example of edge pixels that detected from a decoded depth image (b) A depth image is divided into non-overlapped blocks.



(a) (b) (c)

Fig. 4 (a) Detected depth edge blocks (b) Depth edge blocks after repositioning (c) Depth edge blocks after repositioning and expansion.

2.2. ADAPTIVE BLOCK POSITIONING AND EXPANSION

After edge pixel detection, a depth image is divided into $M \times M$ non-overlapped blocks as shown in Fig. 3(b). Block size is set adaptively based on the image resolution as

$$M = 2^{round\left(\log_2 \frac{W}{a}\right)}, \tag{4}$$

where W is the width of the depth image, a is the constant (a = 125) and the *round(·)* function rounds the input value to its nearest integer value. The non-overlapped blocks that contain edge pixels are considered as edge blocks as shown in Fig. 2(a) and only these edge blocks will be processed. In order to avoid the boundaries of these edge blocks lying too close to the smooth depth edge regions as the problem

described in the introduction, the positions and sizes of these edge blocks are adjusted before depth refinement. Basically, the block repositioning aims at locating most of the depth edge pixels at the block's center. This repositioning can be efficiently realized by using the average of edge pixels' positions within the processing block as the block's center. Then, the top-left position of the repositioned edge block can be determined by

$$B = (x_B, y_B) = round\left(\frac{\sum_{P_i \in \varphi} P_i}{N} - Q\right), \quad (5)$$

where P_i is depth pixel's position (x_i, y_i), φ represents the set of depth edge pixels inside the processing block, N is the total number of depth edge pixels inside the $M \times M$ block, and Q=(M/2, M/2). In Eq. (5), the subtraction of Q is used to get the xy-coordinate of the top-left position of the block. Fig. 4 shows examples of edge block locations before and after repositioning. Center of the edge blocks can be located to vertical depth edges after this simple process as shown in Fig. 4(b). However, this repositioning may cause some depth edge pixels that lie inside the edge block before reposition to outside region of the block as indicated in the circle region in Fig. 4(b). In order to tackle this problem, size of the edge block is also needed to adaptively expand for covering all the missing depth edge pixels. This block expansion process can be realized by using the minima of horizontal and vertical coordinates between the edge pixels inside the block before repositioning and the repositioned coordinates B as the new top-left position B' of the expanded block. Thus, the top-left position of the expanded edge block can be expressed as

$$B' = (x'_B, y'_B) = \left(\min\left(x_B, \min_{P_i \in \varphi}\{x_i\}\right), \min\left(y_B, \min_{P_i \in \varphi}\{y_i\}\right)\right) \quad (6)$$

where the function min(a, b) is used to find the minimum value between a and b. The expanded block size $M' \times N'$ can be respectively determined by the maxima of horizontal and vertical distances from B' to its lower-right block coordinate (x_B+M, y_B+M) of the repositioned block and from B' to maxima edge pixel coordinates inside the block before repositioning. Thus, M' and N' can be expressed as

$$M' = \max\left(x_B + M - x'_B, \max_{P_i \in \varphi}\{x_i\} - x'_B\right) \quad (7)$$

$$N' = \max\left(y_B + M - y'_B, \max_{P_i \in \varphi}\{y_i\} - y'_B\right) \quad (8)$$

where the function max(a, b) is used to find the maximum value between a and b. Fig. 4(c) shows the edge blocks after the adaptive block expansion. By using these repositioned and expanded edge blocks for depth values refinement, the refinement accuracy could be significantly improved.

2.3. DEPTH REFINEMENT

After the edge block repositioning and expansion, a modified block truncation method is used to refine the depth values. It first classified the processing block into two layers, foreground (R_F) and background region (R_B) by comparing depth values with the mean depth value (D_m) of the block as:

$$R_F = \{(x, y) | D(x, y) \geq D_m\}, \quad (9)$$

$$R_B = \{(x, y) | D(x, y) < D_m\}, \quad (10)$$

where $D(x, y)$ is the depth value at location (x, y). After the layer classification, mean values of foreground region and background region are calculated and denoted as m_F and m_B. Depth values of detected edge pixels are refined by selecting the nearer value of m_F and m_B as follow:

$$D'(x, y) = \begin{cases} m_F, & |D(x, y) - m_F| \leq |D(x, y) - m_B| \\ m_B, & otherwise \end{cases}, \quad (11)$$

