DEPTH IMAGE BASED VIEW SYNTHESIS WITH MULTIPLE REFERENCE VIEWS FOR VIRTUAL REALITY

Sarah Fachada, Daniele Bonatto, Arnaud Schenkel, Gauthier Lafruit

Laboratories of Image, Signal processing and Acoustics
Université Libre de Bruxelles (ULB)

ABSTRACT

This paper presents a method for view synthesis from multiple views and their depth maps for free navigation in Virtual Reality with six degrees of freedom (6DoF) and 360 video (3DoF+), including synthesizing views corresponding to stepping in or out of the scene. Such scenarios should support large baseline view synthesis, typically going beyond the view synthesis involved in light field displays [1]. Our method allows to input an unlimited number of reference views, instead of the usual left and right reference views. Increasing the number of reference views overcomes problems such as occlusions, tangential surfaces to the cameras axis and artifacts in low quality depth maps. We outperform MPEG's reference software, VSRS [2], with a gain of up to 2.5 dB in PSNR when using four reference views.

***Index Terms* —** View synthesis, depth image based rendering, free navigation

1. INTRODUCTION

Free viewpoint navigation is an increasingly important domain in virtual reality. Several standardization committees are actively working on it, such as MPEG (with the MPEG-I Visual group), to enhance the viewer experience in emerging domains such as immersive rendering and holographic displays [3]. We propose a new software which overcomes most of the common challenges of view synthesis used in these domains, by applying more input views.

In the context of view synthesis for free navigation, the available datasets often contain a huge number of views and corresponding depth maps, e.g. the ULB dataset [4] used in MPEG. However, many depth image-based rendering [5] methods limit themselves to use a low number of reference views to synthetize virtual ones. For example MPEG's reference software VSRS [2] proceeds only with two reference views, the left and right views, also used to compute the corresponding depth maps with stereo matching algorithms [6]. Moreover, the baseline between the input views is often required to be small, restricting the view synthesis to positions close to the reference cameras. This is also the case for light fields [1], where the high density of input views does not allow huge perspective changes in every direction.

We developed a view synthesis software that surpasses the limitations on the number of reference views of VSRS. First, increasing the number of input views helps in overcoming large holes due to disocclusions. Disocclusions show parts of the scene which remain invisible in the reference views and that are usually only partially solved by inpainting, as [7]. Another objective is to render the objects which surfaces are very tangential to the camera optical axis. For such a surface, very few information about depth and color is available per pixel. In that case, the background (or empty pixels) might be seen through the foreground objects in the synthesized view, in a process similar to the creation of holes due to disocclusions. Finally, increasing the number of views also conceals imperfect depth maps: the low quality of the depth is compensated by the redundancy of the additional views information.

Using multiple reference views implies to blend all the obtained results. While some algorithms only pick one or two main views and fill disocclusions with information from additional views [8], we have chosen to not prioritize any view and warp all the input views to blend them at the end of the algorithm. This is more stable to color changes across the views. In order to blend our synthesized images together, the images are separated in two frequencies and weighted differently for each frequency, in an approach similar to [9].

2. PROPOSED METHOD

Our algorithm consists in synthesizing the target view once for each input view, and blending all the results together by assigning a per-pixel quality to each synthesized view. The resulting image is inpainted to fill the remaining missing information, mainly at the borders of the image, as the disocclusions are mostly handled by the multiple input views. The six degrees of freedom can be obtained, provided the reference views contain enough data to create the new virtual view. An example of view synthesis from four reference images is displayed in figure 1.

2.1. Warping

The first step consists in warping the inputs using directly the disparity rather than performing back and forth 3D re-projection of the pixels, which is sensitive to rounding operations. For each reference view, the translation and rotation between each input and the target cameras are applied to get the corresponding result. The translation is obtained by moving each pixel according to its disparity and the rotation is obtained with a homography.

For a camera movement in the image plane, for each pixel $p = (p_x, p_y)$, its translation $t = (t_x, t_y)$ in the image is given by the disparity:

