Graph-Based Representation for Multiview Image Geometry

Thomas Maugey, *Member, IEEE*, Antonio Ortega, *Fellow Member, IEEE*,
and Pascal Frossard, *Senior Member, IEEE*

***Abstract*— In this paper, we propose a new geometry representation method for multiview image sets. Our approach relies on graphs to describe the multiview geometry information in a compact and controllable way. The links of the graph connect pixels in different images and describe the proximity between pixels in 3D space. These connections are dependent on the geometry of the scene and provide the right amount of information that is necessary for coding and reconstructing multiple views. Our multiview image representation is very compact and adapts the transmitted geometry information as a function of the complexity of the prediction performed at the decoder side. To achieve this, our graph-based representation (GBR) carefully selects the amount of geometry information needed before coding. This is in contrast with depth coding, which directly compresses with losses the original geometry signal, thus making it difficult to quantify the impact of coding errors on geometry-based interpolation. We present the principles of this GBR and we build an efficient coding algorithm to represent it. We compare our GBR approach to classical depth compression methods and compare their respective view synthesis qualities as a function of the compactness of the geometry description. We show that GBR can achieve significant gains in geometry coding rate over depth-based schemes operating at similar quality. Experimental results demonstrate the potential of this new representation.**

***Index Terms*— Multiview image coding, 3D representation, view prediction, graph-based representation.**

I. Introduction

MULTIVIEW image processing has received considerable attention in recent years. In particular, hardware technologies for the capture and the rendering of multiview content have improved significantly. For example, depth sensors and auto-stereoscopic displays have become popular recently [1]. This has led to novel immersive applications and thus to more challenges for the research community. One of the main open questions in multiview data processing is the design of representation methods for multiview data [2]–[4], where the challenge is to describe the scene content in a compact form that is robust to lossy data compression. Many approaches have been studied in the literature, such as the multiview format [5], light fields [6] or even mesh-based techniques [7]. All these representations contain two types of data: i) the color or luminance information, which is classically described by 2D images; ii) the geometry information that describes the scene's 3D characteristics, represented by 3D coordinates, depth maps or disparity vectors.[1] Effective representation, coding and processing of multiview data partly rely on the proper manipulation of the geometry information.

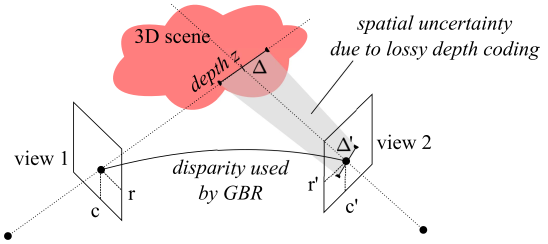


Fig. 1. Pixel (r, c) in view 1 is associated to pixel (r', c') in view 2, given its geometry (depth value z). An uncertainty Δ about this pixel's depth leads to a spatial inaccuracy Δ' in view 2. This basic observation is the origin of the main drawbacks of depth-based representations. In contrast, our GBR uses disparity values built with the "just enough" precision to guarantee a good rendering quality.

The multiview plus depth (MVD) [8] format has become very popular in recent years for 3D data representation and coding. Depth information can be used to build a reliable estimation of scene geometry, enabling encoders to extract the correlations between views [9] and decoders to synthesize virtual views [10]. Many recent multiview video coders rely on depth signals to enhance their coding performance [11]. However, the representation of geometry with depth maps has one main drawback: if lossy compression is applied to depth images, as done in classical coders, the resulting error affects the quality of synthesized images. This is the case even if the depth gives a good estimation of the 3D scene geometry. More specifically, an error Δ in the depth value for a first viewpoint (due to quantization for example) leads to a spatial error Δ' when determining the position of the corresponding pixels in neighboring views. This is illustrated in Fig 1. This error would not be a problem if it were possible to model

Manuscript received September 24, 2014; revised January 30, 2015; accepted January 31, 2015. Date of publication February 6, 2015; date of current version March 13, 2015. This work was supported in part by the Swiss National Science Foundation under Grant 200021 and Grant 126894 and in part by the Hasler Foundation through the Project entitled Novel Image Representation for Future Interactive Multiview Systems. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jie Liang.

T. Maugey is with INRIA/IRISA, Rennes 35042, France (e-mail: thomas.maugey@inria.fr).

P. Frossard is with the Signal Processing Laboratory, Institute of Electrical Engineering, Swiss Federal Institute of Technology in Lausanne, Lausanne 1015, Switzerland (e-mail: pascal.frossard@epfl.ch).

A. Ortega is with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089 USA (e-mail: antonio.ortega@sipi.usc.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2015.2400817

[1] Note that no explicit scene geometry information is transmitted in the multiview case.

accurately the impact of quantization on rendered view quality, as this would allow guaranteeing that a sufficient number of bits is always spent on the depth representation in order to minimize the impact of the error. But while attempts have been made, see [12], [13], developing these models is difficult in general, leading to potential inefficiencies in the achievable rate-quality trade-off for the reconstructed views.

By nature, a depth map represents the geometry of one view, and since it is used for prediction or even view synthesis, it can be viewed as a dense vector field. Analogously, a motion vector field is generally used for predictive video coding. We note that methods that consist in coding directly the depth maps are in fact performing quantization of the depth vectors. Yet, in the similar context of video coding, direct quantization of motion information has been considered [14], [15] but is in general not used. By analogy, we argue that quantization of depth maps is not the most efficient to code this information, even if smart RD optimization criteria are used [12], [13], [16]–[20]. In practice in video coding it has been shown that it is much more efficient to adapt the rate used for motion vector information at the block matching stage, *e.g.*, by selecting the "right" block size, rather than by directly quantizing a dense motion vector field. Here we adopt a similar philosophy and, instead of lossy compression of depth maps, we propose lossless transmission of a geometry representation that captures only the information needed for the required view predictions.[2] Our goal is to transmit "just enough" geometry information for accurate representation of a given set of views.

