Floating polygon soup

Thomas Colleu, Luce Morin, Stéphane Pateux, Claude Labit

FLOATING POLYGON SOUP

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

[1] IETR/INSA Rennes. 20, avenue des Buttes de Coesmes, 35708 Rennes, France

[2] Orange Labs. 4, rue du Clos Courtel, 35512 Cesson Sevigne, France

[3] INRIA-Rennes Bretagne Atlantique. Campus de Beaulieu, 35042 Rennes, France.

ABSTRACT

This paper presents a new representation called floating polygon soup for applications like 3DTV and FTV (Free Viewpoint Television). This representation is based on 3D polygons and takes as input MVD data. It extends the previously proposed polygon soup representation [1] which is appropriate for both compression, transmission and rendering stages. The floating polygon soup conserves these advantages while also taking into account misalignments at the view synthesis stage due to modeling errors. The idea for reducing these misalignments is to morph the 3D geometry depending on the current viewpoint. Results show that artifacts in virtual views are reduced and objective quality is increased.

Index Terms— Multiview video plus depth, virtual view synthesis, 3D morphing, polygon soup.

1. INTRODUCTION

The year 2010 has seen the popularity of 3D video exploding, starting with the success of 3D movie 'Avatar' in cinemas in December 2009 and with electronic companies announcements of their 3D-ready televisions arriving at home. Here, the functionality that justifies the term '3D' is stereoscopy, where there is only one more image compared with traditional 2D video. But it is sufficient to show the potential of 3D video to improve the description of a scene and the feeling of depth for the users. Active research is now focused on **multi-view video** in order to increase the number of images of the same scene at the same time [2, 3, 4]. Multi-view video brings mainly two functionalities to the users. The first functionality is free viewpoint navigation: similarly to the famous "bullet time effect" in the movie 'The matrix', the viewer can change the point-of-view within a restricted area in the scene, thus having a navigation functionality. The second functionality is auto-stereoscopy: the viewer enjoys stereoscopic visualization without the use of special glasses. Multi-view auto-stereoscopy brings more freedom in terms of viewing position and number of users. These two functionalities raise new applications like FTV (Free viewpoint TV) and 3DTV.

The chosen **representation** plays a central role in a multi-view video system. Indeed, it influences the data load to be transmitted, the compression method to be used, as well as the computational complexity during the view synthesis stage and the final video quality at the display stage.

In this paper, we present an original representation for multiview videos that we call *floating polygon soup*. This representation takes as input multi-view video plus depth (MVD) data. It extends the previously proposed **polygon soup** representation [1] which is appropriate for both compression, transmission and synthesis stages. The floating polygon soup conserves these advantages while also taking into account artifacts at the synthesis stage.

Section 2 gives a state-of-art about depth image-based representations and related view-synthesis issues. In section 3, an overview of the previously proposed *polygon soup* representation is given. The *floating polygon soup* representation is described in section 4. Section 5 presents quality evaluation of intermediate view synthesis obtained with the proposed representation and discusses the issues raised by our representation.

2. STATE OF ART

Depth image-based representations. Existing representations for multi-view video often contain some geometric information about the scene in addition to color information, such as depth image-based representations. A depth map is an image that associates one depth value (i.e. the distance to the camera) to each pixel. It enables

T. Colleu performed the work while at Orange Labs. This work is supported by the RUBI3 and PERSEE project

to synthesize intermediate views using a perspective projection method. A depth-based representation can be composed of a single viewpoint (a.k.a 2D+Z [5]) or multiple viewpoints (Multiview Video plus Depth MVD [6, 7]). Moreover, in order to deal with the compromise between data load due to multiple viewpoints and image quality, the Layered Depth Video (LDV) selects 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 [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, synthesizing intermediate views using compressed depth maps creates visually disturbing artifacts, especially around depth discontinuities (objects boundaries) as studied in [9]. With this in mind, several approaches have been proposed to compress depth maps while preserving depth discontinuities [10, 11, 12].

