Depth-Based Image Processing for 3D Video Rendering Applications

Terence Zarb, Carl James Debono
Department of Communications and Computer Engineering, University of Malta, Msida, MSD 2080, Malta
t.zarb@ieee.org, c.debono@ieee.org

***Abstract* – Continuous advances in video rendering platforms and image processing techniques have made possible innovative multimedia services that are changing the way visual media is consumed. An interesting application is free-viewpoint television (FTV), which allows the viewer to interactively select the scene viewing angle. To allow smooth navigation within the scene, virtual views must be rendered between the actual cameras. One solution for generating 3D scenes is through the capture of multi-view videos and their corresponding depth maps, with the latter supplying scene geometry information, which is essential for view synthesis. In this paper, we present a depth image-based rendering (DIBR) algorithm which utilizes improved image processing techniques to synthesize virtual views with rendering quality surpassing that of current state-of-the-art algorithms. On average, a peak signal-to-noise ratio (PSNR) gain of 2.5 dB and 1.5 dB for the *Ballet* and *Breakdancers* sequences, respectively, is achieved when compared to current solutions.**

***Keywords* – 3D video; free-viewpoint television; multi-view video plus depth; view synthesis; depth image-based rendering**

I. INTRODUCTION

Recent developments in video capturing and display hardware, broadband communication channels, image processing, and multimedia coding techniques, have paved the way for emerging 3D Video (3DV) applications, such as Free-viewpoint Television (FTV). FTV is a system for viewing natural video that allows viewers to watch a scene from arbitrary positions. To provide this flexibility, multi-view capture is required, where the same scene is filmed from discrete viewpoints using multiple cameras. The captured Multi-View Video (MVV) is formatted into a 3D scene representation format, which is ultimately coded and transmitted. To provide a sense of continuity in between camera views, virtual views must be rendered in the missing space. This process is known as virtual view synthesis [1].

One of the most popular rendering techniques proposed for practical FTV architectures is Depth Image-Based Rendering (DIBR). This method requires multi-view depth maps together with the texture MVV for its 3D representation. A depth map represents the 3D scene surface and contains information relating to the distance of the surfaces of scene objects from a viewpoint. Such data can be obtained directly from a depth camera or estimated from the texture videos using stereo matching techniques. The most commonly used representation method is the Multi-view Video plus Depth (MVD) format [2]. In this format, a depth value is associated to each pixel for all input camera views. As a result, for each texture video, there is a corresponding gray-scale depth map video having the same spatio-temporal resolution [2]. The MVD format and DIBR are considered by MPEG as the reference data format and synthesis framework, respectively, for FTV architectures [2].

This paper presents an enhanced DIBR algorithm with improved image processing techniques and novel concepts to generate high-quality virtual views. These enhancements result in improved quality of the synthesized views in FTV and advanced 3DV applications compared to current methods.

The rest of the paper is organized as follows. Section 2 explains the concept of DIBR and outlines its challenges. Section 3 details the proposed solution and its results are presented in Section 4. Finally, Section 5 provides a concluding section, which summarizes the contributions in the paper.

II. DEPTH IMAGE-BASED RENDERING

The concept of DIBR involves the projection of the reference (original) views onto the selected virtual view. Projection is carried out by 3D warping using the texture and the depth information of the reference cameras [3]-[4]. Given a desired virtual position, the two nearest reference camera views are selected and warped to the target viewpoint. This results in two warped images which are then blended to generate the virtual view. The generated image is finally inpainted to fill any remaining empty regions or holes.