Specifically, we propose a new Graph-Based Representation (GBR) for geometry information, where the geometry of the scene is represented as connections between corresponding pixels in different views. In this representation, two connected pixels are neighboring points in the 3D scene. The graph connections are derived from dense disparity maps and provide just enough geometry information to predict pixels in all the views that have to be synthesized. GBR drastically simplifies the geometry information to the bare minimum required for view prediction. This "task-aware" geometry simplification allows us to control the view prediction accuracy before coding.

In more detail, the GBR is constructed as follows. The first view in the set (View 1) is represented by its color information. Then the GBR represents the *new* pixels of View 2 (*i.e.*, pixels that are not present in View 1, such as disoccluded pixels) and links them to particular pixels in View 1. The same approach is repeated along the view direction. Hence, the resulting representation describes 3D points of the scene *once and only once*, *i.e.,* the first time they are captured by one of the cameras, and links them through the different views in the graph. This simplified geometry information is derived from disparity values, at integer or *sub-pixel precision*.

Throughout this paper, we compare our approach to a standard method whose geometry is represented by depth maps. We demonstrate the potential of GBR in controlling the view synthesis error when the geometry coding rate is low. In the experimental section, we study the behavior of both GBR and the depth-based scheme while compressing multiview geometry of static images. In particular, we compare how both schemes can adapt their respective geometry representations as a function of the complexity of the view prediction process. Our experimental results demonstrate that our GBR leads to a compact geometry representation with better control of compression artifacts as compared to direct compression of depth. We only focus on the geometry information in this paper, and on its importance for proper synthesis. The coding of the color information on GBR is an important and interesting problem too, which we are considering outside of the scope of this paper. Our preliminary work on the color coding problem [21] uses a graph-based wavelet transform and an adapted SPIHT algorithm. It is shown that GBR structure leads to efficient compression of disoccluded pixels, outperforming existing methods. Moreover, although our experiments are performed with multiview images, GBR might be useful in the MVC or 3DVC contexts [22], where it is shown that cross-view prediction greatly helps for the coding of key pictures. Hence, GBR might be a promising alternative for inter-view anchor frame coding.

The rest of this paper is organized as follows. In Section II, we discuss related work. In Section III, we present our GBR solution by introducing in detail the graph construction process. We then present the complete coding scheme for the transmission of multiview data with GBR (Section IV). In Section V, we detail the view reconstruction and synthesis techniques. Finally, in Section VI, we present the experiments conducted to compare a baseline depth-based scheme and the prototype GBR approach and we show the benefit of representing geometry with graphs.

II. Related Work

Depth-based representations in multiview image coding suffer from geometry inaccuracies due to lossy compression of depth information, which poses problems in both view prediction quality and compression performance. Different approaches have been proposed recently to improve overall performance while using lossy compression of depth information.

A first type of approaches model the error in geometry estimation due to lossy compression of depth using standard methods. Since depth is not directly displayed and is instead used for view synthesis, the objective is to model the impact of depth coding error on the synthesized view distortion, in order to better control where the coding losses are introduced. In [16], compression performance optimization is done by experimentally simulating the view synthesis performance of some practical operating points and choosing the best one. The minimization is done with a multi-resolution full search. Some other works propose to build a rate-distortion (RD) model for proper rate allocation [12], [17], [18]. For example, in [17], the RD model is estimated region-by-region, with each region corresponding to a different object of the scene. In [18], the RD analysis relies on some complex models

[2] Note that the representation itself will be lossy since only the information needed for rendering the view will be transmitted.

for the image textures. In [19], wavelet properties are used to separate the different components of the scene and to analyze the consequence of inaccuracies in their depth values on each object. Regardless of the chosen RD model, the compression performance optimization remains complex and strongly dependent on scene content and camera settings (baseline, geometry complexity, etc.).

As an alternative, several authors have proposed coding tools for depth maps that are built on the observation that depth maps have sharp edges and very smooth textures. Hence, their objective is to preserve the sharpness of the edges, while spending few bits on the flat or smooth parts. Examples of such coding tools that have been proposed recently include meshes [7], platelets [23], shape-adaptive wavelets [24], new block formats [25], graph-based transforms [26] and coding of depth edges [27]. These tools indeed lead to better view synthesis performance using lossily encoded depth. However, they are only based on an indirect understanding of how depth compression affects view prediction (*e.g.*, edges in the depth signal are important).

Closest to our proposed GBR, several methods have been proposed to reduce redundancy in the geometry representations for multiview data. As an example, the layered depth image (LDI) representation [28], [29] avoids the inter-view redundancies, so that the 3D points of the scene are represented once and only once, in contrast to light field, multiview or depth-based representations, but similar to our proposed approach. In LDI, pixels of multiple viewpoints are projected onto a single view, the redundant pixels are discarded and the new ones (*i.e.*, the ones occluded on this reference view) are added in an additional layer. The main drawback of LDI is that, unlike our method, it directly uses depth, associated to each of the layers. Thus, even if the geometry information is less redundant in LDI, the problem of controlling the error due to depth compression is still present, *i.e.*, no solution is provided to adapt the accuracy of the lossy depth representation to the view synthesis task. In [30], a cyclopean (*i.e.*, one 2D video) version of stereo sequences is provided. The color pixels of both views are concatenated (the redundant pixels appear only once). The depth maps are replaced by a visibility image which indicates in which view each pixel is visible (right, left or both). Thus, as in GBR, a *position* is encoded, rather than the depth that was used to compute the position. However, this method [30] remains limited to integer displacement, and scenarios involving two stereo views, unlike the GBR proposed in this paper, which allows multiple views and non-integer displacements.