Depth based view synthesis. Synthesizing intermediate views using depth maps is generally performed using a point-based method: each pixel is independently reconstructed in 3D and then re-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 [7]. An alternative is to transform the depth maps into a 3D surface using geometric primitives such as triangles [6] or quadrilaterals [13] 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.

During intermediate view synthesis, multiple images are warped or projected into the novel view. A combination strategy is needed to obtain one texture from multiple candidates [14]. View-dependent combination of the multi-images into the novel view provides better photo-realism and helps to reproduce non-Lambertian reflectance effects [15, 16]. In addition, special processing of the occlusion boundaries is used to reduce so-called ghosting artifacts in the novel views [7, 17].

Modeling errors. However, due to modeling errors, images are not warped or projected to the expected position into the novel view, creating local texture misalignments. These misalignments appear in the form of texture deformation if only one image is used, or splitting and blur if multiple images are combined.

Existing solutions correct texture misalignments through matching and warping techniques. *Floating textures* [17] have been proposed for correcting these misalignments during view synthesis. The idea is to compute motion flow between synthesized textures coming from multiple views (matching step) and then use the estimated flow to warp them onto each other so that to reduce texture misalignments (warping step). A similar method was proposed by Takai et al. [18]. In this method, the computation of texture coordinates is reduced to only vertex positions instead of computed motion flow over the whole images. This is computed as a pre-processing step together with a mesh optimization process. Furihata et al. [19] proposed another solution to compute texture correction in a pre-processing step. They compute intensity residual error of each view projected into another one (differencing step rather than matching). Then during view synthesis, the residual error is fed back into the virtual view and processed according to the virtual position (addition step rather than warping). The main difference with the previous methods is that texture misalignment is computed in terms of intensity error instead of motion field which is computationally less expensive but may be less efficient for correcting misalignments.

Finally, in the temporal domain, there are also methods that match and morph the geometry across time in order to reproduce objects motions. Vedula et al. [20] introduced the *3D scene flow* with a volumetric representation. The 3D scene flow gives the motion of each voxel of the representation between two frames of the video. It is computed using 2D motion flow and 3D data. The aim is to reproduce the motion of the scene, but it can also be thought as a problem of texture misalignments between two images of different time instant, similarly to texture misalignments between two views.

The purpose of this paper is to correct texture misalignments due to modeling errors. The proposed method, called *floating polygon soup* consists in morphing 3D geometry depending on the current viewpoint. Morphing the geometry enables to keep the same texture coordinates for each 3D point, on the contrary to warping 2D textures. The representation that we used to evaluate this method is the polygon soup [1]. The next section gives an overview of this representation.

3. POLYGON SOUP REPRESENTATION

3.1. Overview

We propose to construct, from the input MVD data, a representation called polygon soup (figures 1 and 2).

3D polygons. First of all, the rendering primitives are 3D polygons. Polygonal primitives have several advantages: they increase compactness by adapting their size to the surface, they model the continuity of the surface, and graphics processors are optimized to render them. Moreover, all the 3D polygons are not necessarily

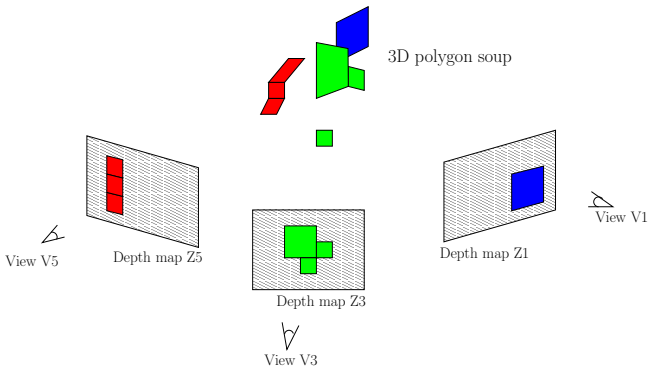


Fig. 1. Polygon soup. Each 3D polygon is defined by a 2D polygon (quad), and by the depth information at each corner of the quad.