A. 3D Warping

During 3D warping, a pixel (x_r, y_r) in the reference image is first back-projected to the world coordinates (X_W, Y_W, Z_W), using the camera calibration parameters and depth values associated with the reference camera. This yields (1) [4], where z_r is the depth value of the projected 3D point with respect to the reference camera, K_r is a 3×3 intrinsic matrix (internal properties of the reference camera), R_r is a 3×3 rotation matrix (orientation of the reference camera), and $\boldsymbol{t}_r$ is a 3×1 translation vector (position of the reference camera).

$$[X_W, Y_W, Z_W]^T = R_r^{-1}(z_r K_r^{-1}[x_r, y_r, z_r] - \boldsymbol{t}_r) \quad (1)$$

The resulting 3D point is then re-projected to pixel (x_v, y_v) in the virtual view using the camera parameters of the virtual camera. This gives (2) [4], where K_v, R_v, $\boldsymbol{t}_v$ and z_v have the same definition as K_r, R_r, $\boldsymbol{t}_r$ and z_r respectively, but now refer to the virtual camera.

$$z_v[x_v, y_v, z_v]^T = K_v(R_v[X_W, Y_W, Z_W]^T + \boldsymbol{t}_v) \quad (2)$$

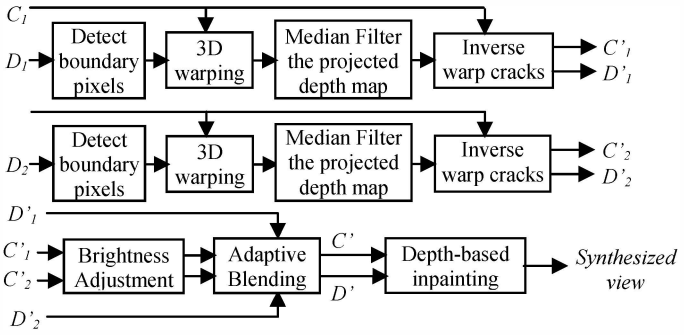


Figure 1. Proposed view synthesis algorithm.

B. *Challenges*

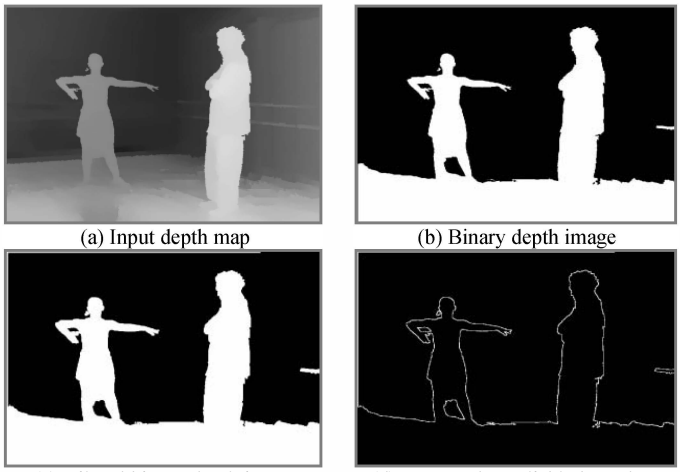
Occlusion and disocclusion regions provide a major challenge for DIBR algorithms. Occlusion regions are defined as areas which are visible in the reference view, but become invisible in the virtual view. This results in unwanted overlapping of projected pixels. On the other hand, disocclusion regions refer to areas which are inexistent in the reference image, but become visible from the target viewpoint, thereby resulting in unfilled pixels or holes [5]. Such regions are located in the background [5]-[6].

Holes also occur when the size of an object in the reference view increases as the viewpoint is shifted. Such holes are known as cracks [6], which also result from round-off errors during 3D warping and from inaccurate depth values [5], [7]. Erroneous depth values also cause background pixels to be projected onto foreground regions. Such occurrences are known as error points [5] and show up as cracks in the foreground objects. The last type of artifact encountered is the boundary noise or ghost contours [6], which occur due to inaccurate camera parameters [3] and mismatches between the texture image and corresponding depth map at the foreground objects' boundaries [4]-[6]. Ghost contours occur at the borders of the disocclusion regions.

Current state-of-the-art DIBR algorithms are presented in [3], [5]-[8], in which several techniques are proposed to improve the rendering quality by mitigating some of the aforementioned challenges. Examples include depth pre-processing [3], illumination compensation and hole classification for inpainting [5], ghost contours avoidance [6] and depth post-processing using bilateral filters [7].

