Free-viewpoint rendering algorithm for 3D TV

Peter H. N. de With, *Fellow, IEEE* and Svitlana Zinger
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, Netherlands
Email: p.h.n.de.with@tue.nl, s.zinger@tue.nl

***Abstract*— The next step after the introduction of high-definition (HD) TV in the market is likely the advent of three-dimensional (3D) television. The introduction of depth in TV scenes can be implemented in various ways, but it is sure that it will impact the complete communication chain from encoder to decoder and display. In this paper, we focus on the receiving side and assume the coded transmission of multiple views of the same scene (multi-view 3D TV). We consider the problem of rendering a new, interactively chosen view for 3D TV. We formulate the challenges of such a rendering and present our original multi-view rendering algorithm, explicitly exploiting the depth of the surrounding views. We provide a step-wise explanation of our algorithm. The algorithm assumes that the depth maps, textures of multi-view cameras and their parameters are available. The performance of our rendering algorithm compares favorably with recent results from literature, which are addressed in the paper.**

***Index Terms*— 3D TV, multi-view coding, free-viewpoint TV, rendering, interactive television, depth signals, DIBR.**

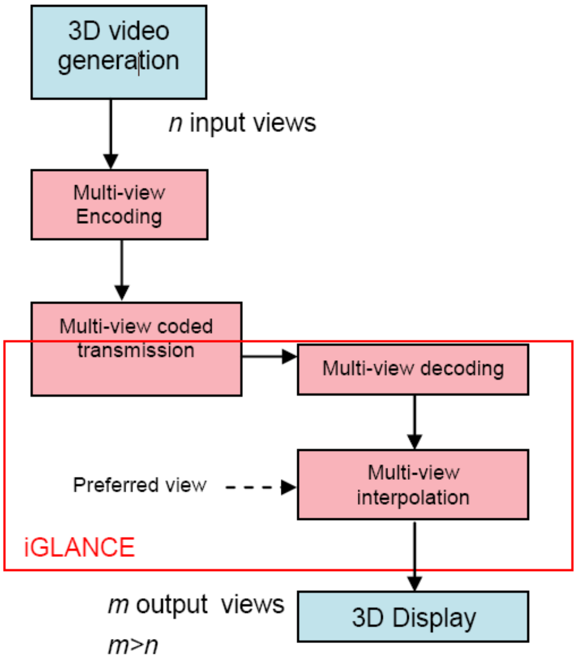


Fig. 1. Block diagram of a multi-view coding system.

I. Introduction

Three-dimensional (3D) video and imaging technology is an emerging trend in the development of digital video systems, as we presently witness the appearance of 3D displays, coding systems, and 3D camera setup. Perceiving 3D information from a screen is a long-awaited progress in cinema, television, medical applications. While we live in a 3D world, TV and computer screens provide us currently only with 2D images. Invention of autostereoscopic and stereoscopic displays gives us an opportunity to improve the visual experiences and to see the world as it is - in 3D. This means that the viewer can perceive depth while looking at such a 3D screen. Obviously, new digital image and video processing algorithms are needed to deal with such data. Preliminary system studies in this field have revealed that the whole communication chain from cameras and encoder to receiver and display, needs to be modified for 3D TV signals. Up to this point, two approaches are explored. First, a fundamental system that broadcasts a single view 3D video, which is based on e.g. a texture video plus a depth signal, or a stereoscopic texture signal. Second, in a more advanced setting, multiple cameras are used for capturing the scene and the user has the opportunity to interactively choose his own view on the scene. This is called free-view or multi-view video and requires the coded transmission of several views to enable the receiver to compute and render intermediate views. Summarizing, to create a 3D TV system, several challenges have to be met: multi-view texture and depth acquisition, multi-view coding, transmission and decoding, and multi-view rendering (see Figure 1).

