Free-viewpoint depth image based rendering

S. Zinger*[,a], L. Do[a], P. H. N. de With[a,b]

[a] *Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, the Netherlands*
[b] *Cyclomedia Technology B.V., P.O. Box 68, 4180 BB Waardenburg, the Netherlands*

Abstract

In 3D TV research, one approach is to employ multiple cameras for creating a 3D multi-view signal with the aim to make interactive free viewpoint selection possible in 3D TV media. This paper explores a new rendering algorithm that enables to compute a free-viewpoint between two reference views from existing cameras. A unique property is that we perform forward warping for both texture and depth simultaneously. Advantages of our rendering are manyfold. First, resampling artifacts are filled in by inverse warping. Second, disocclusions are processed while omitting warping of edges at high discontinuities. Third, our disocclusion inpainting approach explicitly uses depth information. We obtain an average PSNR gain of 3 dB and 4.5 dB for the 'Breakdancers' and 'Ballet' sequences, respectively, compared recently published results. Moreover, experiments are performed using compressed video from surrounding cameras. The overall system quality is dominated by rendering quality and not by coding.

Key words:

*Corresponding author
Email addresses: s.zinger@tue.nl (S. Zinger), luat.do@gmail.com (L. Do), p.h.n.de.with@tue.nl (P. H. N. de With)

free-viewpoint, 3D video, depth image based rendering

1. Introduction

After high-definition television (HDTV), stereoscopic broadcasted and displayed content is expected to be the next innovation in the television market. This means that the viewer can perceive depth while looking at a stereoscopic screen. Many movies are already recorded in a stereoscopic format today, and commercially available stereoscopic displays are emerging. We aim at taking this innovation one step further, by introducing interactivity to the stereoscopic content, and allowing the viewer to freely select the viewing position in a continuous fashion. Obviously, new digital image and video processing algorithms are needed that support the representation of a 3D signal and enable the processing of it. Preliminary system studies in this area have revealed that the whole communication chain from cameras and encoder to receiver and display, needs to be modified for 3D TV signals. To create an interactive free viewpoint 3D TV system, several challenges have to be solved: multi-view texture and depth acquisition, multi-view coding, transmission and decoding, and multi-view rendering (see Figure 1).

The chosen free-viewpoint may not only be selected from the available multi-view camera views, but also any viewpoint between these cameras. It will be possible to watch a soccer game or a concert from the userpreferred 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

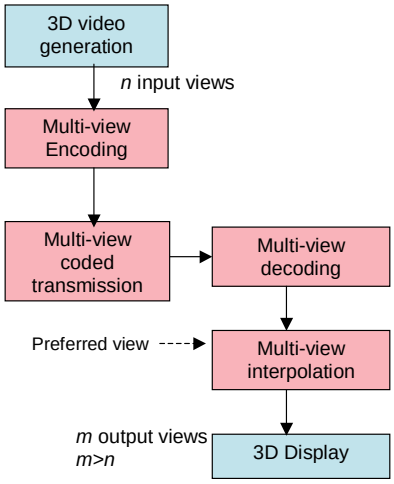

Figure 1: Block diagram of a free-viewpoint 3DTV system.

problem of finding the best free-viewpoint rendering algorithm, when using multi-view texture and depth images to describe the scene. 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. In this search for the most suited algorithm, side issues are emerging, such as the quality evaluation of multi-view rendering algorithms and the influence of video compression on the rendering quality.

The basic idea of most depth image based rendering (DIBR) rendering methods is to perform 3D warping to the virtual viewpoint using texture and depth information of the reference cameras. Artifacts are removed by post-processing the projected images. These images are then blended together and the remaining disocclusions are filled in by inpainting techniques. The basic diagram of most DIBR methods for free-viewpoint is depicted in Fig. 2. In the first step, the left and right reference camera views are warped to the

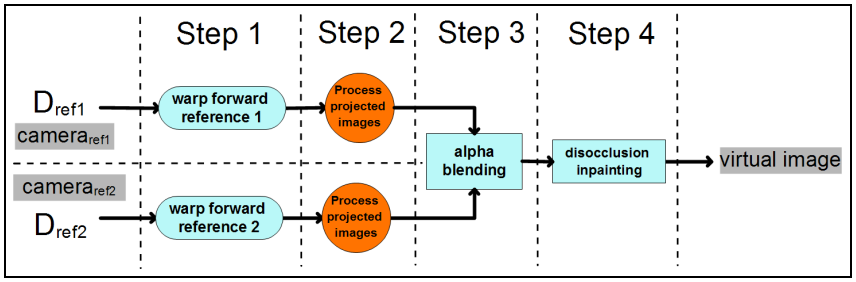

Figure 2: Basic diagram of free-viewpoint DIBR methods

virtual viewpoint using the reference depth maps. In Step 2, the projected images are processed to cope with artifacts. The most perceptually relevant artifacts are cracks caused by sampling, and ghost contours resulting from ill-defined borders in depth maps. The projected texture maps are blended in Step 3. The blending process determines, for every pixel, whether the virtual image should take the left and/or right projected textures. The last step processes and closes the disocclusions that still exist as a result of the geometry of the scene.

The paper is organized as follows. In Section 2 we describe the state-of-the-art on free-viewpoint DIBR. Section 3 describes the challenges of DIBR and presents some approaches to deal with them. The following Section 4 presents our free-viewpoint algorithm and evaluates it experimentally. We conclude with Section 5 where we discuss the work that is done and our plans for future research.

2. Current research on free-viewpoint rendering

Zitnick *et al.* [14] provide a free-viewpoint rendering algorithm which is based on a layered depth map presentation. These results are extended in

the work of Smolic *et al.* [10], where three layers of depth are identified and ranked according to their reliability. The warping results are then obtained for each layer and merged. Three post-processing algorithms are then introduced to deal with rendering artifacts. The quality of this algorithm is not measured, but it is obvious that the algorithm requires a considerable amount of pre-processing and post-processing operations.

Morvan [3] performs DIBR by first creating a depth map for the view to be rendered. The undefined pixels in the depth map are filled in by an empirically defined combination of erosion an dilation operations. Based on the created depth map, an inverse mapping is performed to obtain a texture. Warped textures are combined using the corresponding depth values, such that foreground pixels are given priority. The author performs a number of experiments and concludes that this inverse mapping based rendering outperforms several other algorithms.