III. Design of the Proposed DIBR Algorithm

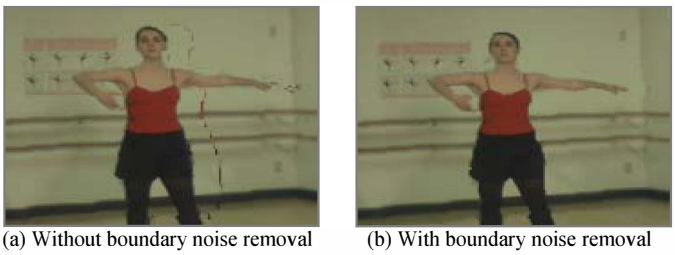
The proposed DIBR algorithm is illustrated in Fig. 1. Two test sequences were used for evaluation and results; *Ballet* and *Breakdancers*. These sequences are provided by the Interactive Visual Media group at Microsoft Research [9]. Each sequence was generated using eight synchronized cameras configured in a 1D horizontal arc. Their resolution and frame rate are 1024×768 and 15 frames per second respectively. Each sequence is 100 frames long and the corresponding depth maps and full camera parameters are also available. The depth maps were computed offline using color segmentation-based stereo matching. For verification purposes, the virtual image rendered with the proposed technique is at the camera 4 location, thus using cameras 3 and 5 as the two reference views. All visual results shown correspond to frame number 44.



(a) Input depth map (b) Binary depth image

(c) Dilated binary depth image (d) Extracted unreliable boundary

Figure 2. Unreliable depth boundary detection.



(a) Without boundary noise removal (b) With boundary noise removal

Figure 3. Boundary noise removal.

A. *Estimation of the Virtual Camera Parameters*

The first step in view synthesis is to estimate the calibration parameters of the selected virtual camera. The intrinsic matrix and translation vector, $\boldsymbol{c}$, are estimated using linear interpolation weighted by the position-dependent parameter. The relation between the two translation vectors $\boldsymbol{c}$ and $\boldsymbol{t}$ is expressed using (3). On the other hand, the rotation matrix is estimated using Spherical Linear intERPolation (SLERP) [10], which is based on the theory of quaternion curves to preserve the orthonormality property of the estimated rotation matrix.

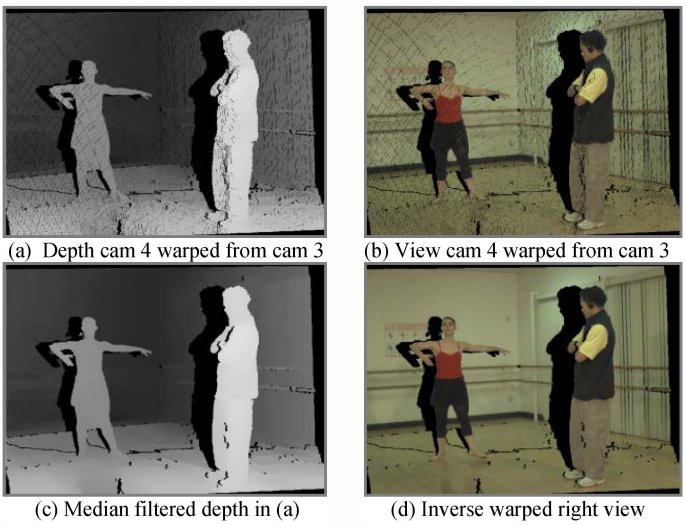
$$\boldsymbol{t} = -R\boldsymbol{c} \tag{3}$$

B. *Depth Boundary Detection*

In the proposed algorithm, pixels at high depth discontinuity regions, which result in boundary noise, are detected and not warped. The reference depth map is first converted to binary and dilated using a 5×5 square structuring element. The unreliable boundary is extracted by subtracting the binary depth image from its dilated version. This morphological operation is effective since it ensures that only pixels with background depth values are detected. This procedure is illustrated in Fig. 2 for the *Ballet* sequence. Fig. 3 (a) and (b) show part of the final view rendered before and after boundary noise removal, respectively.