Especially in 3D TV, we expect the TV of the future to be interactive, such that the user can “freely” choose the viewpoint, which in practice is likely to be within a range. The chosen viewpoint may not only be jumping between multi-view cameras, but also any viewpoint between these cameras. It will be possible to watch a soccer game or a concert from the user-preferred viewpoint, where the viewpoint can be changed at any time. This interactivity adds complexity to the 3D TV system because it requires a smart rendering algorithm that allows *free-viewpoint view interpolation*. Our research reported here concerns such multi-view rendering algorithms and aims at the problem of finding the best free-viewpoint rendering algorithm, when using multi-view texture and depth images to describe the scene. In this search, side issues are emerging, such as the quality evaluation of multi-view rendering algorithms and the influence of video compression on the rendering quality.

In the sequel, we will confine ourselves to the class of Depth Image Based Rendering (DIBR) algorithms of which references are readily accessible. Our work further extends the research of [Morvan, 2009] and [Mori et al., 2008]. This work is novel in the sense that all authors simultaneously came to the conclusion that not only reference camera views but also the novel, rendered views should be accompanied by some form of depth information. This part will be particularly addressed in Section III, where Section IV presents some experimental results.

II. Requirements to rendering

A free-viewpoint rendering algorithm should not only give a good quality but it should also be flexible and consider occlusions and be of limited complexity. Let us briefly discuss these aspects.

- *Baseline distance.* We have to deal with cameras situated on different distances from each other. Interactive 3D TV applications may require interpolating between two views that are relatively far from each other. For professional medical applications, we need a much smaller angle between the cameras, so that the algorithm has to be robust with respect to these angles.
- *Moderate complexity.* Since we aim at a real-time hardware implementation of the rendering, we want to reduce the amount and complexity of image post-processing. The rendering algorithm should be simple while providing an acceptable quality of rendering results.
- *Occlusion processing.* Occlusions are seen as “holes” in the rendered image. Post-processing is normally needed to fill in these missing areas. For the 3D TV, it is important to obtain a visually pleasant, realistic picture. For medical visualization, only a limited number of potentially erroneous (disoccluded) pixels is allowed in order to avoid mistakes in the physician’s decisions.

III. Rendering algorithm

The Depth Image Based Rendering (DIBR) algorithms are based on warping [McMillan et al., 1997] a camera view to another view. Let us specify this in some more detail.

Consider a 3D point at homogeneous coordinates $\mathbf{P}_w = (X_w, Y_w, Z_w)^T$, captured by two cameras and projected onto the reference and synthetic image plane at pixel positions $\mathbf{p}_1$ and $\mathbf{p}_2$. The 3D position of the original point $\mathbf{P}_w$ in the Euclidean domain can be written as

$$(X_w, Y_w, Z_w)^T = (\mathbf{K}_1\mathbf{R}_1)^{-1} \cdot (\lambda_1\mathbf{p}_1 + \mathbf{K}_1\mathbf{R}_1\mathbf{C}_1), \quad (1)$$

where matrix $\mathbf{R}_i$ describes the orientation and location of the camera i, $\mathbf{K}_i$ represents the 3×3 intrinsic parameter matrix of camera i, and $\mathbf{C}_i$ gives the coordinates of the camera center. Assuming that Camera 1 is located at the world coordinate system and looking in the Z direction, i.e. the direction from the origin to $\mathbf{P}_w$, we can write the warping equation into

$$\lambda_2\mathbf{p}_2 = \mathbf{K}_2\mathbf{R}_2\mathbf{K}_1^{-1}Z_w\mathbf{p}_1 - \mathbf{K}_2\mathbf{R}_2\mathbf{C}_2. \quad (2)$$

Equation 2 constitutes the 3D image warping equation that enables the synthesis of the virtual view from a reference texture view and a corresponding depth image. This equation specifies the computation for one pixel only so that it has to be performed for the entire image. Besides this complexity, we have also to compute the depth of the synthetic view.