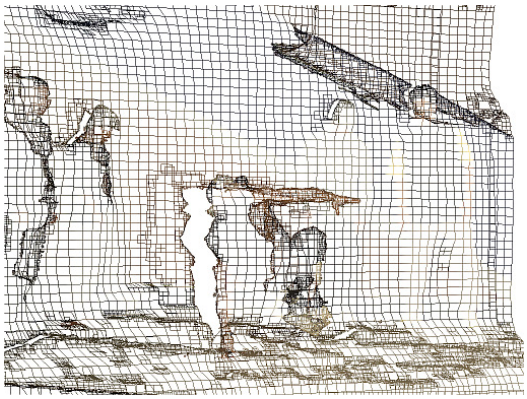
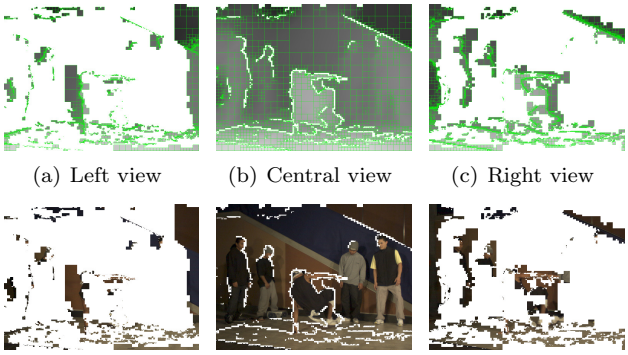


Fig. 2. Example of a polygon soup, seen from an arbitrary viewpoint (without texture).

connected to each others and can overlap, forming a kind of mixture of polygons which is often called a polygon soup. Here, the disconnection feature is very important. First, it ensures that foreground and background objects are not connected, thus preserving depth discontinuities. Second, it allows to easily remove redundant or unreliable polygons, thus increasing the compactness. The overlapping feature is also a key point. Since multi-texture and multi-geometry are available from the input data, overlapping polygons (i.e. coming from different views) can be selected or merged depending on the desired viewpoint, thus ensuring view-dependent quality of virtual views. In a word, disconnection and overlapping allow to play with the compactness and image quality tradeoff by removing unnecessary polygons and overlapping necessary ones. Figure 2 shows an example of such 3D polygons that can overlap and be disconnected.

Stored in 2D. The representation forms a 3D polygon soup at the view synthesis stage. However, for compactness and compression efficiency, the polygons are stored in 2D with depth values at each corners. These 2D polygons are extracted from the depth maps using a quadtree decomposition method. These 2D polygons are also called 'quads'. Figure 3 shows an example of 2D quads extracted from MVD data and for 3 views.



(a) Left view (b) Central view (c) Right view

Fig. 3. Example of polygon soup stored in 2D. Top line: final set of selected quads in each view. Bottom line: associated color information.

The decomposition of the depth maps into quadtree allows to retrieve the (x, y) positions of the quads, thus a compression method exploiting this structure can be employed. Each leaf of the tree corresponds to a quad. Using this structure, it is easy to remove redundant polygons by pruning the corresponding leaf in the tree. Moreover, the decomposition can be adapted to the geometry of the scene such that large quads approximate flat surfaces and small quads preserve geometric details and discontinuities.

Textured by the images. The texture of the polygons is given by the original images: the block of pixels in the image corresponding to a quad is mapped onto the 3D polygon using texture mapping techniques supported by graphics hardware. Since, multiple images are available, it is possible to texture the polygons with multiple textures depending on the view-point. This results in a multi-textured appearance of the polygon that better reproduces specular effects of real-world scenes. In addition, this kind of multi-view videos can be efficiently compressed using the standardized H.264/MVC compression method. Finally, transmitting the original views ensures maximum image quality at original view-points.

