Compact quad-based representation for 3D video

Thomas Colleu, Luce Morin, Claude Labit, Stéphane Pateux, Raphaèle Balter

▶ **To cite this version:**

Thomas Colleu, Luce Morin, Claude Labit, Stéphane Pateux, Raphaèle Balter. Compact quad-based representation for 3D video. 3DTV-Conference 2009, The True Vison, Capture, Transmission and Display of 3D Video, May 2009, Postdam, Germany. pp.1–4. hal-00457585

HAL Id: hal-00457585
https://hal.science/hal-00457585

Submitted on 9 Mar 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

COMPACT QUAD-BASED REPRESENTATION FOR 3D VIDEO

T. Colleu[1,3] *L. Morin*[2] *C. Labit*[1] *S. Pateux*[3] *R. Balter*[3]

[1] INRIA, Centre Rennes - Bretagne Atlantique - Campus de Beaulieu - 35042 Rennes
[2] Université Européenne de Bretagne, INSA/IETR - 20, Avenue des Buttes de Coësmes - 35043 Rennes
[3] Orange Labs - 4, rue du Clos Courtel - 35512 Cesson-Sévigné

ABSTRACT

The context of this study is 3D video. Starting from a sequence of multi-view video plus depth (MVD) data, the proposed quad-based representation takes into account, in a unified manner, different issues such as compactness, compression, and intermediate view synthesis. The representation is obtained into two steps. Firstly, a set of 3D quads is extracted by using a quadtree decomposition of the depth maps. Secondly, a selective elimination of the quads is performed in order to reduce inter-view redundancies and thus provide a compact representation. Experiments on two real sequences show good quality results at the rendering stage and a small data overload compared to mono-view video.

***Index Terms*—** 3D video, data representation, multiview video plus depth, quadtree, 3D quads, compression.

1. INTRODUCTION

3D video is expected as the logical evolution of 2D video. Two types of 3D video applications are envisioned. The first one, called 3DTV for *3 dimensional television*, provides a relief sensation to the user by reproducing the human binocular vision. Display devices allowing 3D visualization are available. They may not require to wear special glasses. They display two views (autostereoscopic displays) and even N=8,10,12... views (multiview autostereoscopic displays) in order to maximize the user comfort. The second application, called FTV for *Free-viewpoint TeleVision*, allows for interactive selection of the viewpoint and direction in the scene within a certain operating range.

In order to achieve 3DTV and FTV, one can capture all the views required at the rendering stage. This method may be used for stereoscopic video (2 views) but it can hardly be generalized to N views due to acquisition and storage complexity. An alternative is to reduce the number of cameras and to synthesize the required intermediate views by using information about the geometry of the scene. Many studies are being conducted on this issue. In particular, within the normalization group ISO-MPEG, the working group FTV is currently studying the representation and coding of multiview data in order to achieve a compression standard suitable for 3D video. Here, the quality of synthesized intermediate views is fundamental.

In this context, this paper presents a representation based on 3D polygons that takes, as an input, MVD data (*Multi-View plus Depth*) and that is appropriate for both the compression, transmission and rendering stages. Section 2 presents the previous work on 3D video using depth maps. Then, an overview of the proposed representation is given in section 3. Sections 4 and 5 detail the main steps of the method. Section 6 presents the results.

2. 3D VIDEO USING DEPTH MAPS

Considering 3D video applications displayed on multiview autostereoscopic screens, interest for depth image-based representations has increased a lot. A depth map is an image that associates one depth value (i.e. the distance to the camera) to each pixel. It enables to synthesize intermediate views using a perspective projection method.

Depth image-based representations. The simplest representation consists of using only one view made up of an image plus a depth map per time instant (2D+Z) [1]. But occluded regions are not contained in the depth map, and therefore disocclusion regions are not well reconstructed during view synthesis and might create strong visual artifacts.

For a wider range of viewpoints, multiple views made of 2D+Z data must be used. It is called MVD (Multi-view Video Plus Depth) data and enables to synthesize an intermediate view based on a set of views. This gives a very good quality since most of the occluded regions in one view can be filled with the other views [2, 3]. The redundancies between all views are usually high since the same scene is captured from several views, therefore the data load is high.

In order to deal with both the disocclusion areas and inter-view redundancies, a solution is to select a certain view as reference and extract, from the other views, only the information which is not contained in the reference view, i.e. the occluded areas [4, 5, 6]. This is called LDV (Layered Depth Video). The advantage is that the inter-view redundancies are reduced while the disocclusion areas are available. However, some color differences can appear since only a central view plus the occlusion areas are used. Moreover, the construction

and compression of such data is still an open problem. In the Layered Depth Images (LDI) [4], all the side information is projected into the reference view so that one pixel can contain multiple depth and color values, yet this leads to a loss of quality due to resampling during the projection.

