GRAPH-BASED REPRESENTATION AND CODING OF MULTIVIEW GEOMETRY

Thomas Maugey[1], *Antonio Ortega*[2], *Pascal Frossard*[1]

[1]Ecole Polytechnique Fédérale de Lausanne (EPFL) [2]University of Southern California

ABSTRACT

We propose a new approach for describing the geometry information of multiview image representations. Rather than transmitting the raw geometry of the scene, under the form of depth information, we build a graph that represents the connections between corresponding pixels in different views in a multiview image set. The graph starts with the reference image and recursively represents the next levels (*i.e.,* images) by storing the new pixels (those that cannot be derived from the previous image) and their connections to the lower level. The decoder uses these connections to recover the multiple images. In addition to being natural and more easily controlled, the proposed graph-based representation can be compressed more efficiently than depth images. This new representation offers promising perspectives for effective and flexible coding in multiview imaging.

Index Terms — Graph representation, multiview image coding, depth maps

1. INTRODUCTION

Systems for transmission of three-dimensional images may use different types of geometrical information in the 3D data processing representation. This geometrical information determines the construction of the whole transmission chain, from capture to view rendering. It is transmitted along with the texture data and used for encoding techniques (to exploit correlation between pixels) or to render virtual viewpoints [1] at the decoder. This geometrical information has a coding cost that naturally depends on its level of precision. An accurate geometry signal leads to significant improvements of the texture coding efficiency and rendering quality but also has higher coding costs. In other words, systems have to properly choose the amount of geometrical information that they include in the data representation. This trade-off between accuracy and coding efficiency is usually solved when fixing the type of data representation method [2].

The representation of 3D images has taken different forms in the literature. First, some systems consider the classical image-based representation, such as multiview images sets. For those approaches, the correlation between the views is described with disparity vectors [3] that indicate color similarities between blocks of two neighboring images. The level of geometrical information is typically low (block precision), which leads to suboptimal performance when the views are too distant. Some more advanced image-based representation methods handle this problem by concatenating a large number of images captured from close viewpoints. This is the ray-space or the light field representation [4]. In that case, the geometrical information is hidden in the concatenation of equidistant images. Whereas coding and interpolation techniques can efficiently exploit this information, such a representation requires very specific acquisition conditions, which are not always practical. Then, other methods explicitly use geometrical information such as depth images [5] that describe the distance between the scene and the focal plane of every camera. Depth images are very powerful tools that lead to many efficient techniques for correlation extraction and view synthesis. This representation, however, has one main limitation. The link between coded depth quality and view rendering efficiency is not explicit. More precisely, the distortion of the depth image does not describe exactly the induced error in the virtual viewpoint generated with this lossy depth. This problem has been addressed in the literature, mostly by introducing better metrics [6, 7], but the impact of geometric approximations during the compression is still non-straightforward and leads to unpredictable and undesirable effects. Finally, there are approaches that use even more geometrical information such as mesh-based representations [8]. Although they explore interesting ideas that merge computer vision and image processing tools, they still remain inefficient compared to the previously mentioned approaches in terms of coding performance.

The choice of the proper level of geometry in the 3D data representation is not totally clear from the literature. Since time-of-flight sensors are getting more and more accurate [9], depth seems to be the most popular and promising technique. Nonetheless, this representation does not make explicit the relation between geometry precision and view synthesis accuracy. In this work, we propose a more natural geometry information that is of moderate size, but leads to effective view reconstruction algorithms. After observing that the knowledge of the scene geometry leads to connections between pixels of different viewpoints, we propose to directly represent these links with a graph. The graph contains all the geometrical information needed for multiview image reconstruction at the decoder side. The advantage of such an approach is that it works directly with the inter-pixel connections and offers a better control of the compression artifacts. We have compared this approach with depth map representations in terms of compression performance with promising results (rate savings can achieve 33%). The proposed graph-based representation is also flexible and enables rate scaling, as less information has to be sent when fewer views need to be generated at the receiver.