In light of the above, a good way of representing the geometry signal might be to reduce the spatial redundancy across the views *and* to adapt the transmitted geometry information to the view synthesis task, which are the two main purposes of our proposed GBR. Following the idea that transmitting the displacement of pixels between views rather than encoding the raw depth signal can help to achieve these two objectives, the GBR described in this paper, initially introduced in [31] and [32], replaces depth information by graph connections that relate corresponding pixels in different views. The work developed in this paper extends our preliminary work on GBR [31], [32] by proposing i) a new graph construction strategy that handles even the most complex scene geometries, ii) a novel geometry coding algorithm that achieves competitive performance as compared to standard depth map coding techniques and iii) novel representation and rendering techniques that enable non-integer disparity compensation.

III. Graph-Based Multiview Representation

A. Multiview Image Data

We describe now our new Graph-Based Representation approach in detail. Let us consider a scene captured by N cameras with the same resolution and focal length f. The n-th view is denoted by I_n, with $1 \leq n \leq N$, where $I_n(r, c)$ is the pixel at row r and column c. We consider translation between cameras, and we assume that the views are rectified. In other words, the geometrical correspondence between the views I_n only has horizontal components. We also work under the Lambertian assumption, which states that each 3D point of the scene has the same lighting condition when viewed from every possible viewpoint. We assume that a depth image, Z_n, is available at the encoder for every viewpoint, I_n. Since the views are rectified, the relation between the depth z and the disparity d for two camera views is given by $d = \frac{f\delta}{z}$, where δ is the distance between the two cameras. In what follows, the geometry information is given by disparity values that are computed from the depth maps Z_n and the camera parameters. Our goal is to design a compact multiview representation of these N camera views that provides control of the geometry information accuracy.

B. Geometrical Structure in Multiview Images

We first analyze the effect of camera translation on the image content. Let us consider two views I_n and I_{n+1} captured by cameras that are separated by a distance δ. For the sake of clarity, we first consider integer disparities, *i.e.*, full pixel displacements, and we explain how we handle the sub-pixel precision later. The geometrical correspondence between pixels in these two views takes the form of $I_{n+1}(r, c) = I_n(r, c + d)$, where d is a disparity value. When this relation holds, pixels in certain regions in view I_{n+1} can be directly associated to pixels in corresponding regions in I_n. These correspond to the elements of the scene that are visible in both views. Alternatively, the elements that are visible only from one viewpoint are often designed under the general name of occlusions, even if their occurrence is not only due to object occlusions. More exactly, we can categorize these pixels that are present or absent only in one view, into four different types as illustrated in Fig. 2. First, a new part of the scene appears in the view because of camera translation. This usually appears from the right or left (depending on translation direction) and the new pixels are not related to object occlusions. They are called *appearing* pixels. 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. Conversely, some background pixels may become hidden by a foreground object. These are called

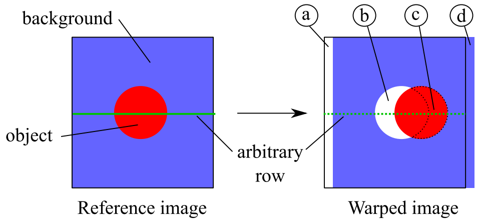


Fig. 2. Illustration of camera translation for a simple scene with a uniform background, and one foreground object. Types of pixels in depth-based interview image warping: pixels can be a) appearing, b) disoccluded, c) occluded and d) disappearing. The green plain line is an arbitrary row in the reference view and the dashed line is the corresponding row in the target view.

the *occluded* pixels. Finally, some pixels disappear in the viewpoint change, and they are called *disappearing* pixels.

We illustrate these different types of pixels and consider a row of pixels in the target view in Fig. 2. Starting from the left border, we notice that the row first contains several appearing pixels, and then some pixels of the reference view. Then, the row presents some disoccluded pixels before showing again pixels of the reference view. After that, the row contains occluded pixels that are hidden in the target view. The rest of the row matches to the reference view until a series of disappearing pixels are given at the end of the row. We want now to describe the pixels in this target row in the second view by maximizing references to elements from the corresponding row in the first view. This can be achieved by navigating between the reference view and the "new" pixels of the target view. This navigation can be guided by connections between corresponding pixels in both views. We thus propose to construct a graph that is exactly made of these connections. This graph is derived from the depth information and the number of connections varies linearly with the number of foreground objects in the view. A more formal description of the graph construction method is given next.

C. Graph Construction

A graph with N levels describes 1 reference view and $N-1$ predicted views and is constructed based on the depth maps Z_n, $1 \leq n \leq N-1$. More precisely, the depth maps are converted to *integer disparity* values D_n, $1 \leq n \leq N-1$. Since the object displacement is only horizontal in our setup with rectified views, the considered graph is constructed independently for each image row. For each row r, the graph incorporates color and geometry components, which are described by two matrices Γ_r (of size $N \times W$) and Λ_r (of size $NW \times NW$), where N is the number of levels (*i.e.,* the number of views encoded by the graph) and W is the image width in pixels. The color values in row r are given by Γ_r. The matrix Λ_r is a connectivity matrix between the NW pixels (the ordering of the NW is done from left to right in the view order, *i.e.*, the pixels of the first view are indexed from 1 to W, those of the second view from $W+1$ to $2W$, *etc.*). A connection between a pixel i and a pixel j is represented by $\Lambda_r(i, j) = 1$. In the graph construction, both the color and connectivity matrices are initialized to 0, which means "no connection" and "no color value," respectively.