The diagram for a recent algorithm, developed by Mori *et al.* [5] is depicted in Fig 3. The principal steps of the algorithm are now explained. First, the depth maps of the reference cameras are warped to the virtual viewpoint. Then the empty pixels are filled with a median filter. Afterwards, the depth maps are processed with a bilateral filter, to smoothen the depth maps while preserving the edges. Then, the textures are retrieved by performing an inverse warping from the projected depth maps back to the reference cameras. Ghost contours are removed by dilating the disocclusions. After that, the texture images are blended and the remaining disocclusions are inpainted using the method proposed by Telea [11]. The use of a median filter, such as in Mori *et al.* [5] for filling empty pixels does not solve

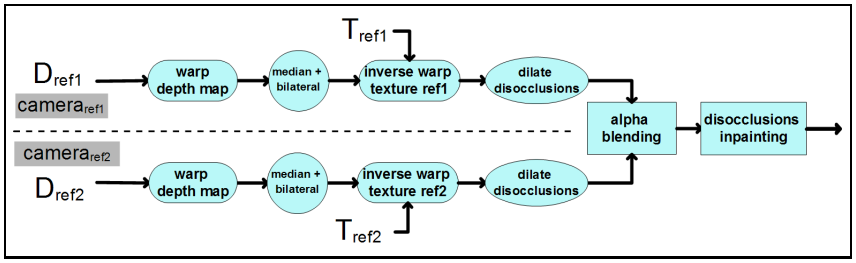

Figure 3: Rendering algorithm by Mori *et al.*

the problem of filling in empty that are organized in a line structure (further called cracks). We have found that not all cracks will be filled in this way. We have observed that cracks will appear at a virtual viewpoint when we warp a surface from one viewpoint to another viewpoint. If the surface is at the background, then cracks consist of empty pixels. But if the surface is part of the foreground, the cracks will probably contain already warped background pixels. When a bilateral filter is applied afterwards, the errors, because of cracks in the foreground, still persist. Subsequently, the textures retrieved by inverse warping are not correct. Another disadvantage of bilateral filtering is that some pixels in the disoccluded areas will be assigned a non-zero value. The inpainting technique, that the Mori algorithm employs, uses only texture information and no depth information. This is a clear shortcoming in the performance of the Mori algorithm which has led us to the solution to make use of the surrounding depth information to more reliable fill in the disoccluded regions.

In our algorithm described in Section 4, we perform forward warping for both depth maps and textures simultaneously. This considerably decreases the number of warping operations which is important since we aim at a

real-time hardware implementation. Filling disocclusions in our algorithm is performed without the dilation of disoccluded regions, thus avoiding the loss of correct texture information. Our inpainting procedure relies on depth information, making the resulting textures less blurred.

3. Imperfections of DIBR and solutions

In this section, imperfections from the DIBR algorithms are explained in detail and we present several solutions for each of them. Different combinations of these solutions lead to various rendering algorithms. Depending on the quality and the implementation issues, such as hardware, software running on a GPU, a specific rendering algorithm may be adopted.

3.1. Cracks and holes due to sampling rate

The first problem we have to overcome is cracks due to sampling in the x and y-direction of the reference image. Fig. 4 explains an example of the variable object size depending on the viewpoint. In Fig. 4, the angle of surface S with the reference viewpoint is smaller than that with the virtual viewpoint. To reconstruct surface S at the virtual viewpoint, we have to sample the visible part of S in the reference image and warp it to the virtual image. When the sampling rate is the same as the image resolution, we see that the number of samples (area of S') at the reference image for surface S is 25 pixels, while the area of S' at the virtual image is 40 pixels. So, for the virtual image, we have to reconstruct an area of 40 pixels with only 25 data samples. If every data sample contains one pixels, there will be empty areas, or cracks, at the virtual image. This can be seen in Fig. 5. The

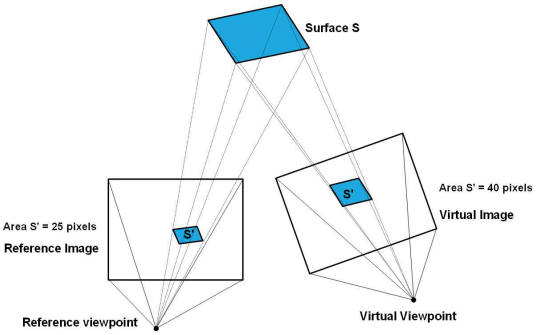

Figure 4: Object sizes change at different viewpoints

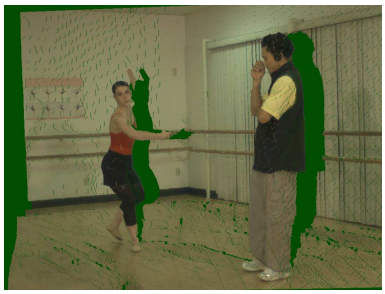

Figure 5: Cracks after warping

large empty areas, appearing as shadows at the right side of each person, are disocclusions. Let us now consider the possible solutions for this problem.

3.1.1. Oversampling of image space

The simplest solution to cope with cracks is oversampling the image space. When we sample the reference image, in de the x and y direction, with a sampling rate that is twice the resolution, most cracks will disappear. However, the disadvantage is that we have to perform warping four times, which is a costly operation. In our earlier experiments, we have concluded that it is too expensive to perform oversampling only to fill in the cracks, which consist of usually less than 3% of the image. Oversampling is depicted in Fig. 6.

3.1.2. Enlarging the warped beam

Another way to fill in the cracks is to perceive the warping of a reference pixel to the virtual image as a light beam traveling from one viewpoint to another viewpoint. When we enlarge the warp beam that hits the virtual

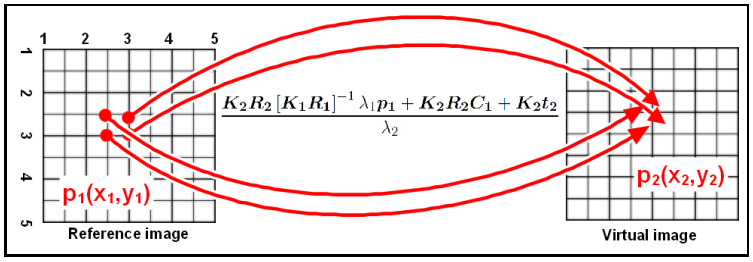

Figure 6: Oversampling the image space. The warping equation is explained in [2] and in [3]

image, the cracks will be automatically filled in (Fig. 7). The disadvantage of this method is that we have to copy every warped pixel several times, for example 8, to the virtual image. Another downside is that some neighbouring warped pixels will have the wrong value because the beam is too big.