Let us now expand this concept in a more general setting. In multi-view video, the information for warping is taken from the two surrounding camera views I_L and I_R to render a new synthetic view I_{new}. Typically, two warped images are blended to create a synthetic view at the new position:

$$I_{new} = Warp(I_L) \oplus Warp(I_R), \quad (3)$$

where $Warp$ is a warping operation and the operation $\oplus$ denotes blending of the warped views. Such an approach requires several post-filtering algorithms, in order to improve the visual quality of the results and it is especially important to close the empty areas on the resulting image caused by occlusions. Initial images I_L and I_R may be divided into layers prior to performing the warping as described in [Zitnick, 2004].

The latest research has shown that a better rendering quality is possible when we first create a depth map for a new image [Mori et al., 2008] [Morvan, 2009]. Using this depth map, we perform an “*inverse mapping*” in order to obtain texture values for I_{new} - the new image to be rendered. In this case, we have two main stages of the rendering algorithm:

1) create a depth map $Depth_{new}$ for I_{new};
2) create a texture of the new image I_{new}.

The above two stages form the backbone of the rendering algorithm and this is still similar to the approach of [Mori et al., 2008]. However, we will now present our rendering algorithm which involves clearly different techniques for rendering the new depth and image texture.

A. Depth Map Creation

The depth map creation consists of the following steps.

1. **Combine warped depth maps.** Combine the depth maps warped from the closest left and right cameras:

$$Depth_{comb} = C(Warp(Depth_L), Warp(Depth_R)), \tag{4}$$

where C defines the operation of combining two depth maps of the closest cameras by taking the depth values that are closer to the cameras. For example, $C(Warp(Depth_L(x,y)), Warp(Depth_R(x,y))) = Warp(Depth_L(x,y)$ if $Warp(Depth_L(x,y)$ is closer to the camera. In practice, combining warped depth maps means taking a maximum or minimum value of each couple of corresponding pixels.

2. **Median filtering.** The filtering function called $Median(Depth_{comb})$ involves applying median filtering to the depth map obtained at the previous step. We take a 3×3 window for the median filter, which allows us to find pixel values that occurred due to rounding of pixel coordinates during the warping.

3. **Occlusion processing.** The resulting image still contains empty regions - disoccluded areas. The following operation finds values for filling in these regions by taking the closest found background value of the depth map obtained at the previous step. We perform search in 8 directions from each empty pixel and take the value closest to the depth of the background:

$$Depth_{new} = Occ\_filling(Median(Depth_{comb})). \tag{5}$$

B. Texture Creation

Besides the depth image, we need to compute the texture of the new view, which is the final objective. The texture I_{new} is created by the following operations.

1. **Warping textures for the new view.** The new texture image is based on pixels from the left and right texture images. We select the pixels from the left and right image according to the "inverse warping" of the new depth image. This results in

$$Texture_L = Warp_L^{-1}(Depth_{new}) \tag{6}$$

and

$$Texture_R = Warp_R^{-1}(Depth_{new}), \tag{7}$$

where $Warp^{-1}$ is "inverse warping" - from the location of the new image to a position where the existing left (or right) view is located. When warping, we use the coordinates of the depth map to get the corresponding coordinates of the texture at the left (right) view.

2. **Blending.** Blend the textures obtained at the previous step:

$$\begin{aligned} Texture_{blended} = {} & Dilate(Texture_L) \\ & \oplus Dilate(Texture_R), \end{aligned} \tag{8}$$

where $Dilate$ is depth map-based filtering which aims at preventing ghosting artifacts. These artifacts result from warping the contours of objects that are often represented in the texture as a mixture of foreground and background pixels.

3. **Occlusion filling.** Finally, the rendered image is created from the blended texture and by filling the occlusion holes:

$$I_{new} = Occ\_filling(Texture_{blended}). \tag{9}$$

In our optimized rendering approach, both texture and depth are warped simultaneously but kept separated [Do et al., 2009]. This leads to less warping operations, thereby increasing the execution speed of the algorithm.