We now describe in detail the construction of the graph. We show in Fig. 3 a graph construction example, with 5 levels that correspond to 1 reference view and 4 synthesized views. For the sake of clarity, we first describe in detail the graph construction of an arbitrary row r by considering only one predicted view I_2, one reference view I_1 and its associated disparity map D_1. The first level corresponds to the reference view, and thus $\Gamma_r(1, j) = I_1(r, j)$ for all $j \leq W$. Then, the connection values $\Lambda_r(i, j)$ and the color values $\Gamma_r(2, j)$ are computed based on the following principles:

- The pixels intensities are represented in the graph level (*i.e.,* view) where they appear first, which means that the second level only contains new pixels that are not present in the reference view.
- The connections $\Lambda_r(i, j)$ simply connect each new pixel to the position of its neighbor in the previous level. More precisely, a new pixel represented in a level l can be a point that is hidden by a foreground object in the previous views and becomes disoccluded at level l. If this foreground object was not in the scene, the pixel would have been visible in the previous views, near the other background pixels. The "neighbor" of this new pixel in the lower level $l-1$ is thus the pixel that is right next to the disoccluded area.

Based on these general rules, we describe now precisely how each of the pixel types shown in Fig. 2 is handled in our graph-based representation. First, the values of appearing pixels $\Gamma_r(2, j)$, are assigned to the corresponding graph vertices, and no connections are drawn to other pixels in previous levels, except for the last appearing pixel, which is linked to the first pixel of the previous level (appearing pixels are thus attached to the side of the reference image). In the example of Fig. 3, we see that the dark blue appearing pixel, in I_2, is linked to the first pixel of level 1, which means that $\Lambda_r(W+1, 1) = 1$.

As for the disoccluded pixels, let us take the case where the last pixel before foreground is at position c in level 1, and that d is its disparity value (*i.e.*, $d = D_1(r, c)$). The position of the first disoccluded pixels is thus $c+d+1$ in level 2, and the number of disoccluded pixels is equal to $D_1(r, c+1) - d$. The color values of these disoccluded pixels are stored at their position in the color matrix Γ_r. Additionally, we store the connection value $\Lambda_r(c, c+d+W+1) = 1$, that links the last background pixel of the reference view to the pixels that are disoccluded. For example, in Fig. 3, the foreground object is red. The disoccluded pixels are stored in $\Gamma_r(2, 3)$ and $\Gamma_r(2, 4)$. In level 1, the last pixel before the foreground object is at position 1 (light blue pixel). The graph thus links this pixel to the first disoccluded pixel of level 2. These two pixels are considered as neighbors. The disparity d of the background in the example of Fig. 3 is equal to 1, so that the entry $\Lambda_r(1, 22)$ in the connectivity matrix is set to 1.

The occluded pixels represent color values that are absent in the second view, and only visible in the reference one. They are described by a jump in the reference view and the jump value is stored in the connectivity matrix as a connection between the last pixel of the foreground, and the first visible pixel of the background. In the example of Fig. 3, the last pixel of the

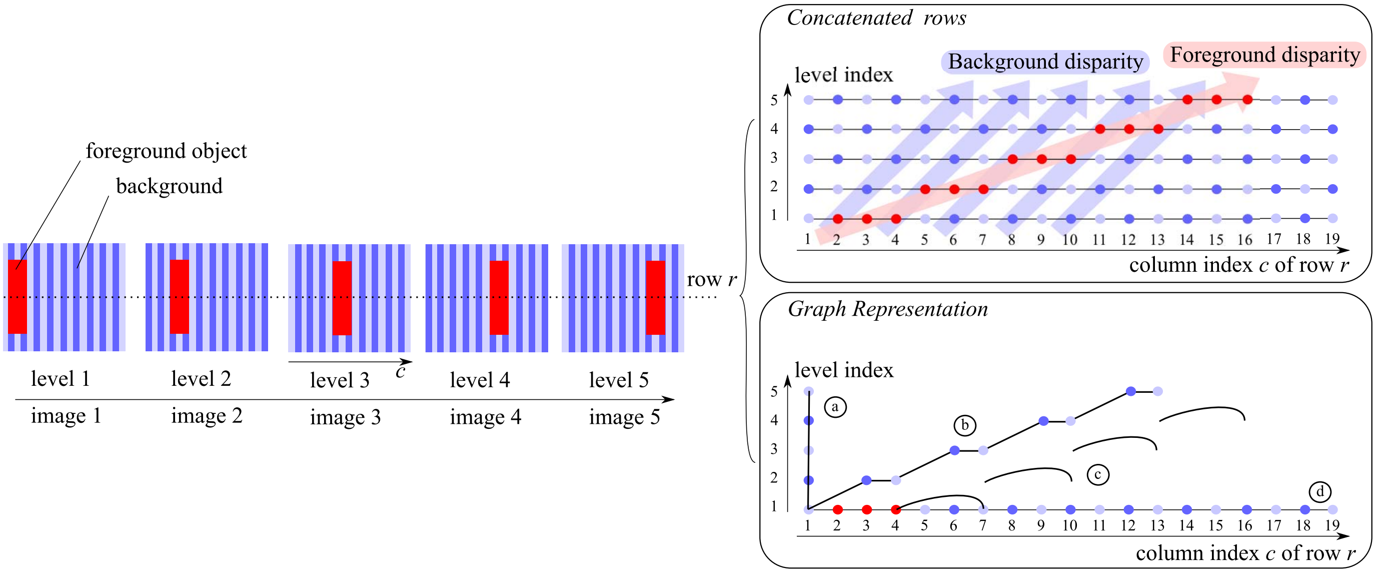


Fig. 3. Graph construction example: the blue texture background has a disparity of 1 at each view and the red rectangle foreground has a disparity of 3 for each view. This example graph contains all different types of pixels: a) appearing, b) disoccluded, c) occluded and d) disappearing.

foreground object is at position 4 in level 1 while the first background pixel visible in level 2 is at position 7. Thus, we have $\Lambda_r(4, 7) = 1$. Since no new pixel is contained in the second view, we do however not store any value in the color vector. The connections representing this jump are deduced from the disparity values of both foreground and occluded pixels. We give, later, a precise algorithm to determine in all the cases this jump value. Finally, the disappearing pixels are not represented by any connection nor color value.