2. GRAPH-BASED GEOMETRY REPRESENTATION

2.1. Scene capture assumptions

Let us consider a scene captured by N cameras with the same resolution and focal length f. The n-th image is denoted by I_n, with $1 \leq n \leq N$, where $I_n(r, c)$ is the pixel at row r and column

This work has been partially supported by the Swiss National Science Foundation under grant 200021-126894.

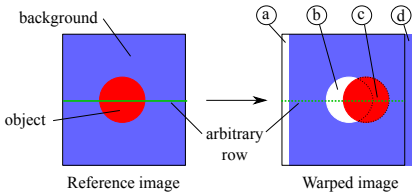


Figure 1. Type of artifacts happening when depth-based image warping: pixels can be a) appearing, b) disoccluded, c) occluded and d) disappearing. Green plain line is an arbitrary row in the reference image and the dashed line is the corresponding one in target image.

c. We only consider translation between cameras, and we assume that the views are rectified. In other words, the geometrical correlation between the views I_n is only horizontal. We assume that accurate depth images Z_n for every viewpoints I_n are available at the encoder. From the rectification assumption, the link between depth z and disparity d between two camera images is given by $d = \frac{f\delta}{z}$, where δ is the distance between the two cameras. In the following, we mainly deal with disparity values, which we compute from the depth maps Z_n and the camera parameters. Our goal is to design the best multiview representation of these N camera images. In the following, the *synthesized* images are the $N-1$ images generated using the *reference* image and additional structure and color information introduced below. We do not consider virtual view synthesis in this paper.

2.2. Geometrical structure representation

Before introducing our graph representation in detail, we analyze the effect of camera translation on the image content. Let us consider two images I_n and I_{n+1} captured by cameras that are separated by a distance δ. Since we consider only full pixel displacements, the geometrical correlation between these two images takes the form of $I_{n+1}(r,c) = I_n(r,c+d)$, where d is a disparity value (derived from Z_n, which is supposed to be available) and I_n is arbitrarily chosen as the reference image. The pixels for which this relation holds belong to the regions in image I_{n+1} that can be directly associated to regions in I_n. They correspond to the elements of the scene that are visible in both images. The elements that are visible only from one viewpoint are usually designed under the general name of occlusions, even if their appearance is not only due to object occlusions. Specifically, we can categorize these missing pixels into four different types as illustrated in Fig. 1. Because of camera translation, a new part of the scene appears in the camera. It usually comes from the right or left (depending on translation direction) and these pixels are not related to object occlusions. They are called *appearing* pixels (a). During camera translation, foreground objects move faster than the background. As a result, some background pixels may appear behind objects and are thus called *disoccluded* pixels (b). Conversely, some background pixels may become hidden by a foreground object. These are called the *occluded* pixels (c). While some pixels appear in the camera frame because of the viewpoint translation some pixels disappear, *disappearing* pixels (d), from the frame.

If we consider a row of the target image in Fig. 1 (dashed green horizontal line), we notice that it is made of these different types of pixels. More precisely, if we start from the left border, the row first contains several appearing pixels, and then some pixels of the reference image. Then, the row presents some disoccluded pixels before coming back to reference image pixels. After that, the row meets occluded pixels that correspond to a jump in the reference image. The rest of the row refers to the reference image until the disappearing pixels. If we now assume that we want to describe pixels in this target row referring to a maximum number of elements from the corresponding row in the reference frame (plain green horizontal line), we clearly see that it could be done by navigating between the reference image and the "new" pixels of the target image. This navigation can be guided by connections. The graph that we propose to construct is made of these connections. This graph is derived from depth information and is sent instead of depth to the decoder. The number of connections depends linearly on the number of foreground objects in the image. Similarly, the size of these connections evolves linearly with the separation between cameras and object disparities. A more formal graph construction method is given in the next section.