In addition, many contributions in the field of global model reconstruction present efficient algorithms for merging multiple depth-maps and polygonal meshes [7, 8]

Depth maps compression. Depth maps are gray level images, so they can be compressed with an efficient video codec such as H.264. However, depth maps describe the surface of a scene and have different properties compared to an image describing the texture. Therefore, rendering intermediate views using compressed depth maps creates visually disturbing artifacts, especially around depth discontinuities (objects boundaries) as studied in [9]. With this in mind, a platelet-based depth coding algorithm has been proposed [10]. This algorithm employs a decomposition of the depth maps using geometric primitives such that the depth discontinuities are preserved. This method provides a better rendering quality for a given compression rate.

Depth based rendering. Rendering intermediate views using depth maps is generally considered as a point-based method: each pixel is independently projected in 3D and then projected into the desired intermediate view. As a result, many small holes appear in the intermediate view and must be filled with post-processing techniques [3]. An alternative is to transform the depth maps into a surface using geometric primitives such as triangles [2] or quadrilaterals [11] and to disconnect these primitives at depth discontinuities so that the background and foreground are not connected. This solution eliminates the post-processing stage but requires a graphic processor.

3. PROPOSED REPRESENTATION

In section 2, some issues have been identified concerning data compactness, depth map compression, and intermediate view synthesis. The contribution of this study is to propose a new data representation that takes into account all these issues as a whole.

This representation is based on a set of 3D polygons that are defined with 2D+Z data: a polygon is delimited by a block of pixels in one view; the polygon's depth is defined by the depth information at the corners of the block; the polygon's texture is given by the block's texture in the image.

Polygonal geometric primitives have several advantages. First, the size of the polygons can be adaptively determined so as to keep the number of polygons low. Thus a compact representation and an efficient rendering stage can be obtained (such as in [11]). Second, as presented in [10], a polygonal decomposition of the depth maps that preserves the object boundaries results in a compression algorithm offering a better rendering quality compared to an H.264 algorithm, for a given compression rate. Third, polygons are frequently used as primitives at the rendering stage ([2, 11]) since they model the continuity of the surface and thus avoid some post-processing operation that fill empty pixels in the rendered image [3]. On the contrary to [2, 11] where the polygons are used only at the rendering stage, the representation proposed here is directly based on polygons. Polygon extraction will be explained in section 4.

Finally, polygons are selectively eliminated to reduce redundancies between views (similarly to [6]), and also to reduce artifacts around discontinuities due to mixing colors between the background and foreground. This technique is presented in section 5. Figure 1 sums up the different steps of the method as well as its application framework (compression and rendering of a different number of views).

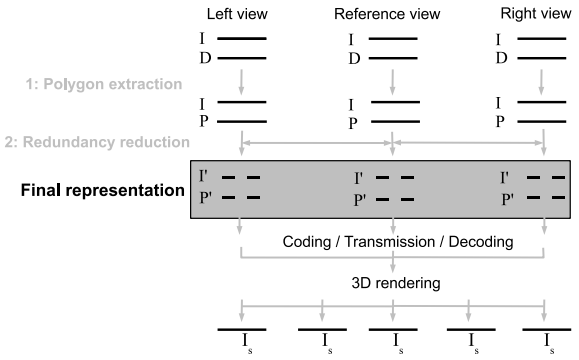

Fig. 1. Construction of the representation. I: image, D: depth, P: polygons, I'/P': I/P reduced, I_s: synthesized image

4. POLYGON EXTRACTION

In this section, each depth map is processed independently so that a set of polygons is created for each view. The chosen polygon type is the quadrilateral (quad). A quadtree decomposition is used to extract and structure the set of quads. This choice is motivated by the context of video coding. Indeed, the texture information will probably be coded based on a classical video coding algorithm such as a block based coding, then the coherence between the block-based depth information and texture information can be exploited. An example of a quadtree decomposition is shown in figure 4. The decomposition technique used here is divided into two steps which we call discontinuity preservation and geometry refinement:

Discontinuity preservation. The blocks in the image are sub-divided based on a depth criterion. More precisely, let $p1$ and $p2$ be two pixels in the block B and their depth be Z_{p1} and Z_{p2}. Considering threshold T_d, then the block is sub-divided if:

$$\max_{p1,p2 \text{ neighbors } \in B}(abs(Z_{p1} - Z_{p2})) > T_d$$

Geometry refinement. From the coarse representation obtained, a block is again sub-divided based on a planarity

criterion. More precisely, for each pixel p of a block B and an error threshold T_p, the block is sub-divided if:

$$\max_{p \in B}(dist(P, \pi_B)) > T_p$$

where π_B is the plane approximating the depth values in B with a least squares method, and $dist(P, \pi_B)$ is the euclidean distance between π_B and the 3D point P (corresponding to pixel p). When a block satisfies this criterion, then a quad Q_B is associated to it, based on the depth values of the block's corners.