The GBR construction strategy introduced above is presented in a general form in Algorithm 1. The inputs are two luminance views I_1 and I_2, the depth image Z_1 and the distance between the two cameras δ. First, we convert the depth image into a dense disparity map D (lines 4 to 6). The non-integer disparity value is simply rounded to the closest integer. Then, the graph construction is done row by row. The pixels of I_1 are first inserted in the first level of the luminance matrix (lines 8 to 10). We then insert the appearing pixels in level 2 of the luminance matrix (lines 11 to 13). After this operation, we go through the dense disparity map of I_1 and detect disocclusions (lines 22 to 28) and occlusions (lines 28 to 30, and Algorithm 2). Finally, for building a graph with more than 2 views, one simply needs to repeat the operations from lines 11 to 37 for every predicted view, while taking as starting point the most recent view. This leads to H matrices Λ_r and Γ_r (H is the number of rows) that are concatenated in two 3D matrices Λ and Γ and constitute the complete GBR data structure. We note that the construction of such a graph can easily be extended to other view orderings. In other words, the reference view can be any of the N views, with the remaining views are derived from the reference or previously built views. If an intermediate view is chosen as a reference, two independent graphs would be generated: one going to the left and one to the right. We leave the problems of the optimization of the view order, and in particular the selection of the reference view, for future work.

With the above graph construction method, the graph representation is sparse (only a small fraction of entries is non-zero). This avoids all redundancy in the color value description since the pixels values stored at a given level in Γ_r are only those that are not present in the lower levels. Another important advantage of this graph representation is in the multi-level structure, where the connections in one level are related to connections in other lower levels. Therefore, a reconstruction algorithm only needs to go through these connection chains to reconstruct the different multiview images. This makes the reconstruction process simple and well suited for inter-view navigation, with smooth transitions between the views. Finally, the main property of the proposed GBR structure is its ability to control the geometry information. Indeed, the input depth information is compressed into a geometry signal "just enough" for view prediction.

The above GBR description corresponds to the representation of N input views for which color and depth images are known. Note that this information is sometimes not available, but our proposed GBR can also be used in cases where only a single view is known, and where the other views to synthesized are virtual. The GBR construction in such a scenario is described in detail in Section V-B.

D. Non-Integer Disparity

The graph introduced before only deals with integer disparities in the construction of connections between pixels in different views. However, the actual disparity values obtained from depth data might not always be integer. Hence, the rounding operation in the graph construction (see line 5 of Algorithm 1) induces a geometry error in the view prediction process. Indeed, sub-pixel disparity values imply that a pixel corresponding to an object in the scene is not necessarily captured at integer pixel positions in different views, which

Algorithm 1 GBR Construction for Two Levels

```
Input:
 1: $\{I_1, I_2\}$ - luminance images of height $H$ and width $W$
 2: $Z_1$ - the depth map corresponding to view 1
 3: $\delta$ - the distance between the two views
Output: The color and geometry matrices $\mathbf{\Gamma}$ and $\mathbf{\Lambda}$

Algorithm:
    //Convert depth $Z_1$ to dense disparity map $D$ with
    rounding operation
 4: for $r := 1$ to $H$ and $c := 1$ to $W$ do
 5:   $D(r,c) \leftarrow \left\lfloor \frac{f\delta}{Z_1(r,c)} + 0.5 \right\rfloor$
 6: end for

 7: for $r := 1$ to $H$ do

      //Insert $I_1$ in the first level of the color matrix $\mathbf{\Gamma}$
 8:   for $c \leftarrow 1$ to $W$ do
 9:     $\mathbf{\Gamma}(r,c,1) \leftarrow I_1(r,c)$
10:   end for

      //Insert the $D(r,1)$ appearing pixels ((a) in Fig.3) in
      the $2^{nd}$ level of $\mathbf{\Gamma}$
11:   for $c := 1$ to $D(r,1)$ do
12:     $\mathbf{\Gamma}(r,c,2) \leftarrow I_2(r,c)$
13:   end for
      //Link the last appearing pixel to the first pixel of
      level 1
14:   $\mathbf{\Lambda}(r, W + D(r,1), 1) \leftarrow 1$

15:   $c_1 \leftarrow 2$ //current column index in $I_1$
16:   $d_p \leftarrow D(r,1)$ //previous disparity value
17:   $c_{\text{stop}} \leftarrow D(r,c_1)+1$ //column index in level 2 that serves
      as stopping criterion

18:   while $c_{\text{stop}} \leq W$ do
19:     $d_c \leftarrow D(r,c_1)$ //current disparity value
        //test if $d_c \neq d_p$ in the case of occlusion or
        disocclusion
20:     if $d_c \neq d_p$ then
21:       $\Delta_{\text{disp}} = d_c - d_p$
          //test if $\Delta_{\text{disp}}$ corresponds to a disocclusion $(> 0)$
          or an occlusion $(< 0)$
22:       if $\Delta_{\text{disp}} > 0$ then
23:         $c_{\text{stop}} \leftarrow c_{\text{stop}} + \Delta_{\text{disp}}$
            //Fill the disoccluded pixels ((b) in Fig. 3) in
            the second level of $\mathbf{\Gamma}$
24:         for $c_2 \leftarrow c_1 + d_p$ to $\min(c_1 + d_p + \Delta_{\text{disp}} - 1, W)$ do
25:           $\mathbf{\Gamma}(r,c_2,2) \leftarrow I_2(r,c_2)$
26:         end for
            //Include the link between the two neighbors in
            the 3D space ((b) in Fig. 3) in $\mathbf{\Lambda}$
27:         $\mathbf{\Lambda}(r, c_1 - 1, c_1 + d_p + W) \leftarrow 1$
28:       else
            //Deal with the occlusion using Algorithm 2
29:         $(\mathbf{\Lambda}, \mathbf{\Gamma}) \leftarrow$ OcclusionManager($c_1$, $c_2$, $d_p$, $D$, $\mathbf{\Gamma}$, $\mathbf{\Lambda}$, $I_2$)
30:       end if
31:     else
32:       $c_{\text{stop}} \leftarrow c_{\text{stop}} + 1$
33:     end if
34:     $d_p \leftarrow d_c$
35:     $c_1 \leftarrow c_1 + 1$
36:   end while

37: end for
```