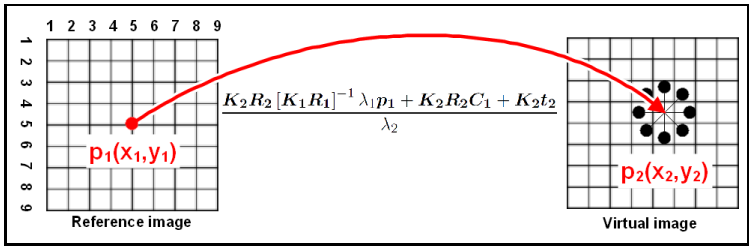

Figure 7: Cracks are handled by enlarging the warp beam

3.1.3. Inversing warping cracks

A third solution is to label the cracks and perform an inverse warping of the labeled pixels to retrieve the textures. First, we process the projected depth map with a median filter. Depth maps consist of smooth regions with sharp edges, so filtering with a median will not degrade the quality. Afterwards, we compare the input and output of the median filter and perform

an inverse warping when pixels have changed. This method is illustrated in Fig. 8.

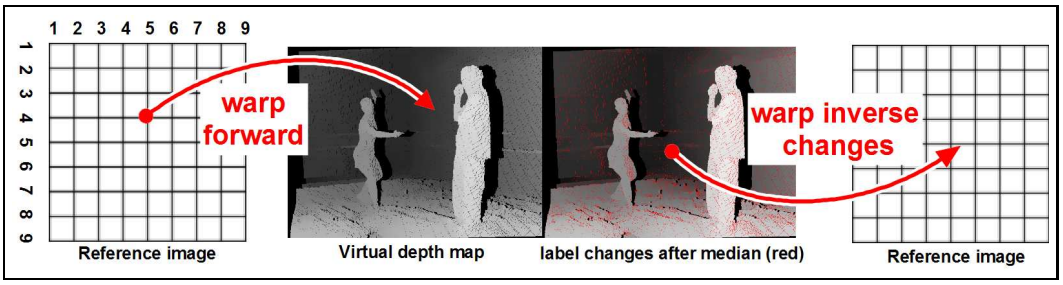

Figure 8: Median filter depth maps and inverse the cracks

3.2. Ghost contours due to high discontinuities in depth maps

Inaccuracy in depth maps causes textures to be warped to wrong places in the virtual image. This is most visible at edges of objects, where the discontinuities in depth maps are high. In depth maps, these edges are sharp and only one pixel wide, while in texture maps, they cover usually two to three pixels. This results in foreground textures at the edges, which are warped to the background. When we do not remove these pixels, the resulting rendered image will have a ghost contour of the foreground objects. This is shown in Fig. 9.

3.2.1. Removing contours by dilation

We notice that ghost contours at the projected images occur at borders of disocclusions. One solution to remove the contours is to dilate the disocclusions. An important drawback of this method is that not only ghost contours will be removed, but also correct texture pixels. The removal of ghost contours with dilation is illustrated in Fig. 10.

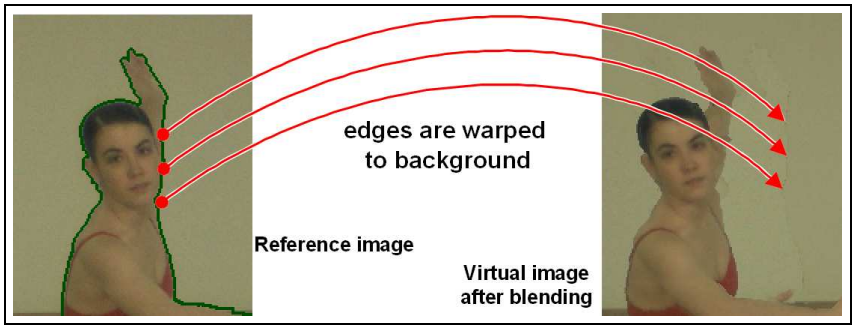

Figure 9: Ghost contour within the virtual image due to ill-defined borders

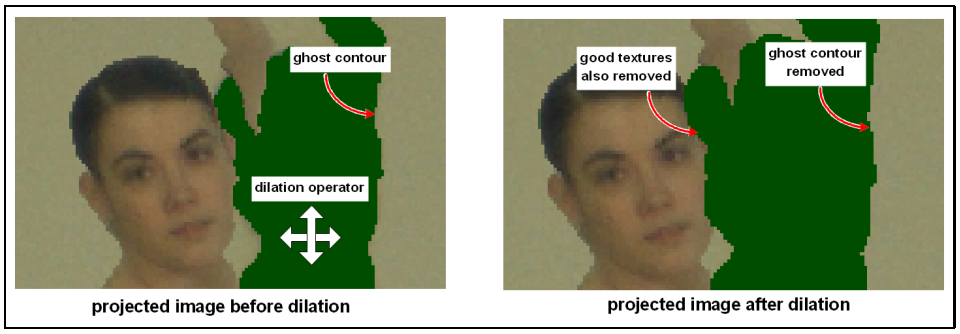

Figure 10: Ghost contours removed by dilation

3.2.2. Removing contours by warping

An alternative solution is to remove only the warped pixels at the edges of high discontinuities, but only restricted to edges at the background side. After a projection of the reference image to the virtual image, we find the edges at high discontinuities, warp them again and erase them from the virtual image. Because we know that the edges cover two to three pixels, the neighboring pixels are also erased. The removal of ghost contours by warping is illustrated in Fig. 11. The advantage of this method is that only ghost contours are erased. However, we need an additional pass to find the