C. *Depth-Based 3D Forward and Inverse Warping*

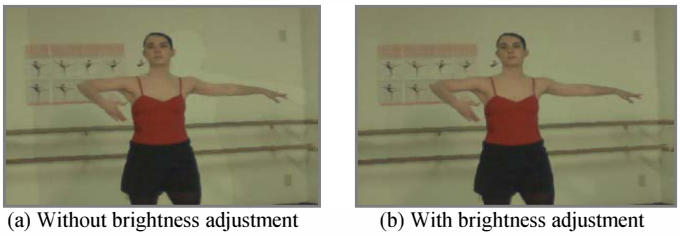
The reference texture and depth images are forward warped simultaneously using (1) and (2). The virtual texture and depth images projected from camera 3 are shown in Fig. 4 (a) and (b), respectively. Narrow holes and cracks are noticeable throughout,



(a) Depth cam 4 warped from cam 3 (b) View cam 4 warped from cam 3

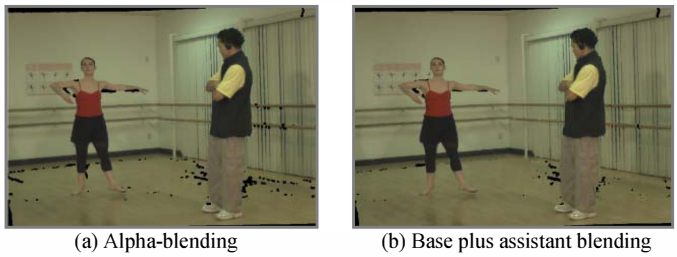
(c) Median filtered depth in (a) (d) Inverse warped right view

Figure 4. Results of forward and inverse warping for right view only.



(a) Without brightness adjustment (b) With brightness adjustment

Figure 5. Brightness adjustment.



(a) Alpha-blending (b) Base plus assistant blending

Figure 6. Blending functions.

as well as large disocclusion regions around foreground objects. Error points are also visible in the foreground objects.

The warped depth maps are median filtered using a 3×3 window and only the modified pixels are inverse warped. This reduces the amount of warping operations to almost half of those needed in traditional algorithms that use inverse warping. In this process, the warped depth pixels are projected back to the reference camera to retrieve the corresponding texture pixels. The results of filtering and inverse warping are shown in Fig. 4 (c) and (d), respectively. The narrow holes and cracks are closed and error points on foreground objects are removed since median filtering reduces the spatial inconsistencies of depth values belonging to the same object. During warping, the Z-buffer technique is used to mitigate the problem of overlapping pixels, where the most foreground pixel is selected.

D. Depth-Based Brightness Adjustment

Since the two input texture images normally have different brightness levels, color discontinuities may result during blending. The brightness adjustment algorithm adopted in this design is based on the method proposed in [5], in which the brightness of the assistant view is adjusted to match that of the base view. The base view is defined as the virtual view that is warped from the closest reference image and the assistant view is the other warped image. The novelty in our technique lies in the use of the available depth data to exclude foreground pixels in the brightness adjustment calculation. This is done to prevent the brightness level of foreground regions from biasing the result. Fig. 5 highlights the effect of brightness adjustment.

E. Adaptive Blending

In the proposed algorithm, two blending functions are used: alpha-blending and base plus assistant blending, expressed by (4) and (5), respectively.

$$I_V(u,v) = (1-\alpha) I_L(u,v) + \alpha I_R(u,v) \qquad (4)$$

where (u, v) are pixel coordinates, I_L and I_R are the left and right warped texture images, respectively, I_V is the blended view, and α is the position-dependent interpolation parameter.