2.3. Graph construction

The proposed graph representation intends to avoid redundancies in the color description (*i.e.*, only "new" pixels are described) and additionnally to present more intuitive geometry information with links between corresponding pixels in different views. We show in Fig. 2 a simple graph construction example, with 5 levels (1 reference and 4 synthesized images). More generally, a graph with N levels describes 1 reference image and $N-1$ synthesized ones. Its construction requires the depth maps Z_n, $1 \leq n \leq N-1$. Since the object displacement is only horizontal, we consider independent graph construction for each row of the images (however coding can eventually be done across lines). Such a graph is made of two components, which are described by two matrices of size $L \times W$, where L is the number of levels (*i.e.,* the number of images encoded by the graph) and W is the image width. These two matrices are the color values $\Gamma_r = [\gamma^r_{i,j}]_{i\leq L, j\leq W}$ and the connections $\Lambda_r = [\lambda^r_{i,j}]_{i\leq L, j\leq W}$ and represent color and geometry information for all pixels of all images. They are both initialized to 0, which means "no connection" and "no color value". For the sake of clarity, let us first describe in detail a 2-level graph construction of an arbitrary row r by considering only one synthesized frame I_2, one reference frame I_1 and its associated depth map Z_1. Refer to Fig. 2. The first level corresponds to the reference level, and thus $\gamma^r_{1,j} = I_1(r,j)$ for all $j \leq W$. Then, the connections always indicate the links between the current level and the next one. Hence, since we are considering only two levels, we have $\lambda^r_{2,j} = 0$ for all $j \leq W$ in our example.

Then, the $\lambda^r_{1,j}$ connections and $\gamma^r_{2,j}$ color values are constructed under the following principles. The pixels intensities are represented in the level where they appear first. For this 2-level example, it means that the second level only contains pixels that were not present in the reference image. The connexions $\lambda^r_{1,j}$ simply consists in linking these "new" pixels to the position of their neighbor in the previous level. We describe now more precisely how each of the pixel types in Fig. 1 is handled in our graph-based representation. First, the appearing pixels are simply given in the corresponding $\gamma^r_{2,j}$ values, without connectivity information (they are implicitly attached to the side of the image). In the example of Fig. 2, we see that the dark blue appearing pixel (a) is stored in level 2 at its position in I_2, *i.e.,* in $\gamma^r_{2,1}$. For the disoccluded pixels, since they do not appear in the reference image, their color value is stored in the color matrix $[\gamma^r_{2,j}]$, where j correspond to their positions in I_2. In Fig. 2, the disoccluded pixels (b) are stored in $\gamma^r_{2,3}$ and $\gamma^r_{2,4}$. Additionally, at reference level and at the position

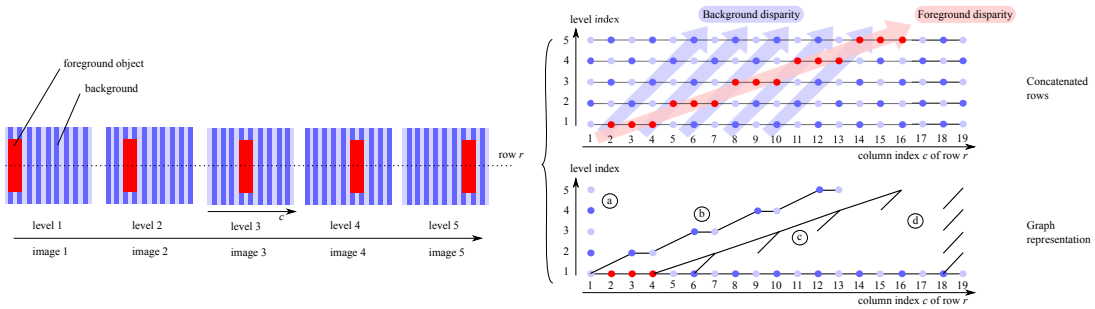


Figure 2. Toy graph construction example: blue texture background has a disparity of 1 at each level and red rectangle foreground a disparity of 3 for each level. Graph contains all different types of pixels: a) appearing, b) disoccluded, c) occluded and d) disappearing.