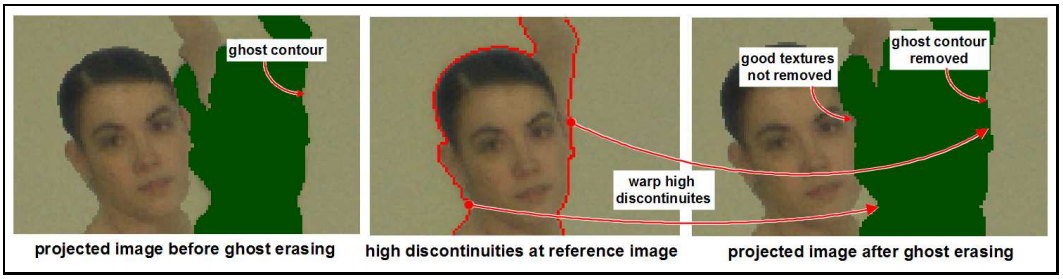

Figure 11: Ghost contours removed by warping

high discontinuities. We now briefly explain how this is accomplished. The objective is to find background pixels that have a neighboring foreground pixel. In general, we have to compare every pixel with its eight surrounding neighbors to verify if there is a high discontinuity. However, it is sufficient to label a pixel as a ghost contour, if only one of its neighbors is a foreground pixel. So for every pixel, we check the following condition.

$$\forall_{x,y} \in S, \quad \sum_{i=-1}^{1} \sum_{j=-1}^{1} D(x+i, y+j) - 9 * D(x,y) > T_d, \tag{1}$$

where S is the image space, D denotes the depth map of the reference camera and T_d is a predefined threshold.

If the above condition is true, then pixel $D(x, y)$ is labeled as a ghost contour. We warp the labeled pixels and erase them from the background.

3.2.3. Omitting the warping of edges at high discontinuities

If we zoom in at high discontinuity edges, as seen in Fig 12, we observe that the textures of those edges are a mixture of both the foreground and background. Therefore, it is better to fully omit the warping of those edges. So, our third method is to warp only pixels that are not labeled as ghost

contour pixels. For every pixel, we verify with Equation (1), whether the pixel must be warped or not. The advantage is that we do not have to erase the ghost contour in a separate pass. However, because the edges cover usually two pixels, we have to expand the labeled edges by one pixel.

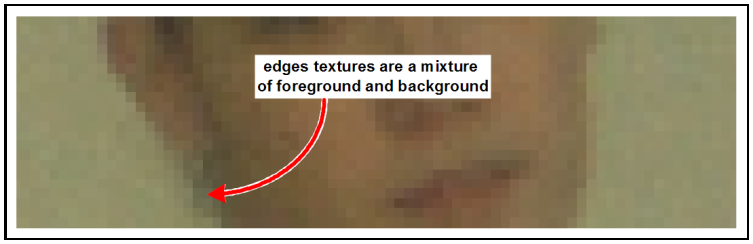

Figure 12: Edges at high discontinuities

3.3. Disocclusion inpainting

After blending the two projected images, disocclusions may still occur. These are areas that cannot be viewed from both reference cameras. Most inpainting techniques are developed to reconstruct small regions of an image. These regions can be inpainted by textures from the edges of the disocclusions, while maintaining the structure of the textures. Although these techniques are good, the resulting inpainted regions contain a certain amount of blurring. In our case, the disocclusions are not just random regions of an image. They are newly uncovered areas of background without texture information, and certainly not part of foreground objects. When we assume that the disocclusions should be background, we may use the depth information at the edges of the disoccluded region for inpainting with more accurate textures. This method is illustrated in Fig. 13 and consists of several steps. First, we search for every pixel in the disoccluded region in eight directions

for the nearest edge pixel. Then, we only take into account the edge pixels with the lowest depth value. With these pixels, we calculate a weighted average according to Equation (2). The advantage of our method is that there is no blurring between foreground and background textures. The drawback is that the inpainted region becomes a low frequency patch, when the disoccluded region is very large. The last picture of Fig. 13 shows that blurring of foreground and background textures occurs when we do not consider depth information for inpainting. Our inpainting algorithm is based on a weighted interpolation from neighboring pixels that do contain texture values, which is specified by

$$\forall_{u,v} \in O, \quad P(u,v) = \frac{\sum_{i=1}^{N} d_i^{-2} * t_i}{\sum_{i=1}^{N} d_i^{-2}}, \tag{2}$$

where O is the disoccluded region, N represents the number of edge pixels at the background, d is the distance of P to the edge of the disoccluded region, t is the texture value of an edge pixel.

This method is similar to recently published work of Oh *et al.* [7], but they had to define the disoccluded regions and their boundaries first. The disoccluded regions are inpainted using the method of Telea [11] which is expensive, since it requires analyzing the whole boundary region around disoccluded pixels. We process each pixel in the disoccluded region separately in order to achieve a high quality of the resulting image, while keeping our algorithm simple and fast.

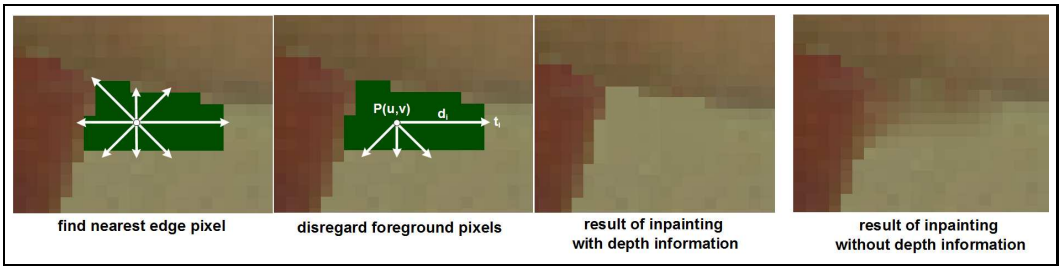

Figure 13: Inpainting with and without depth information

4. Proposed algorithm and its performance

Instead of creating a depth map for a free-viewpoint, we copy both the textures and depth maps to the virtual viewpoint and use our techniques from the previous section to fill in the cracks. This cuts the amount of warping operations to almost half. Furthermore, we do not dilate the disocclusions, because correct texture pixels would then be removed also. Thus our proposed algorithm does not create a virtual depth map first and does not use the dilation operator. The proposed algorithm is depicted in Fig. 14. At