$$I_V(u,v) = \alpha I_B(u,v) + (1-\alpha) I_A(u,v) \qquad (5)$$

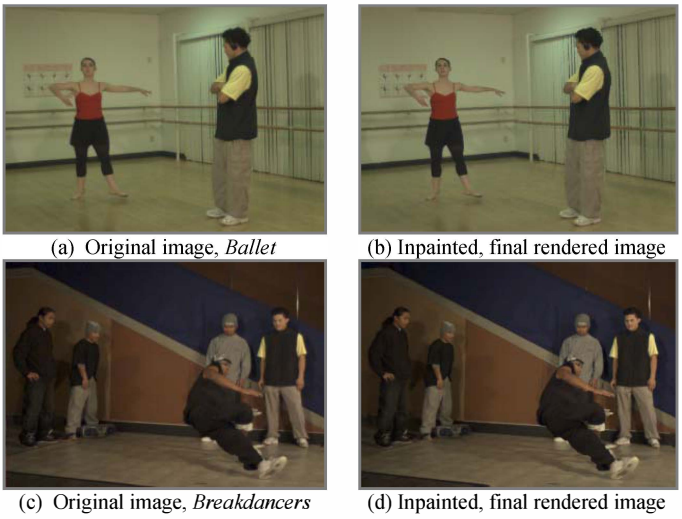
where I_B and I_A are the base and assistant views, respectively, and α is 1 or 0 for non-hole and hole pixels in I_B, respectively. In our novel approach, the selection between the two functions is done adaptively depending on the currently selected viewpoint. In alpha-blending, before applying (4), the empty pixels in each warped view are first filled by the corresponding pixels from the other projected view, and the two resulting images are adjusted so that they have the same remaining holes.

Base plus assistant blending is very effective for viewpoints that are close to a physical camera, since the disocclusions in the base view cover a relatively small area. For the other viewpoints, such blending results in annoying artifacts, such as unwanted discontinuities in foreground objects and jagged edges in textured regions. These occur because the two projected views are not perfectly aligned due to inaccurate camera parameters and warping round-off errors. Therefore, for these viewpoints, alpha-blending is more effective since the aforementioned artifacts are less pronounced due to the averaging operation. The results of both blending functions are illustrated in Fig. 6.

F. Depth-Based Inpainting

As evident in Fig. 6, the blended image still contains holes due to erroneous depth values and remaining disocclusion regions. In our proposed method, these two types of holes are treated differently. Each hole is classified as either a disocclusion region, which is always located in the background and occurs at foreground-background borders, or hole due to depth error, which can occur anywhere. Classification is done using *k*-means clustering, detailed in [5]. If the hole is classified as disocclusion, it is asymmetrically dilated towards the background direction to remove any remaining ghost pixels and is filled using only the neighboring background pixels. Otherwise, both foreground and background pixels are used.

During hole filling, the Euclidean distance e_i between the current hole pixel (x, y) and each selected neighbor i is calculated, and the empty pixel is interpolated using (6) [6].



(a) Original image, *Ballet* (b) Inpainted, final rendered image

(c) Original image, *Breakdancers* (d) Inpainted, final rendered image

Figure 7. Results of inpainting for *Ballet* and *Breakdancers* sequences.

Due to its averaging nature, this technique tends to blur highly textured regions. However, by utilizing depth information to determine the hole nature and to subsequently select the appropriate neighboring pixels, there is no blurring between foreground and background textures. Moreover, each pixel is treated independently, making the algorithm simple and fast.

$$I(x,y) = \left[\sum_{i=1}^{N}\left(e_i^{2} \times t_i\right)\right] \Big/ \sum_{i=1}^{N} e_i^{-2} \qquad (6)$$

where I is the synthesized image, N is the number of selected neighbors, and t_i is the pixel value of neighbor i.

Since unreliable boundary pixels are not warped, the rendered objects' edges look unnaturally sharp. Thus, to improve the perceptual quality, the edges in the inpainted depth map are detected using a Canny edge detection filter, and the corresponding texture pixels are low-pass filtered by an average filter. The final views for the *Ballet* and *Breakdancers* sequences are illustrated in Fig. 7 (b) and (d), respectively. It is evident that the holes in Fig. 6 were appropriately filled.