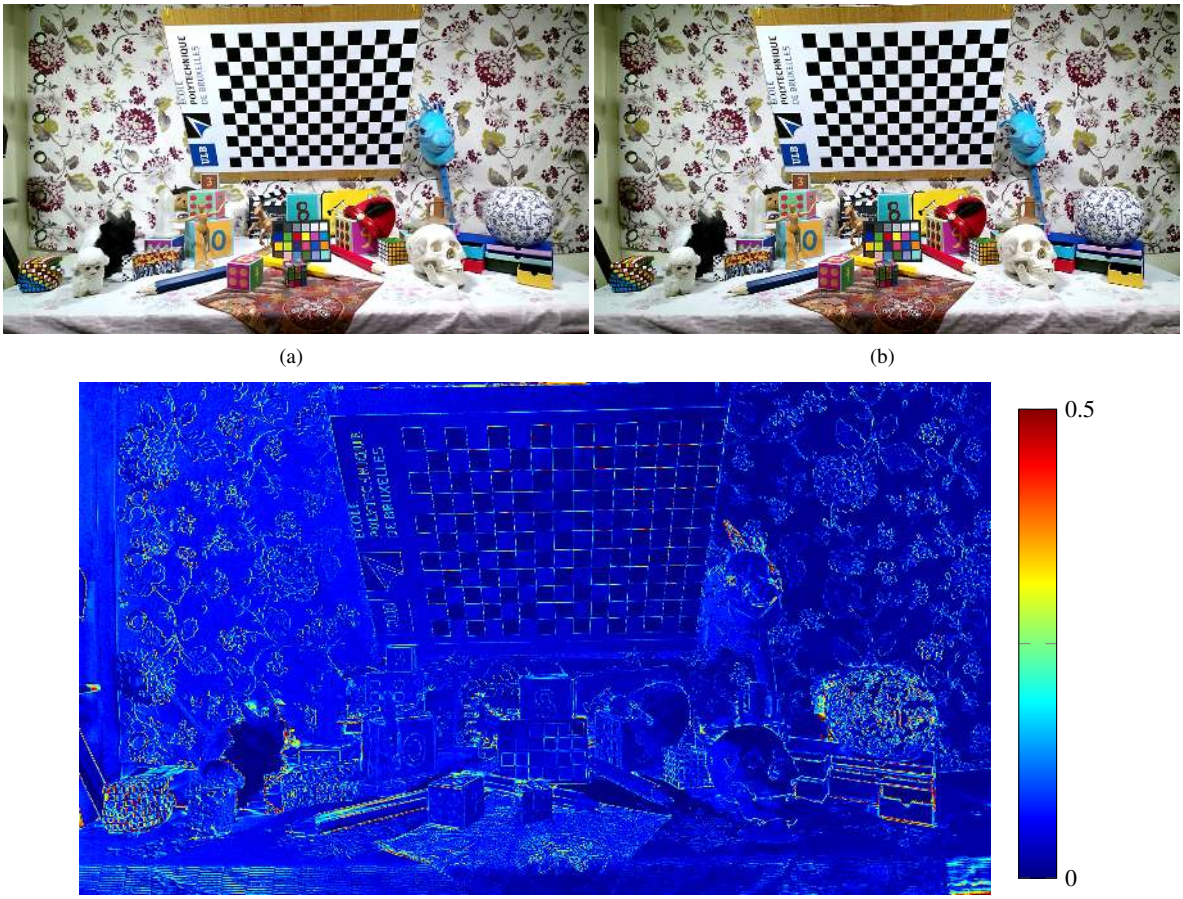
$$t = \frac{fT}{d} \tag{1}$$

where $T = (T_x, T_y, T_z)$ is the translation vector of the camera, d is the depth at (p_x, p_y) and f is the focal length.

For a movement along the Z axis of the camera, it is

$$t' = \frac{T_z(p + t)}{f\,(d - T_z)} \tag{2}$$

Three degrees of freedom of translation are obtained by combining (1) and (2). The rotation is obtained with a simple homography. After the translation, each pixel p is displaced following



(a) (b)

(c)

Figure 1: View synthesis from four input views, 100mm forward movement (step-in) towards the scene. (a) Synthesized view. (b) Reference view. (c) Normalized per pixel error (The scale only displays the error from 0 to 0.5 for readability).

the rotation of the camera to get the final result:

$$p' = (p'_x, p'_y, 1)^T = \frac{1}{R_3 \cdot (p_x, p_y, 1)^T} R(p_x, p_y, 1)^T. \quad (3)$$

where R is the rotation matrix and R_3 its last row.

With those equations, a map is obtained, indicating the new position of each pixel in the new warped image.

The input view is divided into triangles with the pixels centers as vertices. The triangles are deformed using the translation and rotation equations before being filled with interpolated colors. The colors are obtained by the tri-linear interpolation between each three vertices of the triangle. Discontinuities between objects creating disocclusions and tangential surfaces may lead to triangles with very elongated shapes (see figure 2), which will not be kept in the final result, as they get eliminated during the blending phase.

2.2. Resizing

In order to improve the final quality of the synthesized views, during the rasterization of the warped triangles, the synthesized view is upscaled. Hence, the displacement of each pixel according to its depth is more precise and the resulting image, once downscaled to the same size as the input, is sharper. Such rescale is less computationally expensive than rescaling both the input views and depth maps at the beginning of the warping phase as in VSRS. The rasterization can be GPU accelerated.