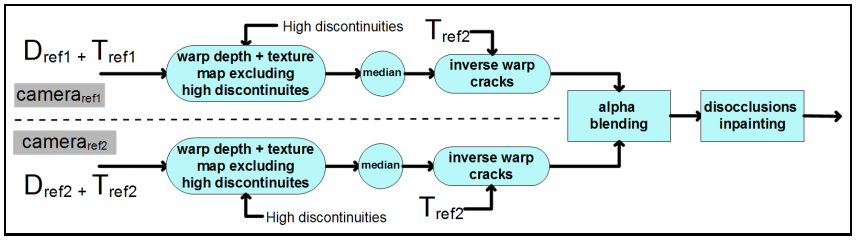

Figure 14: Proposed DIBR algorithm

first, we perform a 3D warping from the reference cameras, but copy both the depths and the texture maps to the virtual viewpoint. The cracks in the projected depth maps are then closed by a median filter, with a win-

dow of 3×3 pixels. The changed pixels are warped back to the reference cameras to retrieve the textures as in Section 3.1.3. In order to remove the ghost contours, we first verify the selection of pixels for warping. Edges at high discontinuities are not warped at all as discussed in Section 3.2.3. After blending, the remaining disocclusions are inpainted by the method described in Section 3.3.

Let us now provide a step-by-step description of our algorithm.

Step 1. Warping depth maps and copying texture values to the corresponding locations. We warp depth maps and create textures for the new viewpoint by copying the texture values to the pixel locations defined by depth map warping. The warping is specified by

$$[D_{warped1}, T_{warped1}] = Warp(HD(D_{ref1})), \tag{3}$$

$$[D_{warped2}, T_{warped2}] = Warp(HD(D_{ref2})), \tag{4}$$

where D_{ref1} and D_{ref2} are depth maps of the first and second reference cameras, respectively, function HD labels the pixels at high discontinuities as shown in Equation (1), $Warp$ is a warping operation, $D_{warped1}$ and $D_{warped2}$ are depth maps, warped from D_{ref1} and D_{ref2}, respectively. $T_{warped1}$ and $T_{warped2}$ are textures at the new viewpoint. In Equation (1), we take threshold $T_d = 80$ for the experiments presented below. In general we recommend taking the threshold equal to about 25% of the maximal depth value.

Step 2. Median filtering and defining changed pixels. We apply median filtering to the $D_{warped1}$ and $D_{warped2}$ and find the indexes $Index_{to\_warp1}$ and $Index_{to\_warp2}$ of pixels whose values have changed. This index computa-

tion is specified by

$$Index_{to\_warp1} = Cracks(Median(D_{warped1})), \tag{5}$$

$$Index_{to\_warp2} = Cracks(Median(D_{warped2})), \tag{6}$$

where $Median$ is a median filter with a 3×3 window, and $Cracks$ detects pixels that have changed during median filtering.

Step 3. Texture crack filling by inverse warping. The cracks on warped textures are filled in by inverse warping, which is warping from the new view to the reference camera views. This covers the following relations:

$$[D_{warped1}, T_{warped1}] = Warp^{-1}(Index_{to\_warp1}), \tag{7}$$

$$[D_{warped2}, T_{warped2}] = Warp^{-1}(Index_{to\_warp2}). \tag{8}$$

Step 4. Create the texture for the new view. Blending (function $Blend$) the two warped textures and the inpainting the resulting image gives

$$[D_{new}, T_{new}] = Inpaint(Blend(T_{warped1}, T_{warped2})), \tag{9}$$

where $Inpaint$ defines inpainting procedure as described in Section 3.3.

Let us now discuss two ways of quality measurements: PSNR evaluation involving the camera configuration and distortion (PSNR) of a rendered view depending on the video compression for the surrounding camera views. Since the number of cameras is limited, the camera setup is of primary importance for obtaining a good quality of free-viewpoint rendering. The first series of measurements evaluates the quality of the rendering while varying the angle between the two nearest cameras. This measurement technique has been described in [5]. The RGB images are first transformed to the YUV color

space. Then the Peak Signal-to-Noise Ratio (PSNR) of the Y values is calculated. The results are depicted in Fig. 15 and 16, where our measurements correspond to an average over 20 randomly chosen frames. We can see that

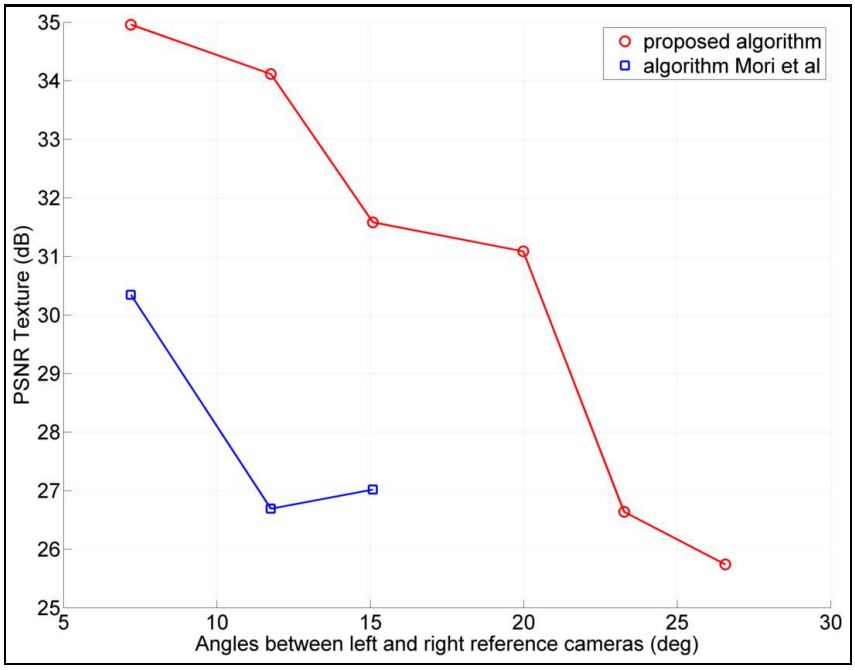

Figure 15: PSNR versus angle between cameras for the 'Ballet' sequence

our rendering algorithm increases the average PSNR with 3 dB and 4.5 dB, for the 'Breakdancers' and 'Ballet' scenes, respectively, as compared to the results presented in [5]. The large difference in PSNR is caused by the fact that we only use median filtering instead of bilateral one and our inpainting is depth based. The subjective quality difference is smaller, we have only noticed some differences on the edges of objects and the smoothness of the pictures.