Algorithm 2 OcclusionManager

```
Input:
 1: $c_1$, $c_2$, $d_p$, $D$, $\mathbf{\Gamma}$, $\mathbf{\Lambda}$, $I_2$ (see Algorithm 1)
Output: The color and geometry matrices $\mathbf{\Gamma}$ and $\mathbf{\Lambda}$

Algorithm:
    //The last pixel of the foreground object
 2: $c_{last} \leftarrow c_1$
    //The pixel to link with $c_{last}$ and determined in the loop
    from lines 6 to 18
 3: $c_{temp} \leftarrow c_1$
    //Disparity of $c_{last}$
 4: $d_{temp} \leftarrow d_p$
 5: $stop = 0$

    //The loop is looking for the pixel linked with $c_{last}$
    after the occlusion, a disocclusion may appear
 6: while $stop = 0$ do
 7:   $c_{temp} \leftarrow c_{temp} + 1$
 8:   $d_{temp} \leftarrow d_{temp} - 1$
 9:   $d_{cur} \leftarrow D(r, c_{temp}, 1)$

      //if a disocclusion appears
10:   if $d_{temp} \neq d_{cur}$ then
        //size of the disocclusion
11:     $N_{disoc} = d_{cur} - d_{temp} - 1$
        //Handle the disocclusion as in Algorithm 1 lines 22
        to 28
12:     for $c_2 := c_{last} + d_p + 1$ to $\min(c_{last} + d_p + N_{disoc}, W)$ do
13:       $\mathbf{\Gamma}(r, c_2, 2) \leftarrow I_2(r, c_2)$
14:     end for
15:     $\mathbf{\Lambda}(r, c_{last}, c_{last} + d_p + W) \leftarrow 1$
16:     $stop = 1$
17:   end if
18: end while
    //Link the last pixel of the foreground $c_{last}$ to the first
    pixel after the occlusion $c_{temp}$ determined by the lines
    6 to 18
19: $\mathbf{\Lambda}(r, c_{last}, c_{temp}) \leftarrow 1$
20:
```

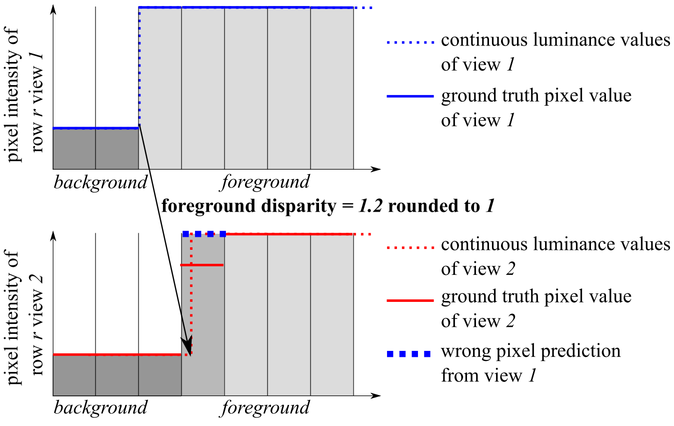

pixel intensity of row r view 1
continuous luminance values of view 1
ground truth pixel value of view 1
background
foreground
foreground disparity = 1.2 rounded to 1
pixel intensity of row r view 2
continuous luminance values of view 2
ground truth pixel value of view 2
wrong pixel prediction from view 1
background
foreground

Fig. 4. Illustration of prediction error between two views when float disparity occurs for a given row r, view 1 and view 2.

leads to prediction errors when disparity is constrained to be integer. A concrete example is given in Fig. 4, where the transition between background and foreground may fall in a non-integer pixel position for one of the views. In such a case, simple view prediction that consists in copying the pixel intensity from the reference view may fail, and smarter interpolation tools may be needed.

In order to compensate for these geometry errors, we extend our GBR to *sub-pixel precision*. We denote ε the level of sub-pixel precision, so that $\varepsilon = 0.5$ indicates that half pixel accuracy is supported. Sub-pixel precision is handled by simply enlarging the grid. Consider the case where $\varepsilon = 0.5$. We upsample the horizontal pixel grid of the target view by a

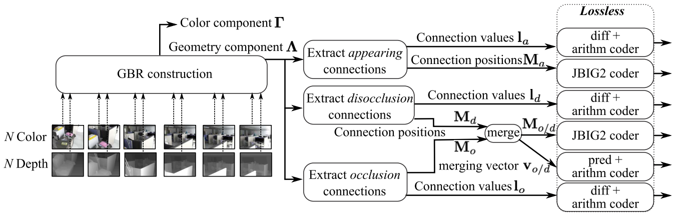

Color component $\mathbf{\Gamma}$
Geometry component $\mathbf{\Lambda}$
GBR construction
N Color
N Depth
Extract *appearing* connections
Extract *disocclusion* connections
Extract *occlusion* connections
Connection values $\mathbf{l}_a$
Connection positions $\mathbf{M}_a$
Connection values $\mathbf{l}_d$
$\mathbf{M}_d$
Connection positions
merge
$\mathbf{M}_{o/d}$
$\mathbf{M}_o$
merging vector $\mathbf{v}_{o/d}$
Connection values $\mathbf{l}_o$
Lossless
diff + arithm coder
JBIG2 coder
diff + arithm coder
JBIG2 coder
pred + arithm coder
diff + arithm coder