3.2. Processing steps

Using such a polygon soup involves several processing steps for constructing this representation, compressing it, and synthesizing virtual views. These different steps are listed:

- **Construction:** Depth estimation; Quadtree decomposition; Redundancies reduction.

- **Compression:** QuadTree-based compression or block-based compression
- **View-synthesis:** Polygon warping; Texture mapping; Post-processing; Quadtree decomposition (if block-based compression)

3.3. Properties

The proposed polygon soup can be analyzed in terms of general properties:

- Construction complexity: Most of the complexity is transfered to the construction of the representation in order to achieve compactness, and to decrease view synthesis complexity.
- Compactness: Compactness is achieved by keeping the number of polygons low and reducing inter-view redundancies.
- Compression compatibility: A new quadtree-based compression method has been introduced in [1] for the polygon soup. However, block-based compression is also possible providing that the quadtree decomposition step is transferred at the user side of the system.
- View synthesis complexity: Virtual views are synthesized using graphics hardware and polygonal primitives. Polygons avoid the apparition of sampling artifacts compared with depth-based view synthesis, thus the view synthesis complexity is lower. Moreover, since unreliable polygons are removed during the construction of the representation, the polygon soup reduces the so-called ghosting artifacts and thus no additional process are required to avoid them. Additional process like hole filling and filtering are still required for high image quality.
- Navigation range and image quality: Original views are preserved ensuring maximum quality at original viewpoints. Navigation in between original views is done by view-synthesis using the polygon soup as the geometry of the scene. However, because of modeling errors, texture misalignments appear in the synthesized views and reduce the final image quality. Therefore, the next section presents a new method to reduce these texture misalignments.

4. FLOATING POLYGON SOUP

This section presents a method to reduce modeling errors that create artifacts in the synthesized images. Starting from n views and associated polygon soup, the idea is to deform each polygon depending on the current viewpoint. We call this: *Floating polygon soup.*

The proposed method is divided into two steps: matching and morphing. The matching step consists in matching 3D geometries pairwise while using an image registration constraint. This matching step can be seen as a preprocess before the synthesis of arbitrary viewing positions. Then, the morphing step consists in interpolating the geometries between their matched correspondences according to the current viewing position. It thus realizes view-dependent geometry. View-dependent texture is also performed, as all views are projected into the current viewing position and blended to allow for smooth combination.

Figure 4 summarizes the floating geometry method for virtual view synthesis.

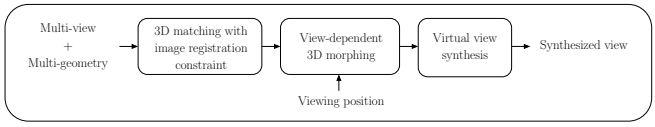


Fig. 4. Overview of the floating geometry method for virtual view synthesis.

We use the following notations:

- a view point V_i is defined as a pair (G_i, I_i), where G_i defines geometry (depth map, 3D model,..) and I_i is the acquired color image also used as a texture image.
- $\pi_i()$ and $\pi_{G_i}^{-1}()$ denote projection on view V_i and backprojection from view V_i using G_i.
- I_i^j (resp. G_i^j) denotes the image (resp. geometry) from view V_i projected in view V_j.
- $W_{I_1 \to I_2}$ is the motion flow between I_1 and I_2.

4.1. 3D matching with image registration constraint

We consider n original views $V_i = (G_i, I_i)_{i=1,n}$, where geometry is assumed already estimated for each acquired image, for instance as a 3D model or a depth map. The matching step consists in defining a point to point correspondence between any pair of geometries (G_i, G_j) based on image registration between I_i and I_j.