IV. Experimental results

Figure 2 portrays an example of a rendered image based on the well-known "Ballet" sequence. The experiment shows that the rendered quality is comparable to the reference views. The poster at the background clearly indicates the intermediate position of the view.

Furthermore, we have validated the quality of our rendering algorithm by changing the distance between the cameras and evaluated the quality of the rendered image. This quality has been measured and expressed in PSNR, where the rendering was performed at the position of an existing camera view. In this way, this camera view was available as a reference picture. Evidently, the larger the distance between the cameras, the more the obtained results deteriorate in quality. Figure 3 shows that we obtained better results than a recent DIBR algorithm [Mori et al., 2008]. The improvement in quality is in the range of 2–4 dB, but our comparison here has limitations. We estimate that this result is comparable to the recent results obtained in standardization groups, of which we do not have published references available. Another quality assessment is concerned with introducing compression for the camera views used for rendering. The compression has a clear influence on the obtained rendering quality. For this reason, we compare various compression techniques, in particular, JPEG and

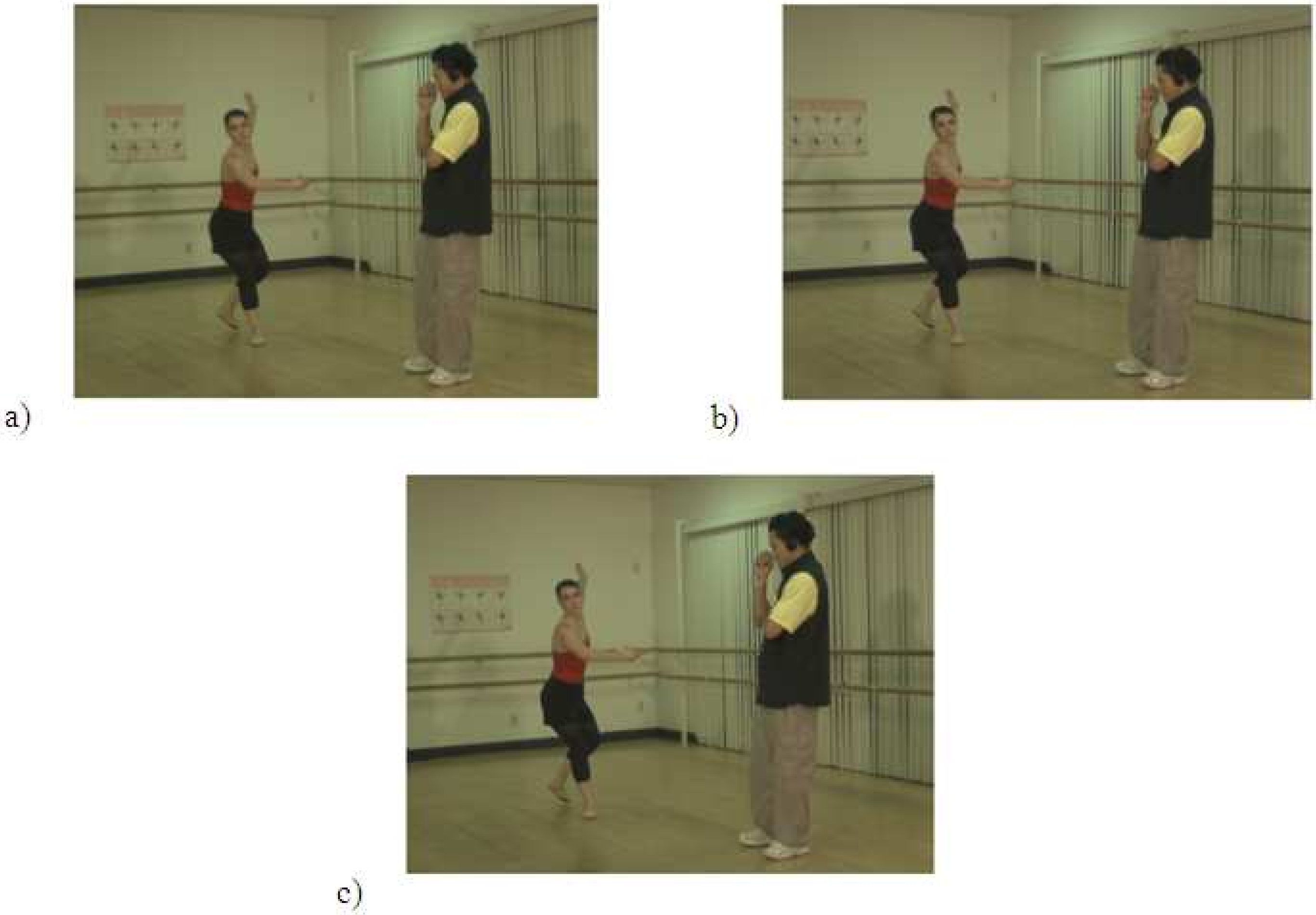
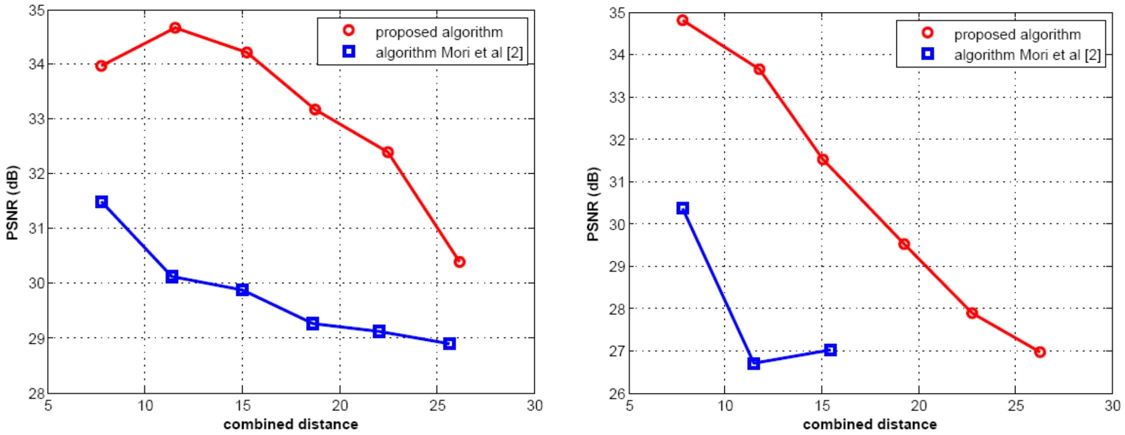