(a) (b)

Figure 2: Image warping in a step-in movement. (a) The input view, (b) The obtained view. The upper triangle is elongated, due to a disocclusion and will be discarded during the blending phase. The lower triangle has a regular shape as it lies on a continuous surface.

2.3. Blending

The following step of our view synthesis consists in blending together all the synthesized images corresponding to each input view. This is done by comparing a per-pixel quality, determined by its depth and the shape of the triangle it lies in: a pixel of good quality has a low depth (e.g. in the foreground) and belongs to a triangle

(a) (b) (c)

Figure 3: Different blending methods in acute close-up. View synthesized from two images and low quality depth map. (a) Blending by argmax, (b) Weighted mean with $\alpha = 5$, (c) Multi-spectral blending: argmax is used on high frequencies and weighted mean on low frequencies.

with a regular shape (e.g. does not lie in a disocclusion).

Taking the pixel with the maximal quality would give a sharper result, while taking the weighted mean is more resistant to errors in the depth maps, color differences between the input views and, in the case of navigation leads to smoother navigation, with less flickering. High and low frequencies are separated with a mean blur with a kernel of size of 10% of the image size, and the most adapted blending is applied to each frequency (see figure 3). The low frequencies are blended with the weighted mean, and the high frequencies are blended by choosing the pixel of highest weight. The final color c of a pixel is hence:

$$c = c_{high} + c_{low} \tag{4}$$

$$c_{low} = \frac{\sum_{i=0}^{n} w_i c_i^l}{\sum_{i=0}^{n} w_i} \qquad \text{with} \quad w_i = \left(\frac{q_i}{d_i}\right)^{\alpha} \tag{5}$$

$$c_{high} = \text{argmax}_{c_i^h}\left(w_i\right) \tag{6}$$

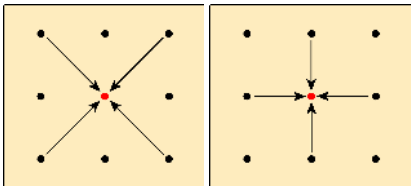
where n is the number of input views, c_i^l, c_i^h the color of the pixel in view i for the low and high frequencies, q_i the quality of the triangle the pixel lies in, α a parameter and d_i the depth at this pixel for the synthesized view i. With the factor $\frac{1}{d_i}$, foreground objects are prioritized even if they do not appear in all the input views. The factor q_i removes the elongated triangles (see figure 2). Hence, the disocclusions can be filled with data from other input views, which would not be the case comparing only the depth of the pixels. We have chosen to define

$$q_i = \begin{cases} 2\mathcal{A}/b^2 & \text{if the triangle keeps the same} \\ & \text{orientation during warping,} \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

where $\mathcal{A}$ is the area of the triangle, b is the length of the second longest side of the triangle. We can verify that we always have $0 \leq q_i \leq 1$ and if the triangle keeps its shape during the warping phase $q_i = 1$. We can choose other formulas for q_i, based on the direction of the normal of the triangle in a 3D projection compared with the direction of the input camera and obtain similar results. However, as we never project the pixels in 3D neither create meshes, this solution would need additional computation.

2.4. Inpainting

The last step consists in inpainting the blended view. Most of the disocclusions are filled by data of the input views and the remaining disocclusions are already filled with pixels lying in "low quality" triangles. However, in the case of backward movements, the input view does not contain the information out of the image, which has to be inpainted.



(a) (b)

Figure 4: Camera configuration for four input views. (a) "X" Configuration. (b) "Plus" Configuration.

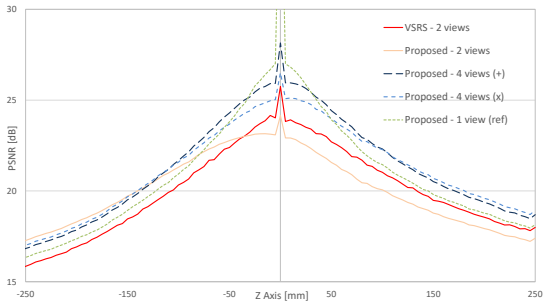


Figure 5: PSNR for various camera configurations and for VSRS, using depth maps generated with DERS, in step-in and step-out experimental conditions. The 4 views are disposed in "plus" configuration (+) and "X" configuration (x) with all the cameras parallel in the same plane. The "X" configuration has a slightly wider baseline.

3. RESULTS

3.1. Quantitative results

We compare our view synthesis for various configurations with MPEG's reference software, VSRS [2] (see figure 5). We have performed the test on the ULB unicorn dataset [4]. We reach gains of 1.8 dB for view synthesis corresponding to lateral movements, 2.5 dB for step-in and 1dB for step-out scenarios, using four views instead of the two reference views in VSRS. We tried different configurations: one, two and four views (see figure 4 for the four-views configurations). As expected, we notice that using wide baseline (4 views (x) configuration in figure 5) is preferred for step-in and step-out configurations, while a lower baseline distance (4 views (+) configuration and 1 view) should be used preferentially for movements in a small volume around the input views.