The process is illustrated on Figure 5. View V_i is first projected into view point V_j using geometry G_i to produce projected image I_i^j:

$$I_i^j = \pi_j(\pi_{G_i}^{-1}(I_i)) \quad (1)$$

As explained earlier, image I_i^j is not aligned with I_j. We thus estimate the flow field $W_{I_i^j \to I_j}$ compensating

for image misalignment in V_j. It is then used to warp I_i^j, and the resulting image is back-projected in 3D using geometry G_j to obtain the *floated geometry* G_i^j. That is :

$$G_i^j = \pi_{G_j}^{-1}(W_{I_i^j \to I_j} \circ I_i^j) \quad (2)$$

G_i^j is consistent with G_j, i.e. it provides same geometry information, but each point in G_i^j is associated with a point in G_i thanks to this definition. Moreover, the projection of V_i into V_j using G_i^j is aligned with I_j, i.e. it performs same image registration as applying optical flow on I_i^j. Thus when synthesizing view V_j, 3D vertex P_i with texture p_i is floated to 3D vertex P_i^j and then projected into V_j on point p_j. As P_i and P_i^j have been matched in order to represent the same physical 3D point, texture blending using texture points p_i and p_j will not generate misalignments in the synthesized view V_j.

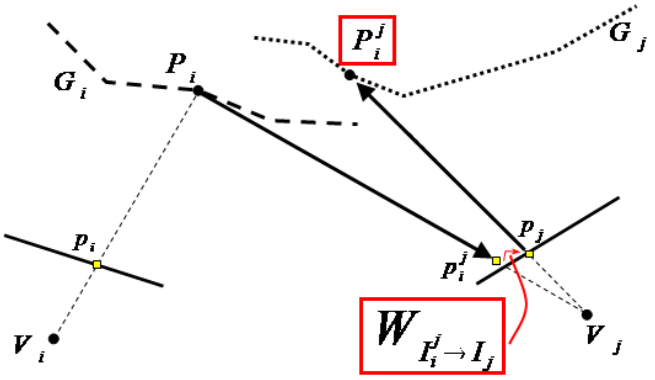


Fig. 5. Computation of a matching vertex. P_i is projected in V_j giving p_i^j which is matched with corresponding pixel p_j. Finally, p_j is back-projected to the approximate geometry resulting in the 3D point P_i^j.

4.2. View-dependent 3D morphing.

Once matched geometry G_i^j has been computed for all pairs of views in a pre-processing step, then the geometry can be floated for any virtual view during the view synthesis stage. The key point here is to ensure texture alignment by keeping the geometries consistent with each other. An intermediate geometry G_i^v that depends on the position of the virtual view is thus computed. To do so, let V_l and V_r be the two original viewpoints closest to the virtual view V_v (called "left" and "right" views for simplicity). Let V_i be the current view to be projected into V_v. The floating geometry for each vertex in V_i has already been computed for left view V_l and right view V_r, namely G_i^l and G_i^r. Then the computation of G_i^v consists in interpolating the geometry between G_i^l and G_i^r, vertex by vertex:

$$G_i^v = \omega_l G_i^l + \omega_r G_i^r \quad (3)$$

where ω is a weight associated to the position of the current viewpoint. It can be defined using interpolation method between two original cameras (SLERP) [21] or by angular weighting scheme [16, 15]. A smooth deformation of the geometry is obtained as the virtual view moves from one original view to another. Figure 6 illustrates this view synthesis step using the same example as in the previous figure. The view being projected is V_i and the left and right views around the virtual view are $V_l = V_i$ and $V_r = V_j$. The 3D vertex P_i is floated onto an intermediate position $P_i^v = \omega_i P_i + \omega_j P_i^j$. Finally, P_i^v is projected into the virtual view, giving the pixel p_i^v.