Fig. 5. Coding strategy for GBR geometry information Λ.

factor of $1/\varepsilon = 2$. The grid of the reference view is kept with an integer regular grid since it is the form of the ground-truth input depth map. The line 5 of Algorithm 1 is modified as follows:

$$D(r, c) \leftarrow \text{round}\Big(\frac{f\delta}{Z_1(r, c)\varepsilon}\Big). \tag{1}$$

In other words, the disparities are still integer but in a larger grid, which leads to a sub-pixel precision. The following graph construction remains similar than in Algorithm 1. The only difference comes at the reconstruction step, as explained in Section V. This new graph functionality thus allows us to consider any sub-pixel precision $\varepsilon = \frac{1}{p}$, with $p \in \mathbb{N}^*$.

IV. GBR Geometry Coding

In this section, we propose an efficient geometry encoding algorithm where the GBR connections are compressed to provide a compact description of multiview image geometry. As we can observe in the example of Fig. 3, the matrix of connections for row r, Λ_r, is sparse. In order to encode it with a small number of bits, we propose the following coding strategy summarized in Fig. 5. Let us take, without loss of generality, the example of a GBR representing only two views (one reference and one predicted view). The connection matrix Λ is processed in order to split the connections into three sets representing respectively the appearing, the disoccluded and the occluded pixels. More precisely, for every non-zero value in Λ, we analyze whether the connections goes to a lower level, an upper level or to the same level, corresponding respectively to appearing, disoccluded and occluded pixels. For each type of connection, we extract one mask of size $H \times W \times (N-1)$, a binary image indicating the position of the connection in the reference level (called $\mathbf{M}_a$, $\mathbf{M}_d$ and $\mathbf{M}_o$ for respectively appearing, disoccluded and occluded pixels), and a vector storing the connection values (*e.g*, k if the connection links pixel i to $i+k$) that are called $\mathbf{l}_a$, $\mathbf{l}_d$ and $\mathbf{l}_o$ for respectively appearing, disoccluded and occluded pixels. The mask of the appearing pixels $\mathbf{M}_a$ is extremely sparse, and is thus coded separately. The occlusion and disocclusion masks are merged together (giving a new mask $\mathbf{M}_{o/d}$), along with a binary vector, $\mathbf{v}_{o/d}$, indicating the connection type in each of the occlusion/disocclusion locations. More precisely, for all elements i, $\mathbf{v}_{o/d}(i) = 0$ if $\mathbf{M}_{o/d}(i) = 1$ and $\mathbf{M}_o(i) = 1$; and $\mathbf{v}_{o/d}(i) = 1$, if $\mathbf{M}_{o/d}(i) = 1$ and $\mathbf{M}_d(i) = 1$. We never have the case where $\mathbf{M}_o(i)$ and $\mathbf{M}_d(i)$ are both equal to 1, since a pixel cannot be linked to an occluded and a disoccluded pixel at the same time. The two masks $\mathbf{M}_a$ and $\mathbf{M}_{o/d}$ are compressed losslessly using a JBIG2 coder. JBIG2 [33] is an efficient image compression standard developed for binary images such as the masks $\mathbf{M}_a$ and $\mathbf{M}_{o/d}$. The three vectors of connection values $\mathbf{l}_a$, $\mathbf{l}_d$ and $\mathbf{l}_o$ are first processed with a differential operator and then losslessly coded with an arithmetic coder. The binary vector $\mathbf{v}_{o/d}$ is coded with an arithmetic coder. However, in order to reduce its entropy we apply a simple predictor, which predicts the next binary element by adding 1 (binary sum) to the previous element. This is justified by the fact that the geometry information often alternates between occlusions and disocclusions (*e.g.,* at each side of an object).

In a nutshell, the proposed coding strategy exploits the sparsity of the connections using an efficient binary image coder. The rest of the algorithm exploits the dependencies between the disparity values themselves to reduce the coding cost. The efficiency of our geometry representation thus comes from i) a simplified lossy version at the representation step in the GBR, and ii) a lossless compression algorithm that exploits sparsity in the geometry sparsity and the correlation between the disparity values.

In order to decrease the geometry coding costs, we can furthermore estimate the geometry in some views, instead of coding it for every view. Hence, we introduce the possibility of removing some levels (*i.e.*, views) from the graph structure and interpolating them at the receiver. In other words, the graph contains $L < N$ levels. In this case, fewer bits are required for encoding the geometry since the number of levels is reduced. When a level is removed from the graph, the graph construction is directly performed between the previous and the next level (*e.g.*, edges connect levels 2 and 4 directly, instead of passing through level 3 and the pixel values of the level that is skipped are stored in the upper level). However, the interpolation of views at the decoder may create some distortion in the geometry. The interpolation of a view at the decoder is done by disparity compensation with the two closest received views (see Sec. V-B). The two disparity-compensated estimations of the interpolated view are then merged, which results in a synthesized view with no disocclusion. The choice of the number of levels L and of which levels are included in the graph is a trade-off between the bit rate required for graph compression, and the distortion of the reconstructed view. In this paper, we choose L and the views involved in the graph with a full search algorithm that evaluates the graph size and the rendered distortion for many configurations. We have observed experimentally that the set of reasonable structures is in fact small, and only "intuitively good" structures, *e.g.*, where views included in the graph are regularly spread over the view set, tend to be efficient. Thus, in practice an exhaustive search will not be needed and it will be sufficient to choose one such regular structure. Note that choosing the views involved in the GBR is similar to choosing the best Group Of Pictures (GOP) structure in MVC or 3DVC standards [34]. Indeed, a skipped view in our graph is equivalent to a synthesized virtual view in standard compression schemes.