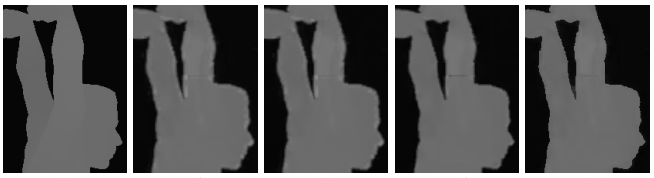
where $D'(x, y)$ is the refined depth value. With this depth refinement, the blurred depth edges can be recovered. Since block size is set to a relative small value (eg: 8×8 for image with resolution of 1024×768) and the refined block is the edge regions, it has very high chance of having only two layers and makes the block truncation very efficient. To further remove the ringing artifacts, a layer based smooth filter is used. A 3×3 average filter is independently applied to the foreground region (R_F) and background region (R_B). Sharp edges can be maintained and the noise around the edges regions due to compression can be minimized.

3. EXPERIMENTAL RESULTS

Experiment is conducted according to codec configuration (EHP profile) of 3DV-ATM 5.0 common testing conditions [13]. The proposed adaptive block truncation filter (ABTF) is implemented in the decoder and applied to the decoded depth images without dilation filter since our purpose is to recover depth images from compression. Results from ABTF are compared with pixel-based filter using adaptive joint bilateral filter (AJBF) [7] and block-based filter using adaptive sharpening filter (ASF) [8]. Three texture views plus three depth views in MVD format are encoded and decoded by the 3DV-ATM. The decoded depth images are enhanced by AJBF, ASF and ABTF and compared with the original depth image without downsampling by Peak signal-to-noise ratio (PSNR). The decoded texture videos with the enhanced depth videos are used to generate the virtual views by 3DV-HTM 4.0 rendering reference software according to common test conditions [13]. Six virtual views between three cameras (evenly distributed) are synthesized for evaluation. The virtual views generated from original texture videos with original depth videos are used as the reference views for objective performance by using PSNR.

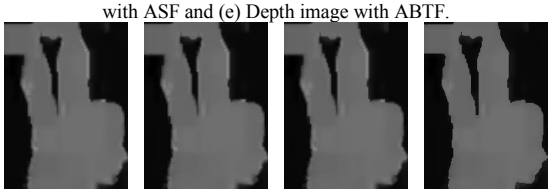
The cropped original, compressed with different QP values, and enhanced depth images of *Undo_dancer* are shown in Fig. 5 and Fig. 6. Their corresponding synthesized views are shown in Fig. 7, Fig. 8. The large distortions around the object boundaries due to subsampling and transform coding are easily observed in the decoded depth as shown in Fig. 5(b). Depth images enhanced by AJBF and ASF still have blurred edges around depth transitional regions as in Fig. 5 (b) and (c). In view synthesis, the blurred edges result in the annoying boundary artifacts as in Fig. 7. Both AJBF and ASF are based on bilateral filter and try to recover the compressed depth image. Depth images enhanced by ASF are better than AJBF, since the ASF optimizes the parameter for each of the blocks adaptively while AJBF uses a fixed

parameter setting. In addition, ringing artifacts are reduced in ASF since it is a block-based depth map filter. From Fig. 5(e), the proposed filter can recover sharp depth edges and the filtered depth image is more similar to its original depth images. By comparing the results of synthesized views from AJBF and ASF, Fig. 7(e) show the proposed filter can significantly reduce the boundary artifacts caused by the blurred depth edges. For larger QP (Quantization Parameter) value, there are more distortions in the edges regions, which increase the difficulty in recovering sharp edges. ABTF can also recover the sharp edges when the QP value is large, and the synthesized boundary artifacts can be removed since the sharp depth edges can be recovered as Fig. 6 (d).