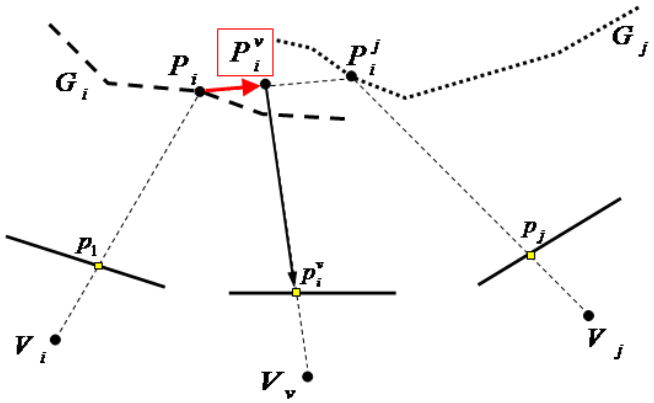


Fig. 6. Virtual view synthesis with floating geometry. Vertex P_i is floated to an intermediate position P_i^v between P_i and P_i^j, then it is projected to the virtual view giving pixel p_i^v.

This process is repeated for each view and associated geometry as illustrated in figure 7 where G_i and G_j are floated to G_i^v and G_j^v respectively. As a result, floated geometries are consistent with each other: they only differ by a change of vertices (remeshing) and texture misalignments are reduced both when synthesizing original views and virtual views.

5. EXPERIMENTS

In our previous work [1], the polygon soup has been compared to MVC-H264 (intra-mode) for coding depth information, showing that rate/distorsion performances are competitive on tested data. In this section we evaluate the performances of the floating method with regard to the quality of the synthesized images. Two tests have been done on views V_1, V_3 and V_5 of *Breakdancers* and *Ballet* sequences[1]. The first test operates on the

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

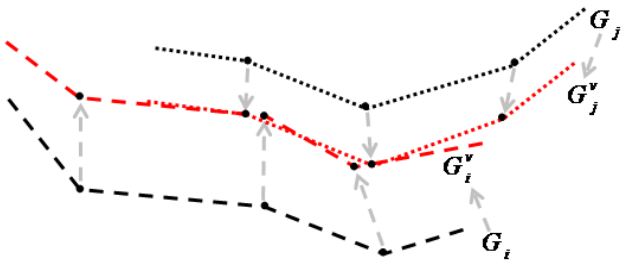


Fig. 7. Virtual view synthesis with floating geometry. G_i and G_j are floated to G_i^v and G_j^v respectively. As a result, the geometries are consistent with each other, and texture misalignments are reduced.

full polygon soup (i.e. before the redundancy reduction step), and the second test is performed on the reduced polygon soup. The quality of the synthesized view was evaluated with and without application of the floating polygon soup method. For objective evaluation purpose, the novel view is located on original camera viewpoint V_2 and V_4 such that an error metric is used for comparison. The Peak-Signal-To-Noise PSNR metric distortion is used here (log function of inverse MSE). Motion estimation was performed using Urvoy et al. [22] algorithm. This motion estimator uses variable size block matching with some regularization.

Figure 8 shows typical misalignments artifacts observed when using the full polygon soup without floating geometry. As expected, blurring and splitting artifacts appear due to texture misalignments. In (a), a blur artifact is shown: the boundary of the hand is blurred. In (c), a splitting artifact is shown: the white line is split in two instead of being one. After applying the floating method, we can observe that these artifacts are reduced. The hand appears sharper and the white line corrected.

The second experiment consists in applying the floating method on the reduced polygon soup. It can thus be performed after transmission, at the user side of the system. Figure 9 shows the evaluation results. Subfigures (a),(b) and (c) give respectively the synthesized view without floating geometry, the original view, and the synthesized view with floating geometry. Subfigures (d), (e) give the images of difference between synthesized views and the original ones. The darker a pixel, the higher the error. Artifacts due to texture misalignments can be observed in the view synthesized without floating geometry (subfigures (a) and (d)): we can see in (d) that errors are visible mainly at texture edges. Results of the floating geometry approach can be seen in (c) and (e). We can see that texture misalignments around edges have been reduced. In (c), the deformation of the face (right cheek of the character) has been corrected.