We also investigate the influence of the coding on the rendering quality.

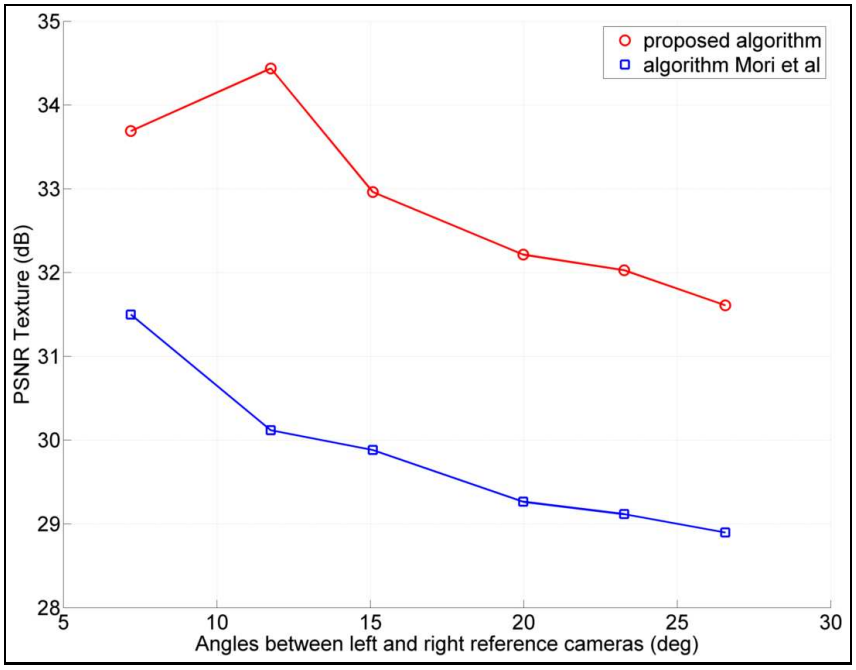

Figure 16: PSNR versus angle between cameras for the 'Breakdancers' sequence

Morvan *et al.* [4] have developed a method for calculating the optimal joint texture/depth quantization settings for the encoder. We have encoded the surrounding camera streams with the regular settings of H.264 [12]. In order to find the optimal joint quantization settings, the joint depth/texture Rate-Distortion (R-D) surface must first be created. Similar to [4] and [1], we have performed a full search to find the minimal distortion because we are only interested in the optimal settings and not in the complexity of the search. The rendering quality is expressed as a maximal PSNR for every joint bitrate. The R-D surfaces for H.264 video are illustrated in Fig. 17 and 18. To minimize the number of outliers in these surfaces, measurements are taken as average values of 10 randomly chosen frames. The applied

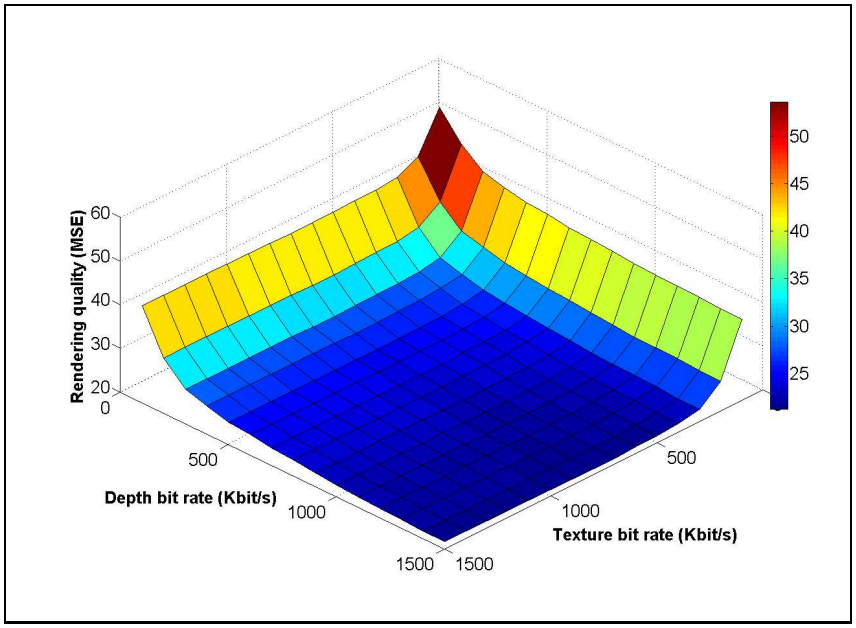

Figure 17: R-D surface for 'Ballet' with H.264 video compression

data sets are the 'Breakdancers' and 'Ballet' sequences. Each video contains 100 frames of texture and its associated depth maps with a resolution of 1024 by 768 pixels. The scene is recorded with 8 cameras, positioned along an arc spanning about 30° from one end to the other. The depth maps are created off-line and give an indication of the depth of each pixel in the image. For high-quality rendering, the depth maps must be very accurate. From Fig. 19, it can be observed that starting at the texture+depth bitrate equal to 1500 kbit/s, the rendering quality does not change much. It means that for this and higher bitrates the rendering quality is only influenced by the rendering algorithm and not by the compression of the video streams from the neighboring cameras.