3.2. Subjective results

Beyond the gain in PSNR, we notice a visual improvement during navigation and more resilience against errors in depth maps. For the synthesis of several frames, as we use only the data of the input views and very few inpainting, the resulting images do not flicker when free-navigation is used, making a video more appealing to watch. In contrast, inpainting the output of a two views based syn-

(a) (b)

Figure 6: View synthesis with low quality depth map, from 2 input views, step-in configuration. (a) VSRS, (b) Proposed.

thesizer without propagating the inpainting between each frame, as in VSRS, gives a sequence of images which is very unstable for video synthesis and hence for navigation.

Eventually, we observe more stability for moderate quality depth maps (see figure 6). Thanks to the quality given during the blending phase to the pixels of each synthesized view, we are more likely to find good pixels than methods based on a limited number of reference views. Hence, increasing the number of views not only generates output with higher PSNR values, but also gives the opportunity to create a content more enjoyable for the end-user.

4. CONCLUSION AND FUTURE WORK

We solved the problem of disocclusions by using multiple reference views, increasing the probability that occluded regions are captured by at least one reference camera. Due to a higher number of views, most of those disocclusions can be filled. The remaining holes of the synthesized views are filled with a plausible color thanks to the division of the reference views into fictitious triangles connecting adjacent pixels and thanks to inpainting. Furthermore, this way to overcome the disocclusions creates smoothness between the output views, which gives a more pleasant navigation in video applications. Moreover, our triangulation of the reference images increases the quality of the synthesis for tangential surfaces, which can be completely rendered with the interpolated pixels of the triangles.

Our results have demonstrated that the PSNR increases by augmenting the number of input references, especially in the case of large movements (step-in and step-out scenarios). Our algorithm has therefore recently been selected as the core for a new reference software (Reference View Synthesizer) in the MPEG-I standardization activities for 360 virtual reality with motion parallax.

Nevertheless, taking more input views increases memory footprint and computation time (up to 10 seconds for an input view of 1920×1080 pixels). In future work, we plan to implement our solution on GPU in order to reduce our computation time.

5. ACKNOWLEDGEMENTS

This work is supported by Innoviris, the Brussels Institute for Research and Innovation, Belgium, under contract numbers 2015-DS-39a/b & 2015-R-39c/d, 3DLicorneA.

6. REFERENCES

[1] Marc Levoy and Pat Hanrahan, "Light field rendering," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 31–42.

[2] Takanori Senoh, Kenji Yamamoto, Nobuji Tetsutani, Hiroshi Yasuda, and Krzysztof Wegner, "View synthesis reference software (VSRS) 4.2 with improved inpainting and hole filing," *ISO/IEC JTC1/SC29/WG11, M40657*, Apr, 2017.

[3] Tibor Balogh, Tamás Forgács, Tibor Agács, Olivier Balet, Eric Bouvier, Fabio Bettio, Enrico Gobbetti, and Gianluigi Zanetti, "A scalable hardware and software system for the holographic display of interactive graphics applications.," in *Eurographics (Short Presentations)*, 2005, pp. 109–112.

[4] Daniele Bonatto, Arnaud Schenkel, Tim Lenertz, Yan Li, and Gauthier Lafruit, "ULB High Density 2d/3d Camera Array data set, version 2," *ISO/IEC JTC1/SC29/WG11 MPEG2017/ M41083*, July 2017.

[5] Christoph Fehn, "Depth-image-based rendering (dibr), compression, and transmission for a new approach on 3d-tv," in *Stereoscopic Displays and Virtual Reality Systems XI*. International Society for Optics and Photonics, 2004, vol. 5291, pp. 93–105.

[6] Takanori Senoh, Kenji Yamamoto, Nobuji Tetsutani, and Hiroshi Yasuda, "Improved fast DERS," *ISO/IEC JTC1/SC29/WG11, M41028*, Jul, 2017.

[7] Alexandru Telea, "An image inpainting technique based on the fast marching method," *Journal of graphics tools*, vol. 9, no. 1, pp. 23–34, 2004.

[8] Ce Zhu and Shuai Li, "Multiple reference views for hole reduction in dibr view synthesis," in *Broadband Multimedia Systems and Broadcasting (BMSB), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 1–5.

[9] Ruggero Pintus, Enrico Gobbetti, and Marco Callieri, "Fast low-memory seamless photo blending on massive point clouds using a streaming framework," *Journal on Computing and Cultural Heritage (JOCCH)*, vol. 4, no. 2, pp. 6, 2011.