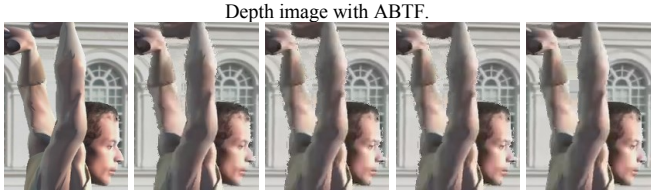
(a) (b) (c) (d) (e)

Fig. 5 (a) Original depth image of 148th frame view 5 of *Undo_dancer,* (b) Compressed depth image (QP=31) (c) Depth image with AJBF (d) Depth image with ASF and (e) Depth image with ABTF.



(a) (b) (c) (d)

Fig. 6 (a) Compressed depth image (QP=41) of 148th frame view 5 of *Undo_dancer,* (b) Depth image with AJBF (c) Depth image with ASF and (d) Depth image with ABTF.



(a) (b) (c) (d) (e)

Fig. 7 Synthesized view of 148th frame view 3 of *Undo_dancer* based on (a) Original texture and depth (b) Compressed texture and depth (QP=31) (c) Compressed texture and depth using AJBF (d) Compressed texture and depth using ASF (e) Compressed texture and depth using ABTF.

(a) (b) (c) (d)

Fig. 8 Synthesized view of 148th frame view 3 of *Undo_dancer* based on (a) Compressed texture and depth (QP=41) (b) Compressed texture and depth using AJBF (c) Compressed texture and depth using ASF (d) Compressed texture and depth using ABTF.

Table 1 (a) presents the objective evaluation improvement of decoded depth images and it shows the enhanced depth images by AJBF and ASF have significant improvement in terms of PSNR. Results of AJBF attain lower PSNR compared with those of ASF. However, the enhanced depth images by AJBF and ASF still contain blurred depth edges that result in annoying boundaries artifacts in the synthesized views. Thus, the PSNR of synthesized views through AJBF and ASF are very similar to those of ATM anchor as in Table 2. Results show that depth image filtered by ABTF and the synthesized texture views are more similar to the reference views compared with AJBF and ASF. ABTF has improvement up to 3.25 dB in the depth image and bitrate reduction of 3.06% in the synthesized views. This exists in *Undo_dancer* sequence, in which depth distances between background and foreground are large and its depth images satisfy our assumption that depth image exhibits a high degree of spatial correlation except at object boundaries. In addition, the proposed ABTF has a low complexity as tabulated in Table 2 (b). There is only around 7% increase in runtime when comparing with the decoding time of the 3DV-ATM 5.0. However, the ABTF can achieve significant improvement for the decoded depth image and synthesized view.

Sequence	AJBF	ASF	ABTF
	ΔPSNR (dB)		
Poznan_hall2	+0.20	+0.22	+0.37
Poznan_street	+0.70	+0.74	+0.76
Undo_dancer	+2.05	+2.10	+3.25
GT_FLY	+0.80	+0.83	+1.00
Newspaper	+1.33	+1.40	+1.53
Kendo	+0.03	+0.03	+0.03
Balloons	+0.70	+0.84	+0.93
Average	+0.83	+0.88	+1.12

(a)

Sequence	Time Ration (%)
Poznan_hall2	104.43
Poznan_street	103.20
Undo_dancer	110.00
GT_FLY	113.91
Newspaper	103.29
Kendo	103.59
Balloons	110.36
Average	106.97

(b)

Table 1 (a) PSNR improvement of depth map compared with ATM 5.0 anchor (b) Decoding complexity of ABTF compared with ATM 5.0 anchor.

Sequence	AJBF		ASF		ABTF	
	ΔPSNR (dB)	BD-Bitrate (%)	ΔPSNR (dB)	BD-Bitrate (%	ΔPSNR (dB)	BD-Bitrate (%)
Poznan_hall2	0.00	-0.05	+0.02	-0.51	+0.06	-1.82
Poznan_street	0.00	+0.07	+0.01	-0.08	+0.01	-0.10
Undo_dancer	+0.01	-0.19	+0.04	-1.05	+0.13	-3.06
GT_FLY	-0.01	+0.27	+0.00	-0.04	+0.05	-1.23
Newspaper	-0.01	+0.23	+0.01	-0.06	+0.08	-1.98
Kendo	-0.01	+0.01	+0.00	+0.02	+0.07	-1.74
Balloons	0.00	+0.16	+0.02	-0.33	+0.08	-1.45
Average	0.00	+0.07	+0.01	-0.29	+0.07	-1.63

Table 2 PSNR improvement of synthesized view compared with ATM 5.0 anchor.