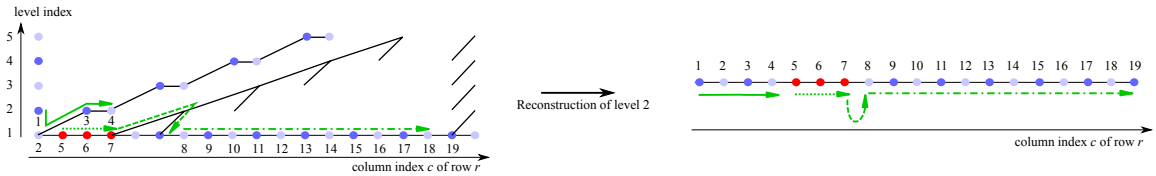


Figure 3. Reconstruction of the level 2 with the toy example of Fig. 2. Green arrows indicate the graph exploration order for view reconstruction.

c of the last pixel before the foreground object on row r, we store the value $\lambda^r_{1,c} = d + 1$, where d is the disparity vector associated to depth value $Z_1(r, c)$. This connection value links the last background pixel of the reference image to the ones in the target level that are disoccluded. For example, in Fig. 2, the foreground object is red. In level 1, the last pixel before this foreground object is at position 1 (light blue pixel). The graph thus links this pixel to the first disoccluded pixel (b) of level 2. The disparity d of the background of example in Fig. 2 is equal to 1, so the connection value stored is equal to 2. In other words, we have $\lambda^r_{1,1} = 2$.

As mentioned in the previous section, the occluded pixels (c) correspond to a jump in the reference image. This jump is stored in the connectivity vector at the following positions: i) the last pixel of the foreground object (connection equal to the foreground disparity) and ii) the last pixel of the corresponding occluded region (connection equal to the background disparity). In Fig. 2, the last pixel of the foreground object i) is at position 4 in level 1. Thus, we have $\lambda^r_{1,4} = 3$, since the red foreground object has a disparity of 3. Secondly, the last pixel of the occluded region ii) is at position 6 in level 1. Since the background disparity is 1, we have $\lambda^r_{1,6} = 1$. We notice that the two connections meet in $c = 7$ in level 2, which corresponds to the position of the last foreground pixel in level 2. This time, since no new pixel is contained in the target frame, we do not store any value in the color vector. Finally, the disappearing pixels (d) are simply indicated by a connection value at the last pixel before them. The connection value is equal to the background disparity. In Fig. 2, the first disappearing pixel is at position 19, thus $\lambda^r_{1,18} = 1$.

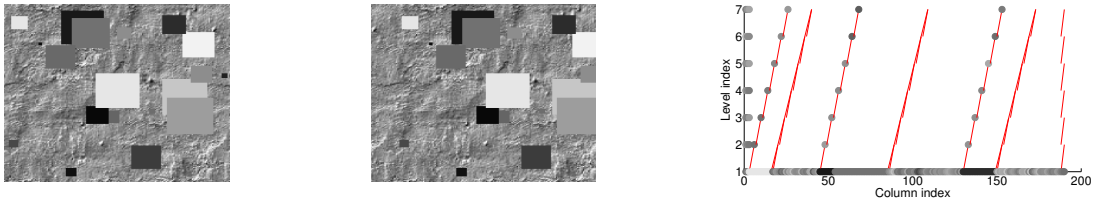
With this graph construction method and looking at the example of Fig. 2, we see that the graph representation is sparse and avoids all redundancy in the color value description since the pixels stored in level 2 of Γ_r are only those that are not present in the reference image. Another important advantage of this graph representation consists in the multi-level structure. The graph construction is done by recurrence (meaning that each level refers to the lower one), and we see in Fig. 2 that interesting properties appear in the graph structure. The connections in one level correspond to connections in other level and can be seen as chains of connections. Thus, intuitively, a reconstruction algorithm only needs to go through these connection chains to synthesize the different multiview images, as discussed below.