Tables 1 and 2 give the PSNR values obtained for

Breakdancers	no floating geometry	floating geometry	gain
virtual view V_2	35.29 dB	36.51 dB	+1.22 dB
virtual view V_4	34.65 dB	36.24 dB	+1.59 dB

Table 1. Comparison of PSNR values without and with floating geometry using three views and reduced polygon soup.

Ballet	no floating geometry	floating geometry	gain
virtual view V_2	33.71 dB	35.00 dB	+1.29 dB
virtual view V_4	35.27 dB	36.80 dB	+1.53 dB

Table 2. Comparison of PSNR values without and with floating geometry using three views and reduced polygon soup.

virtual views V_2 and V_4 and sequences *Breakdancers* and *Ballet*. For *Breakdancers*, the PSNR measure has increased by 1.22 dB (from 35.29 dB to 36.51 dB) for virtual view V_2, and by 1.59 dB (from 34.65 dB to 36.24 dB) for virtual view V_4. Similarly, for *Ballet* sequence, the PSNR measure has increased by 1.29 dB (from 33.71 dB to 35.00 dB) for virtual view V_2, and by 1.53 dB (from 35.27 dB to 36.80 dB) for virtual view V_4.

One can notice that the earing of the character in the original view is not reproduced in the synthesized views. This is a non-Lambertian reflective effect that is not visible in either of the views used for the synthesis. Therefore, floating geometry cannot compensate such an effect.

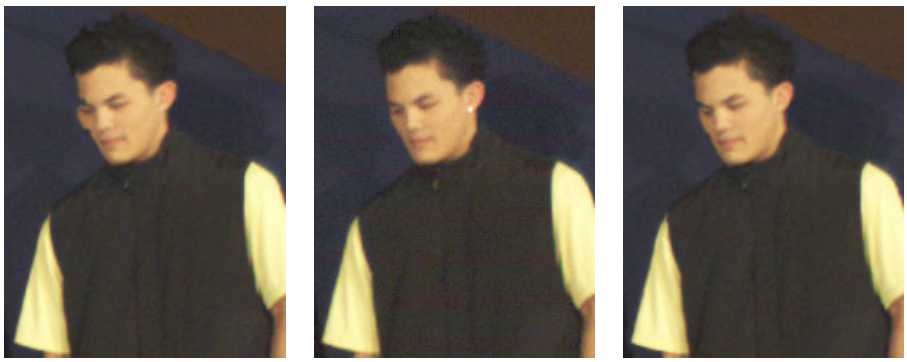
The matching step of the floating method could be computed either at the acquisition side of the system or at the user side. In the first scenario, the additional information involved by the matching should be transmitted and therefore increasing the data load. In the second scenario (i.e. at the user side), the synthesis complexity is increased instead of the data load.

6. CONCLUSION

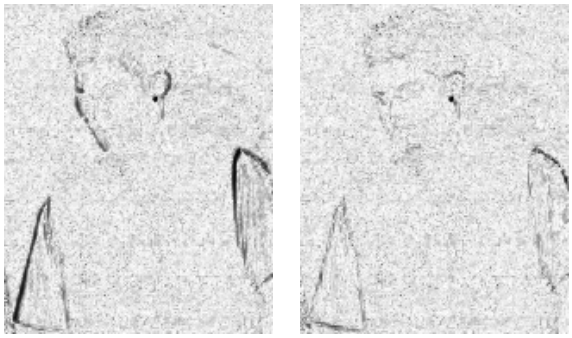
This paper has presented a representation for multiview videos. It extends the polygon soup representation previously proposed by reducing texture misalignment due to modeling errors. The method is called floating polygon soup. It consists in floating the geometry depending on

(a) Blur artifact (b) Result (c) Splitting artifact (d) Result

Fig. 8. Examples of artifacts observed with the full polygon soup: Blur (a) and splitting (c). Results after applying floating polygon soup (b) and (d).