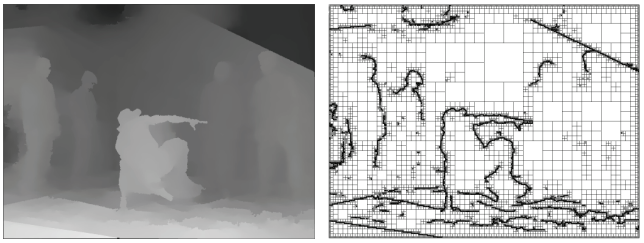

Fig. 2. Quadtree decomposition of a depth map.

5. REDUNDANCY REDUCTION

From the set of quads obtained previously, inter-view redundancies are now reduced. The proposed process also enables to reduce ghosting artifacts around depth discontinuities.

Let Q_d be the desired set of quads extracted from all the views after redundancy reduction, and Q_i the set of quads from view i. The idea is to initialize Q_d with the quads from a reference view V_r and iteratively complete and modify Q_d with Q_i, $i = 1$ *to* N, $i \neq r$. Let i be the current iteration. Q_d is first projected into view V_i. The resulting image contains disocclusion areas. Then the quads from Q_i are added to Q_d if the pixel block that they form in view V_i cover these disocclusion areas. Figure 3 (left) shows the projection of Q_d in V_i. The disocclusion areas can be seen in white. Figure 3 (right) shows the quads from Q_i that are added in Q_d. The large white regions show that many redundancies have been eliminated.

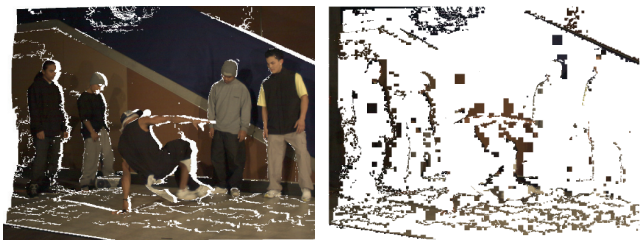

Fig. 3. Redundancies reduction. Left: Projection of Q_d in V_n. Right: Quads from Q_i added in Q_d

Many small quads (in pixel size) are present around discontinuities. They have low resolution and may contain mixed colors between background and foreground (ghosting artifacts). Therefore, it is useful to replace them, where possible, with bigger quads from the side views. To do so, during the projection of Q_d explained previously, the quads smaller or equal to a threshold T_s are not projected. Thus, more quads from Q_i are added to Q_d, creating overlapping areas. Then the quads that fully overlap (in 2D) with bigger quads from Q_i are eliminated. Here, a depth test with threshold T_o is performed to identify adjacent quads. Figure 4 shows the dancer's hand in front of the wall. Picture (a) shows the projection of the set Q_d before the elimination of the quads. The outline of the hand appears in the wall (ghosting artifact). The grey quads are from the previous iteration and the black quads are the new ones added during this iteration. These quads are overlapping. Pictures (b) shows the results after the elimination of the quads. The ghosting artifact is reduced and many small quads have been suppressed.

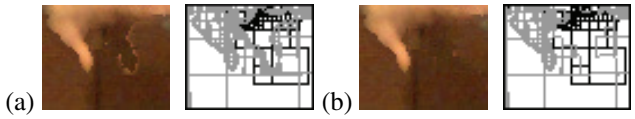

Fig. 4. Elimination of small overlapped quads.

6. RESULTS

Tests were performed on MVD sequences *Breakdancer* and *Ballet*[1]. They were captured with 8 cameras (resolution 1024x768) placed on a horizontal arc spanning about 30°. The depth maps were estimated with a stereo algorithm [2].

Polygon extraction was performed with empirical thresholds $T_d = 10$ for 8 bits depth values, and $T_p = 0,6$. Figure 2 shows the quadtree decomposition. Each depth map contains 786432 pixels. The average number of quads per view is 26671 for *Breakdancer* and 30815 for *Ballet*. Using these quadrilaterals and corresponding textures, intermediate views have been synthesized. Figure 5 shows a zoomed region of the scene where the data from view V_1 has been synthesized into view V_2. The rendering result can be observed with a point-based representation (depth map) (a) and a quad-based representation (b). In (a), a post-processing algorithm is necessary to fill the small disocclusion areas (thin white lines). In (b), the continuity of the surface is preserved and only large depth discontinuities create disocclusion areas.

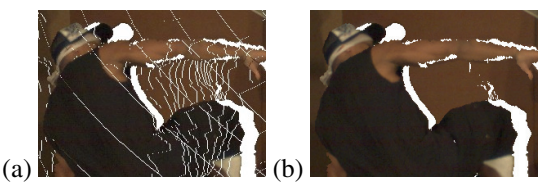

Fig. 5. Comparison point-based VS quad-based rendering

[1]Thanks to the Interactive Visual Media Group of Microsoft Research for providing the data sets.