Fig. 2. Example of rendering using the closest left and right cameras: a) image from the left camera, b) image from the right camera; c) interpolated image in between.

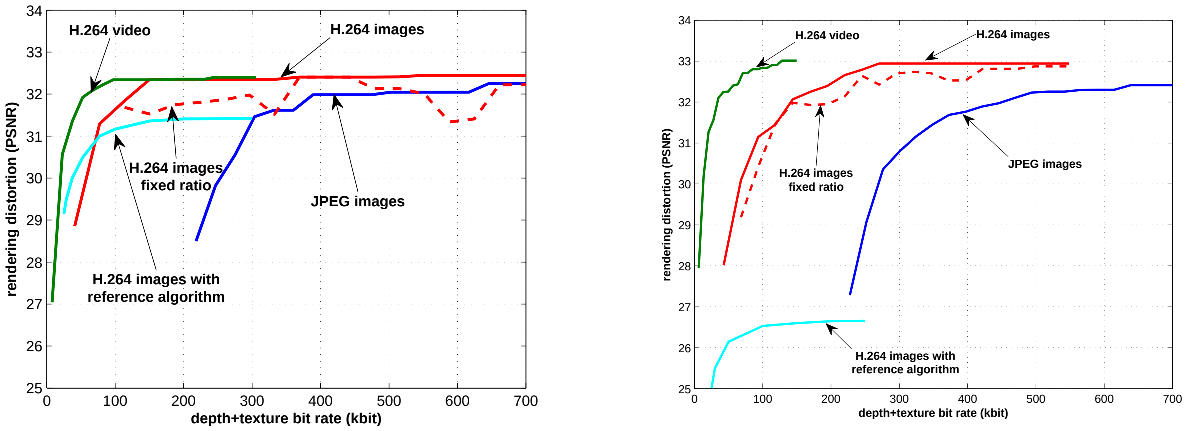


(a) Sequence “Breakdancers”