2.4. View synthesis at the decoder

The graph information can be used directly for view synthesis at decoder. Assume that the two graph components Γ_r and Λ_r are available at the decoder for every row r. The reconstruction of a certain level requires the color values and the connections of all lower levels. The filling of the current level color values is performed by navigating in the graph between the different levels. This navigation starts from the border of the image at the level that needs to be constructed, then follows the connections and refers to the lower levels when no color information is available at current level. We show in Fig. 3 an example of a view synthesis for the image of level 2, based on the graph of Fig. 2. In the following, pixel numbering is done with respect to the column index of I_2, as done in Fig. 2. The reconstruction starts with the appearing pixel 1 at level 2 . Then, it moves to the reference level and fills pixel color until meeting a non-zero connection. In the case of Fig. 3, the first connection is after pixel 2 and links it to pixel 3 and 4 in level 2. After filling all the disoccluded pixels, the reconstruction goes back to the reference level and fills color information (5, 6 and 7) until the next non-zero connection (at pixel 7). The connection in 7 indicates an occluded region. Hence, the reconstruction algorithms jumps in the reference frame and restart the filling (pixel 8 to 19) until the next non-zero connection (disappearing pixel). The reconstruction of the other levels is done recursively. We see that the reconstruction process is very simple, fluid and controllable (contrarily to depth-based representations).

3. EXPERIMENTS

In order to evaluate our proposed approach, we compare its lossless compression rate to that required for encoding a depth map. As we can observe in the example of Fig. 2, the matrix Λ_r is



(a) (b) (c)

Figure 4. Example of two images (from view 1 (a) and 4 (b)) and the associated graph-based representation of row 20 with 7 levels (c)

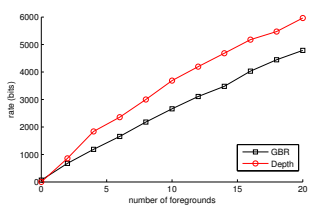


Figure 5. "Structure rate" as a function of the number N_f of foreground objects (averaged over different number of levels $1 \leq N_l \leq 6$)

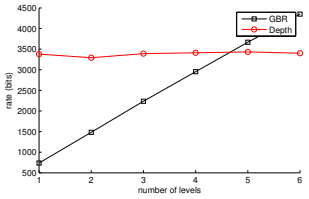


Figure 6. "Structure rate" as a function of the number N_l of virtual views (averaged over different number of foreground objects $1 \leq N_f \leq 20$)

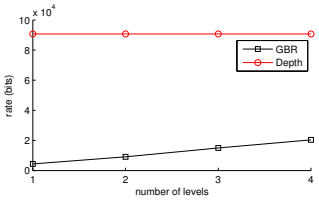


Figure 7. "Structure rate" comparison for natural images (*sawtooth*) in function of the number of views N_l.

sparse. Hence, it can be coded with a small number of bits. For that purpose, we do not code directly Λ_r and rather consider a smaller matrix Φ of size $M \times 4$, where M is the number of non-zero elements in all the Λ_r with $r < H$ (H is the height of the image). The matrix Φ stores all the meaningful connections and it is organized as follows. The first column of Φ contains this row indices r. The second column contains the column indices c, the third column contains the level indices, and finally, the fourth column contains the connection values. Then, we simply consider an arithmetic coding of every column, with, for some of them, a differential operation as preprocessing, in order to decrease the entropy. Alternatively, the depth maps are processed through a differential operator and then compressed with an arithmetic coder.

We first consider synthetic images made of one background and N_f foregrounds objects (all parallel with the camera plane). Every foreground object is a square and has a different depth. We synthesize N_l viewpoints of equidistant cameras. The images of this database are generated randomly. In other words, given a number N_f of foreground objects, the algorithm places them randomly at a random depth value. In order to avoid particular cases, we estimate each rate value presented below by averaging the compression rate of 10 random images as explained above. Fig. 4 shows two images of such synthetic images. The corresponding graph is also shown in Fig. 4(c). We see that, even if the number of connection increases with the number of foreground objects, the graph still remains sparse. We verify this statement by calculating the structure rate, *i.e.*, the depth and graph coding cost, for different number of foreground objects (the results are averaged over a number of levels N_l varying from 1 to 6). Results are shown in Fig. 5. Although the structure rate increases with the number of foregrounds, the graph-based representation is compressed more efficiently than the depth images.