IV. Experimental Results

Fig. 7 (a) and (c) show the original texture images for the *Ballet* and *Breakdancers* sequences, respectively. These are almost identical to the corresponding rendered images; only minor artifacts at high frequency regions are visible. Thus, the proposed DIBR algorithm is capable of synthesizing high-quality virtual views. To obtain objective results, all 100 virtual frames at the camera 4 position are rendered (with alpha-blending), using cameras 3 and 5 as reference views. The synthesis quality is measured using the luminance (Y-) Peak Signal-to-Noise Ratio (PSNR) and Structural SIMilarity (SSIM) index, which are calculated using the corresponding original images as reference. The results are averaged over the 100 frames and are presented in Table 1. The results are also compared to those obtained by state-of-the-art algorithms.

As shown in Table 1, the proposed solution outperforms all current DIBR algorithms in terms of objective quality. Compared to the current solutions, our method improves the PSNR by an average of 2.5 dB and 1.5 dB for the *Ballet* and *Breakdancers* test sequences, respectively. The largest gain is obtained compared to the method in [8], since this technique filters the warped depth maps using a bilateral filter and since inpainting is not depth-based. On the other hand, the best algorithms in terms of objective quality from the reviewed methods are those of [6] and [7]. However, these are still outperformed by our solution by average PSNR gains of 0.96 dB and 0.24 dB for the *Ballet* and *Breakdancers* sequences, respectively, mainly due to more robust inpainting and the introduction of depth-based brightness adjustment.

TABLE I. Experimental Results and Comparisons

	PSNR [dB]		SSIM	
Method	***Ballet***	***Breakdancers***	***Ballet***	***Breakdancers***
[3]	32.2854	31.8150	0.8718	0.8365
[5]	32.7363	31.1573	0.9489	0.9055
[6]	34.9429	33.6818	n/a	n/a
[7]	34.8	33.9933	n/a	n/a
[8]	30.36	31.4733	n/a	n/a
Proposed	***35.8292***	***34.0787***	***0.9234***	***0.8944***

V. Conclusion

This paper presented the design of a novel view synthesis algorithm that can be adopted in 3DV rendering applications. One of the most important metrics in these applications is the rendering quality. Our method adopts improved and novel image processing techniques, which all utilize depth map information to maximize the synthesis quality. These include depth boundary detection for boundary noise removal, inverse warping, brightness adjustment, adaptive blending and inpainting. As a result, the proposed solution outperforms the current state-of-the-art solutions.

References

[1] W. Li, J. Zhou, B. Li, and M. I. Sezan, "Virtual view specification and synthesis for free viewpoint television," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 4, pp. 533-546, Apr. 2009.

[2] A. Smolic, K. Müller, P. Merkle, N. Atzpadin, C. Fehn, M. Müller, O. Schreer, R. Tanger, P. Kauff, and T. Wiegand, "Multi-view video plus depth (MVD) format for advanced 3D video systems," JVT of ISO/IEC MPEG & ITU-T VCEG JVT-W100, Apr. 2007.

[3] K. J. Oh, S. Yea, A. Vetro, and Y.S. Ho, "Virtual view synthesis method and self-evaluation metrics for free viewpoint television and 3D video," *Int. J. Imaging Syst. and Technol.*, vol. 20, no. 4, pp. 378-390, Dec. 2010.

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

[5] X. Yang, J. Liu, J. Sun, X. Li, W. Liu, and Y. Gao, "DIBR based view synthesis for free-viewpoint television," in *Proc. of 3DTV Conf.*, May 2011.

[6] S. Zinger, L. Do, and P. H. N. de With, "Free-viewpoint depth image based rendering," *J. Visual Communication and Image Representation*, vol. 21, no. 5-6, pp. 533-541, Jul. 2010.

[7] P. H. N. de With, and S. Zinger, "Free-viewpoint rendering algorithm for 3D TV," in *Proc. of the 2nd Int. Workshop of Advances in Communication*, May 2009, pp. 19-23.

[8] Y. Mori, N. Fukushima, T. Fujii, and M. Tanimoto, "View generation with 3D warping using depth information for FTV," in *Proc. of 3DTV Conf.*, May 2008, pp. 229-232.

[9] L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 600-608, Aug. 2004.

[10] K. Shoemake, "Animating rotation with quaternion curves," *ACM SIGGRAPH Comput. Graph.*, vol. 19, no. 3, pp. 245-254, Jul. 1985.