(b) Sequence “Ballet”

Fig. 3. Rendering quality for the proposed algorithm and a reference algorithm as a function of the combined distance expressed in meters.

H.264 compression, both in video and picture mode. This evaluation was also performed in [Morvan et al., 2007], and also here, our algorithm performs well. Figure 4 compares the relative performance of the compression within our multi-view rendering framework. When compared to Figure 3, the use of compression gives a loss of about 1.5 dB in performance. The relative comparison shows that H.264 video has the highest performance



(a) Sequence ‘Breakdancers’

(b) Sequence ‘Ballet’

Fig. 4. Rendering quality of various compression systems (JPEG and H.264) with rate-distortion optimized settings for the rate distribution of depth and texture, compared to fixed ratio of depth and texture.

followed by H.264 images. Clearly, JPEG only performs sufficiently well if the bit rate is high enough.

V. Conclusions and future work

In this paper, we have presented a multi-view framework for 3D TV and studied the rendering of intermediate views in relation to compression. We have described a three-step rendering that builds further on a three-step depth map creation of the intermediate view. This depth map is based on combining the warped left and right depth maps, which are subsequently processed with filtering and occlusion processing. The rendering algorithm exploits similar steps. Our experimental results have shown that the rendering algorithm gives a considerable improvement of 2–4 dB when compared to recently published results.

Free-viewpoint rendering is an area of active research, as it is also subject of standardization efforts for 3D TV transmission. Unfortunately, those results are not yet published. From our own research, we have derived a few issues that seem promising and interesting for further exploration.

- Research the main bottlenecks of the rendering presented above. It will be interesting to more accurately analyze the sources of the rendering errors, so that algorithmic improvement can be defined.
- Investigate rendering algorithms with respect to the camera settings. For example, a simpler algorithm can be used for the case that the cameras are close to each other, while for distant cameras, a more sophisticated rendering may be needed.
- Research the possibility of adapting the compression rate to the predicted rendering quality based on the scene complexity and camera positions.

References

[Do et al., 2009] L. Do, S. Zinger, Y. Morvan, P. H. N. de With, “Quality improving techniques in DIBR for free-viewpoint video”, *3DTV-Conference*, Potsdam, Germany, 2009 (to be published).

[McMillan et al., 1997] L. McMillan and R. S. Pizer, “An image-based approach to three-dimensional computer graphics”, Technical Report TR97-013, University of North Carolina at Chapel Hill, 1997.

[Mori et al., 2008] Y. Mori, N. Fukushima, T. Fujii, and M. Tanimoto, “View generation with 3d warping using depth information for FTV,” in *Proceedings of 3DTV-Conference*, pp. 229-232, 2008.

[Morvan et al., 2007] Y. Morvan, D. Farin, and P. H. N. de With, “Joint depth/texture bit-allocation for multi-view video compression,” in Picture Coding Symposium (PCS), 2007.

[Morvan, 2009] Y. Morvan, “Acquisition, compression and rendering of depth and texture for multi-view video”, PhD thesis, Eindhoven University of Technology, 2009.

[Zitnick, 2004] C. L. Zitnick, S. Bing Kang, M. Uyttendaele, S. Winder, and R. Szeliski. “High-quality video view interpolation using a layered representation”. In *SIGGRAPH ’04: ACM SIGGRAPH 2004 Papers*, ACM, pages 600-608, 2004. doi: http://doi.acm.org/10.1145/1186562.1015766.