Redundancy reduction was performed with empirical thresholds $T_s = 2$ (2x2 pixel block) and $T_o = 10$. The following experiments were tested with a configuration of 3 consecutive views (V_1, V_2, V_3) where the central one is considered as the reference. Table 1 gives a comparison of the number of quads before and after the redundancy reduction (first and second rows) and the number of quads for the reference view V_2 (third row). 53% of the quads have been removed compared with the full 3 views. 27% of the quads have been added compared with the number of quads for a single view V_2.

	BreakDancer	*Ballet*	
V_1, V_2, V_3	78690	89079	-53%
V_1, V_2, V_3 *reduced*	35366	41120	
V_2	26052	29775	$+27\%$

Table 1. Number of quads before and after the reduction.

Finally, from this reduced set of quads, the views V_1, V_2 and V_3 can be synthesized. Figure 6 shows the synthesis of V_1. During the rendering stage, if some pixels in the desired image receive the contribution from several quads, then their color is equally blended. As a result, a good quality image is obtained. However, some artifacts appear such as color differences (e.g. in the shadowed areas next to the second man from the right) or the unnatural sharp edges around the dancer. Moreover, the quality of the synthesized views depends on the input depth maps that may contain errors or inconsistencies across views. In order to synthesize intermediate views (e.g. between V_1 and V_2), an additional process would be necessary to fill unknown areas that are not visible in any view.

Fig. 6. Synthesis of view V_1 from V_1, V_2(reference) and V_3

7. CONCLUSION

This paper presents a quad-based representation for 3D video, that takes into account in a unified manner issues identified in the literature such as data compactness, depth maps compression, and intermediate view synthesis. A set of quads is extracted with a quadtree decomposition of the depth maps, and inter-view redundancies are reduced based on a selective elimination of quads. The results show that the quads provide a good trade-off between rendering quality and data compactness. Moreover, the first experiments show that, in the case of 3 views, the redundancy reduction allows to limit the data overload to 27% compared to mono-view video.

Future work will include a study of the visual distortions depending on the number of quads and also depending on the distance to the reference view. Moreover, the construction of the representation can be improved to better manage depth and texture errors or inconsistencies across views. Then, the coding method of this representation must be studied in order to enhance the compactness of the representation. Lastly, the temporal dimension of the video sequence will be considered to improve performance.

8. REFERENCES

[1] C. Fehn, P. Kauff, M. Op de Beeck, F. Ernst, W. IJsselsteijn, M. Pollefeys, L. Van Gool, E. Ofek, and I. Sexton, "An evolutionary and optimised approach on 3d-tv," in *In Proceedings of International Broadcast Conference*, Amsterdam, Netherlands, 2002, pp. 357–365.

[2] C.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, 2004.

[3] A. Smolic, K. Muller, K. Dix, P. Merkle, P. Kauff, and T. Wiegand, "Intermediate view interpolation based on multiview video plus depth for advanced 3d video systems," in *ICIP*, 2008, pp. 2448–2451.

[4] J. Shade, S. Gortler, L. He, and R. Szeliski, "Layered depth images," in *ACM SIGGRAPH*, 1998, pp. 231–242.

[5] W.H.A. Bruls, C. Varekamp, R.K. Gunnewiek, B. Barenbrug, and A. Bourge, "Enabling introduction of stereoscopic (3d) video: Formats and compression standards," in *ICIP*, 2007, pp. 89–92.

[6] K. Müller, A. Smolic, K. Dix, P. Kauff, and T. Wiegand, "Reliability-based generation and view synthesis in layered depth video," in *MMSP*, 2008, pp. 34–39.

[7] P. Gargallo and P. Sturm, "Bayesian 3d modeling from images using multiple depth maps," in *CVPR '05*, Washington, DC, USA, 2005, vol. 2, pp. 885–891.

[8] M. Pollefeys, D. Nistér, J. M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles, "Detailed real-time urban 3d reconstruction from video," *Int. J. Comput. Vision*, vol. 78, no. 2-3, pp. 143–167, 2008.

[9] P. Merkle, A. Smolic, K. Muller, and T. Wiegand, "Multi-view video plus depth representation and coding," *ICIP*, vol. 1, pp. 201–204, 2007.

[10] P. Merkle, Y. Morvan, A. Smolic, D. Farin, K. Muller, P.H.N. de With, and T. Wiegand, "The effect of depth compression on multiview rendering quality," in *3DTV Conference*, 2008.

[11] J. Evers-Senne, J. Woetzel, and R. Koch, "Modelling and rendering of complex scenes with a multi-camera rig," in *Conference on Visual Media Production (CVMP)*, 2004.