Taking into account that viewpoint interpolation is important for 3D

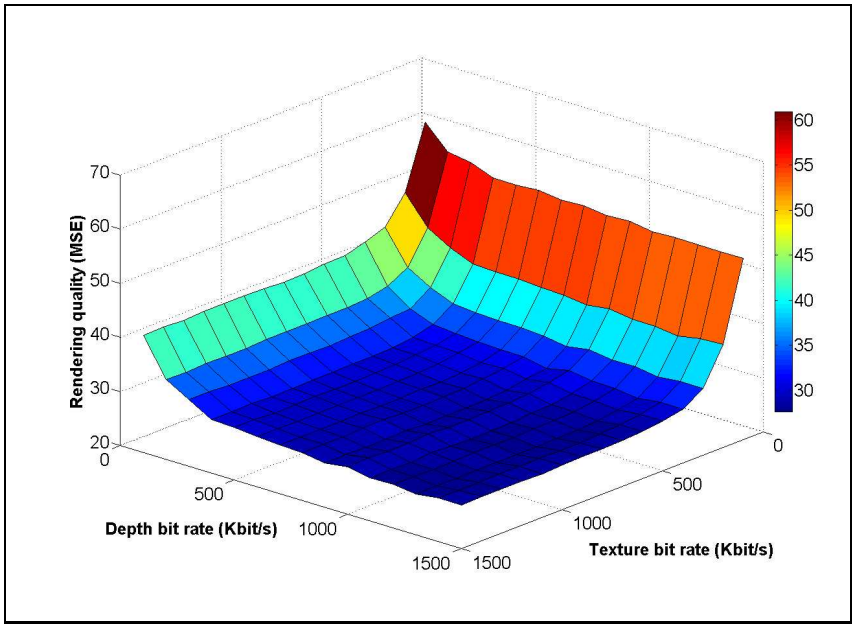

Figure 18: R-D surface for 'Breakdancers' with H.264 video compression

medical imaging, as indicated in [13] and [8], we experiment with rendering a real-world 3D medical data (Fig. 20). From Fig. 21 we can see that the acceptable quality of rendering can be obtained for small angles between surrounding cameras. Currently we are implementing our rendering algorithm on a hardware system, where the angle between the cameras is about 1°. From Fig. 21, we can expect an acceptable quality of the interpolated view for this angle. Rendering medical data as shown above presents some new challenges. For example, unlike the 'Ballet' and 'Breakdancers' sequences, a 3D model of vessels may contain two, three or more layers of objects. This makes the occlusion filling procedure more error prone. We measure the PSNR for the depth maps, since we assume that the accuracy of the rendering is proportional to the accuracy of a depth map for a new view. We notice

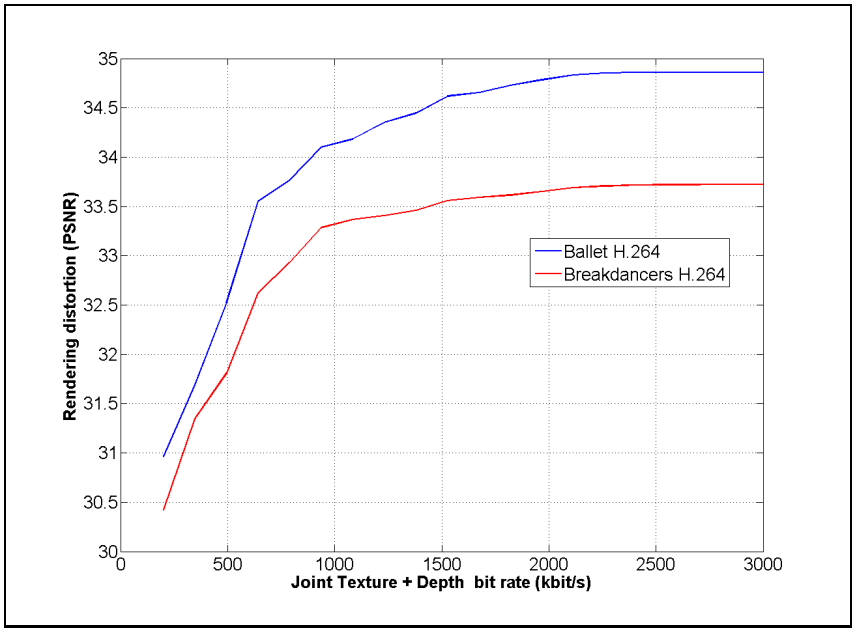

Figure 19: Rendering quality for video encoding with optimal texture/depth bitrates

that the PSNR for textures of the medical data is higher than the PSNR for depth maps. Based on our experiments, we conclude that choosing higher frequency texture elements (texels) for 3D data, results in a lower PSNR score for textures than choosing lower frequency texels. The quality of rendering for medical applications is crucial because the physician's decisions are influenced by the observed images.

5. Conclusions

In free-viewpoint video, the user can ideally navigate through the 3D domain and select his own viewpoint. This paper has explored a rendering algorithm that enables to compute a free-viewpoint between two reference views from existing cameras. This approach works best if a good depth map

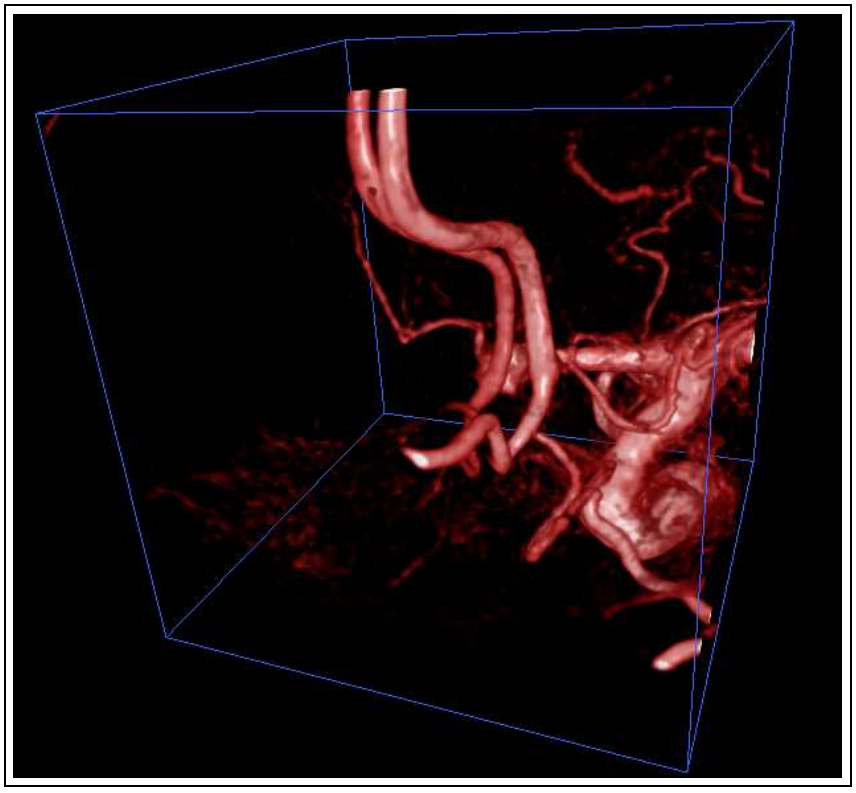

Figure 20: Visualization of blood vessels which is based on real 3D data measurements

is provided for each reference camera view. For rendering, we have analyzed the challenges of DIBR and addressed the following inherent artifacts and the associated solutions. First, the cracks produced by rendering are filled in by performing inverse warping. The advantage of this approach, described in Section 3.1.3, is that the cracks are filled in without the costly procedure of depth map generation for a new view. Second, we have proposed a disocclusion processing method that omits warping of edges at high discontinuities (Section 3.2.3). Although simple, this method produces good results. Third, we have introduced a new disocclusion inpainting approach which explicitly