The advantage of depth-based representation relies in the fact that a single depth image is able to generate different viewpoints (as soon as the camera positions are known). In the contrary, the proposed graph-based representation stores more information when the number of level increases. This is why we compare the two approaches performance for different number of levels. Results are shown in Fig. 6. We see that the graph-based representation requires much less rate than depth maps until $N_l = 4$. After that, the graph structure becomes heavier. However, the 6^{th} viewpoint is distant from the reference image, and depth-based representation would require additional data (such as another depth image) to estimate the object displacement in the occlusions, contrary to our graph-based representation, which contains all the information needed for viewpoint synthesis at the receiver. We also test our method on a natural dataset from Middlebury University, *sawtooth* [10], made of 5 rectified images. We show our results in Fig. 7 and we observe that similar promising performance are obtained.

We see that the concatenation of the different levels is made without redundancies (*i.e.,* no pixel is represented several times) nor constraints on camera position (*i.e.,* in contrary to light field methods, the graph-based representation does not impose close equidistant cameras). Finally, note that we could reduce the number of bits by using lossy compression. Depth compression has however to be performed carefully in order to control the quality of view synthesis. Our ongoing work [11] shows that GBR further leads to better lossy coding performance than depth-based schemes, in particular because the graph better controls the errors induced by compression.

4. CONCLUSION

In this paper, we propose a new form of geometry representation, which consists in coding the pixel connections in multiview images with a graph. First, it brings a more natural and controllable geometry structure than classical depth-based representations, which is appropriate for interactive navigation systems. Secondly, the sparse characteristics of the proposed approach lead to very efficient compressibility of this geometrical information. Future work will focus on the lossy compression of this promising representation.

5. REFERENCES

[1] D. Tian, P. Lai, P. Lopez, and C Gomila, "View synthesis techniques for 3d video," *Proc. of SPIE, the Int. Soc. for Optical Engineering*, vol. 7443, 2009.

[2] A. Alatan, Y Yemez, U Güdükbay, X. Zabulis, K Müller, CE Erdem, C. Weigel, and A. Smolic, "Scene representation technologies for 3dtv - a survey," *IEEE Trans. on Circ. and Syst. for Video Technology*, vol. 17, pp. 1587–1605, 2007.

[3] A Vetro, T. Wiegand, and G. Sullivan, "Overview of the stereo and multiview video coding extensions of the H.264/MPEG-4 avc standards," *Proc. IEEE*, vol. 99, no. 4, pp. 626–642, Apr. 2011.

[4] C. Zhang and T. Chen, "A survey on image-based renderingrepresentation, sampling and compression," *EURASIP J. on Sign. Proc.: Image Commun.*, vol. 19, pp. 1–28, 2004.

[5] K. Müller, P. Merkle, and T. Wiegand, "3d video representation using depth maps," *Proc. IEEE*, vol. 99, no. 4, pp. 643–656, Apr. 2011.

[6] WS Kim, A. Ortega, P. Lai, Tian D, and C Gomila, "Depth map coding with distortion estimation of rendered views," in *Proc. of SPIE, the Int. Soc. for Optical Engineering*, 2010.

[7] B. Rajei, T. Maugey, and P. Frossard, "Rate-distortion analysis of multiview coding in a DIBR framework," *submitted to Annals of Telecommunications*, 2012.

[8] N. Stefanoski, P. Klie, X Liu, , and J. Ostermann, "Layered coding of time-consistent dynamic 3-d meshes using a nonlinear predictor," in *Proc. IEEE Int. Conf. on Image Processing*, San Antonio, TX, US, 2007.

[9] G. Alenya and C. Torras, "Lock-in time-of-flight (tof) cameras: A survey," *IEEE Sensors Journal*, vol. 11, pp. 1917–1926, 2011.

[10] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Computer Vision*, vol. 47, pp. 7 – 42, 2002.

[11] T. Maugey, A. Ortega, and P. Frossard, "Multiview image coding using graph-based approach," in *submitted to IEEE Workshop on 3D Image/Video Technologies and Applications (IVMSP)*, Seoul, Korea, Jun. 2013.