6. CONCLUSION

As transform coding based H.264/AVC and H.265/HEVC were adopted in recent 3DV coding standards using MVD formats, sharp depth edges of the decoded depth images are often blurred or even with ringing artifacts, which results annoying boundary artifacts in the synthesized views. To recover sharp depth edges from decoded depth images, an adaptive block truncation filter with low complexity is proposed. It uses a very effective depth edge block repositioning and expansion method to achieve higher quality depth values refinement. It is because the adaptive block positioning and expansion can increase the sharp depth edges recoverability since it contains sufficient information of each depth layer inside the block. Experimental results show sharp depth edges can be recovered and the boundary artifacts in the synthesized views can also be removed. In addition, the objective evaluation also has improvement in terms of PSNR.

7. REFERENCE

[1] K. Müller, P. Merkle, T. Wiegand, "3-D Video Representation Using Depth Maps," *Proc. IEEE* , vol.99, no.4, pp.643-656, Apr. 2011

[2] M. Tanimoto, M.P. Tehrani, T. Fujii, T. Yendo, "Free-Viewpoint TV", *IEEE Signal Processing Mag.*, vol. 28, no.1, pp. 67-76, Jan. 2011.

[3] P. Kauff, N. Atzpadin, C. Fehn, M. Müller, O. Schreer, A. Smolic, and R. Tanger, B, "Depth map creation and image based rendering for advanced 3DTV services providing interoperability and scalability," *Signal Process.: Image Commun.*, Special Issue on 3DTV, vol. 22, no. 2, pp. 217–234, Feb. 2007.

[4] ISO/IEC JTC1/SC29/WG11, "Tentative test model for AVC-based 3D video coding (3DV-TM)," Doc. M22842, Switzerland, Nov. 2011.

[5] ISO/IEC JTC1/SC 29/WG 11, "Test Model under Consideration for HEVC based 3D video coding", Doc. M12350, Nov. 2011.

[6] Y. Chen; S. Liu; Y. K. Wang, M. M. Hannuksela, H. Li, M. Gabbouj, "Low-complexity asymmetric multiview video coding," *Proc. IEEE Int. Conf. Multimedia and Expo (ICME)*, pp.773,776, Apr. 2008.

[7] D.V.S.X. De Silva, W.A.C. Fernando, H. Kodikaraarachchi, S. T. Worall, A. M. Kondoz, "Improved depth map filtering for 3D-TV systems", *Proc. IEEE Int. Conf. Consumer Electronics (ICCE)*, pp.645-646, Jan. 2011.

[8] K.J. Oh, S.Y. Yea, A. Vetro, Y.S. Ho, "Depth Reconstruction Filter and Down/Up Sampling for Depth Coding in 3-D Video", *IEEE Signal Process. Lett.*, vol.16, no.9, pp.747-750, Sept. 2009.

[9] D.V.S.X. De Silva, W.A.C. Fernando, H. Kodikaraarachchi, S.T. Worall, A.M. Kondoz, "Adaptive sharpening of depth maps for 3D-TV", *Electron. Lett.*, vol.46, no.23, pp.1546-1548, Nov. 2010.

[10] D.V.S.X. De Silva, W.A.C. Fernando, H. Kodikaraarachchi, S.T. Worall, A.M. Kodoz, "A Depth Map Post-Processing Framework for 3D-TV Systems based on Compression Artifact Analysis", *IEEE J. Sel. Topics Signal Process.*, accepted for publication, 2011.

[11] 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", *Proc. IEEE Int. Conf. Image Processing (ICIP)*, pp. 733–736, Nov. 2009.

[12] K.J. Oh, A. Vetro, Y.S. Ho, "Depth Coding Using a Boundary Reconstruction Filter for 3-D Video Systems", *IEEE Tans. Circuits Syst. Video Technol.*, vol.21, no.3, pp.350-359, Mar. 2011

[13] ISO/IEC JTC1/SC29/WG11, "Common Test Conditions for 3DV experimentation", Doc. N12745, Geneva, Switzerland, May. 2012.