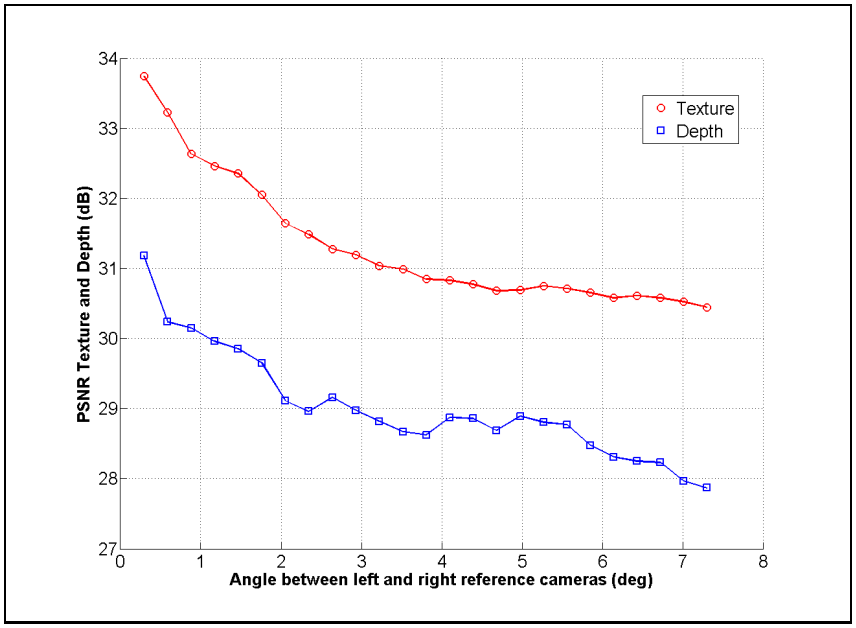

Figure 21: Visualization of blood vessels which is based on real 3D data measurements

uses depth information, see Section 3.3. This approach prevents deterioration of the texture quality caused by interpolation.

In total, our novel free-viewpoint DIBR algorithm clearly outperforms the existing proposals, e.g. [3] and [5]. From an objective perspective, it can be observed that our algorithm outperforms an earlier DIBR method when the angle between the reference cameras varies. Furthermore, we have demonstrated that using encoding, it is possible to compress the data considerably without much sacrificing the rendering quality. For future work, we plan to improve our disocclusion filling method, since we think that this will further enhance the perceptive rendering quality. We are also interested in the analysis of rendering errors [6] since it may lead to improving the quality of the results. Another possible research direction is adapting the compression rate

to the predicted rendering quality based on the scene complexity and camera positions.

While developing our rendering algorithm, we have pursued a good quality while keeping the amount of necessary operations minimal. Therefore, we expect that this algorithm can be implemented in hardware and used in an interactive 3D TV as well as in other systems requiring rendering, such as 3D medical visualization. Our rendering may also be useful in currently developing multi-view coding algorithms [9], [10].

References

[1] L. Do, S. Zinger, Y. Morvan, P. H. N. de With, Quality improving techniques in DIBR for free-viewpoint video, Proceedings, 3DTV Conference 2009: The True Vision - Capture, Transmission and Display of 3D Video, Potsdam, Germany, 2009.

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

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

[4] Y. Morvan, D. Farin, P. H. N. de With, Joint depth/texture bit-allocation for multi-view video compression, Proceedings, Picture Coding Symposium (PCS), Lisboa, Portugal, 2007.

[5] Y. Mori, N. Fukushima, T. Yendo, T. Fujii, M. Tanimoto, View generation with 3D warping using depth information for FTV, Image Communication, vol. 24, issue 1-2, 2009, pp. 65-72.

[6] H. T. Nguyen, M. N. Do, Error analysis for image-based rendering with depth information, IEEE Transactions on Image Processing, vol. 18, no. 4, 2009, pp. 703-716.

[7] K.-J. Oh, Y. Sehoon, Y.-S. Ho, Hole-filling Method using Depth based In-painting for View Synthesis in Free Viewpoint Television (FTV) and 3D Video, Proceedings, Picture Coding Symposium (PCS), Chicago, USA, 2009.

[8] D. Ruijters, S. Zinger, IGLANCE: transmission to medical high definition autostereoscopic displays, Proceedings, 3DTV Conference 2009: The True Vision - Capture, Transmission and Display of 3D Video, Potsdam, Germany, 2009.

[9] A. Smolic, D. McCutchen, 3DAV exploration of video-based rendering technology in MPEG, IEEE Transactions on Circuits and Systems for Video Technology, vol. 14, No. 3, 2004, pp. 348-356.

[10] A. Smolic, K. Muller, K. Dix, P. Merkle, P. Kauff, T. Wiegand, Intermediate view interpolation based on multiview video plus depth for advanced 3D video systems, Proceedings, 15th IEEE International Conference on Image Processing (ICIP), San Diego, USA, 2008, pp. 2448-2451.

[11] A. Telea, An image inpainting technique based on the fast marching method, Journal of Graphics Tools, vol. 9, no. 1, 2004, pp. 23-34.

[12] http://developers.videolan.org/x264.html, x264 a free h264/avc encoder, last accessed on 26 June 2009.

[13] S. Zinger, D. Ruijters, P. H. N. de With, iGLANCE project: free-viewpoint 3D video, Proceedings, 17th International Conference on Computer Graphics, Visualization and Computer Vision (WSCG), Plzen, Czech Republic, 2009, pp. 35-38.

[14] C. L. Zitnick, S. Bing Kang, M. Uyttendaele, S. Winder, R. Szeliski, High-quality video view interpolation using a layered representation, Proceedings, SIGGRAPH '04: ACM SIGGRAPH 2004 Papers, Los Angeles, USA, 2004, pp. 600-608.