(a) Before (35.29 dB) (b) I_2 Original view (c) After (36.51 dB)



(d) Difference (a-b) (e) Difference (c-b)

Fig. 9. Comparison of a synthesized virtual view without and with floating geometry.

the current viewing position such that texture misalignments are reduced. To do so, two steps are computed: matching and morphing. As a result misalignments are reduced and objective quality is increased. The main limitation of this method comes from its dependency on the quality of motion estimation (e.g. when high misalignments are present in the image).

In the future, a method for coding the additional floating geometry information will be studied, and an acceleration of the method will be tested using graphics hardware capabilities. Moreover, the floating geometry can be extended to the temporal domain and used for temporal prediction and coding.

7. REFERENCES

[1] T. Colleu, S. Pateux, L. Morin, and C. Labit, "A polygon soup representation for multiview coding," *Journal of Visual Communication and Image representation (JVCI). Special Issue on multi-camera imaging.*, 2009.

[2] O. Schreer, P. Kauff, and T. Sikora, *3D Videocommunication: Algorithms, concepts and real-time systems in human centred communication*, Book, John Wiley & Sons, 2005.

[3] Marcus A. Magnor, *Video-Based Rendering*, AK Peters Ltd, 2005.

[4] Minh N. Do, Chang-Su Kim, Karsten Müller, Masayuki Tanimoto, and Anthony Vetro, "Multicamera imaging, coding and innovative display: techniques and systems," *Journal of Visual Communication and Image Representation*, vol. 21, no. 5-6, pp. 375 – 376, 2010, Special issue on Multicamera Imaging, Coding and Innovative Display.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[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] M. Maitre and M.Do, "Shape-adaptive wavelet encoding of depth maps," in *Picture Coding Symposium, Chicago, US*, 2009.

[12] S. Yea and A. Vetro, "Multi-layered coding of depth for virtual view synthesis," in *Picture Coding Symposium, Chicago, US*, 2009.

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

[14] A. Bornik, K. Karner, J. Bauer, F. Leberl, and H. Mayer, "High-quality texture reconstruction from multiple views," *The journal of visualization and computer animation*, vol. 12, no. 5, pp. 263–276, 2001.

[15] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik, "Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach," in *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 1996, pp. 11–20, ACM.

[16] C. Buehler, M. Bosse, L. McMillan, S.J. Gortler, and M.F. Cohen, "Unstructured lumigraph rendering," in *SIGGRAPH 2001, Computer Graphics Proceedings*, 2001, pp. 425–432.

[17] M. Eisemann, B. De Decker, A. Sellent, M. Magnor, E. de Aguiar, N. Ahmed, P. Bekaert, and H. Seidel, "Floating textures," *Computer Graphics Forum (Proc. Eurographics EG'08)*, vol. 27, no. 2, 4 2008.

[18] T. Takai, A. Hilton, and T. Matsuyama, "Harmonised texture mapping," in *3DPVT*, Paris, France, May 2010.

[19] Hisayoshi Furihata, Tomohiro Yendo, Mehrdad Panahpour Tehrani, Toshiaki Fujii, and Masayuki Tanimoto, "Novel view synthesis with residual error feedback for ftv," in *Stereoscopic Displays and Applications XXI*, Andrew J. Woods, Nicolas S. Holliman, and Neil A. Dodgson, Eds., San Jose, California, USA, 2010, vol. 7524, SPIE.

[20] Sundar Vedula, Simon Baker, and Takeo Kanade, "Image-based spatio-temporal modeling and view interpolation of dynamic events," *ACM Trans. Graph.*, vol. 24, no. 2, pp. 240–261, 2005.

[21] Ken Shoemake, "Animating rotation with quaternion curves," in *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 1985, SIGGRAPH '85, pp. 245–254, ACM.

[22] Matthieu Urvoy, Nathalie Cammas, Stéphane Pateux, Olivier Déforges, Marie Babel, and Muriel Pressigout, "Motion tubes for the representation of images sequences," in *Proceedings of ICME'09 IEEE International Conference on Multimedia and Expo*, Cancun Mexico, 07 2009, pp. 1–4.