Finally, color compression algorithms may also benefit from the GBR multiview representation, although this outside of the scope of this paper. For example, we have recently shown in [21] that graph edges in the GBR, which link pixels in

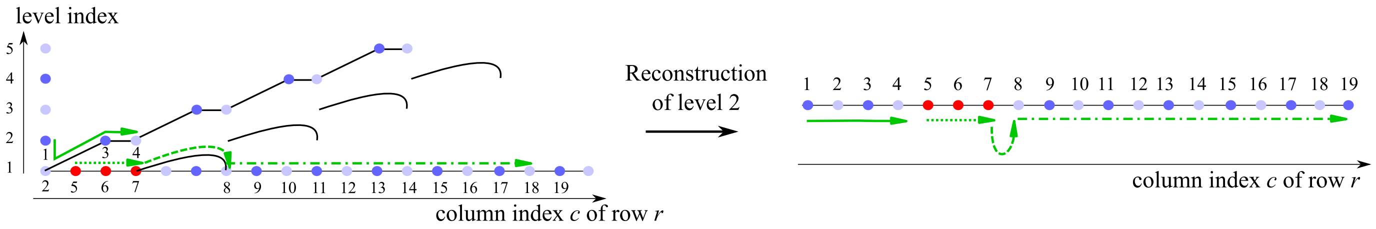


Fig. 6. Reconstruction of level 2 with the toy example of Fig. 3. The green arrows indicates the graph exploration order for view reconstruction.

different views that are neighbors (*e.g.*, a disoccluded pixel and its neighbors in the neighboring view), can be exploited for compression using a graph-based transform (designed over the GBR graph structure).

V. GBR View Reconstruction and Synthesis

A. View Reconstruction From GBR

The graph information described in the previous section is used directly for view reconstruction at the decoder. The reconstruction of a certain view requires the color values and the connections of all lower levels. The reconstruction of the color values in the current view is performed by navigating the graph across its different levels. This navigation starts from the border of the image at the level (*i.e.,* view) that needs to be constructed; it then follows the connections and refers to the lower levels when no color information is available at current level. We show in Fig. 6 an example of a view synthesis for the image of level 2, based on the graph in the example of Fig. 3. The pixel numbering is done with respect to the column index of I_2, as in Fig. 3. The reconstruction starts with the appearing pixel 1 at level 2. Then, it moves to the reference level and fills pixel color values until encountering a non-zero connection. 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 algorithm jumps across columns in the reference view and continues the decoding of the pixels in the reference level for pixel 8 to 19 until it recovers the entire row. The reconstruction of the other views (*i.e.*, the other levels of the graph in multiview images) is done recursively.

We see that the reconstruction process is very simple and that the required geometry information is captured in a flexible and controlled way by the graph connections. In the case of sub-pixel precision $\varepsilon = \frac{1}{p}$, with $p \in \mathbb{N}^*$, the reconstruction process is also simple and intuitive. We recall that the target image size has been increased $\times p$ in the GBR. At the receiver side, the pixels of predicted levels are interpolated from the irregular grid made of the known pixels reconstructed from GBR. More precisely, we run the reconstruction function with the enlarged grid $[1, 2, \dots, Wp]$. After that step, only a small percentage of pixels (at most 1 out of p) can be found at irregular locations on the grid. From those irregularly located pixels, we use a linear interpolation to obtain pixels at locations $[1, p, 2p \dots, Wp]$, which correspond to pixels $[1, 2, \dots, W]$ of the reconstructed level. This is done for every row and every level.

Algorithm 3 Disparity Retrieval From GBR for an Image Row

```
Input:
  The graph connections $\mathbf{\Lambda}$ for row $r$
Output: The disparity map $D$ for row $r$

Algorithm:
  $D(r,1) \leftarrow$ number of appearing pixels
  $d_{\text{temp}} \leftarrow D(r,1)$

  for $c := 1$ to $W$ do

    if $\exists \quad j$ such as $\mathbf{\Lambda}(r,c,j)$ indicates a disocclusion then
      $d_{\text{temp}} \leftarrow d_{\text{temp}}+$ number of discoccluded pixels
      $D(r,c) \leftarrow d_{\text{temp}}$
    end if

    if $\exists \quad j$ such as $\mathbf{\Lambda}(r,c,j)$ indicates an occlusion then
      $d_{\text{temp}} \leftarrow d_{\text{temp}}-$ number of occluded pixels
      $D(r,c) \leftarrow d_{\text{temp}}$
    end if

  end for
```

B. View Synthesis

In the literature, depth maps are not only used for assisting multiview coding, *i.e.*, compressing a set of captured views. This geometry information further enables view synthesis [35], *i.e.*, the generation of new virtual camera views for which no color or depth information is available[3]. This is very helpful in many applications such as free viewpoint television [36]. We explain here how this task can be achieved with our GBR, and how our new compact geometry description is more suited to view synthesis as well. From the connections stored in the graph Λ, we are able to retrieve the disparity information, hence the geometry of the scene. The way these disparity maps are extracted from GBR is detailed in Algorithm 3.

From the disparity maps, the GBR is then able to synthesize any virtual view at intermediate positions between reference view and the first predicted view, exactly as in depth-based scheme. While view synthesis can be performed from an existing GBR, we can also directly build the graph for synthesized virtual views. In other words, this graph would no longer be built with captured views, but with only one reference view and multiple virtual views. The graph construction is done

[3] In contrary to view reconstruction where the color and geometry information of predicted view point is available at the encoder.

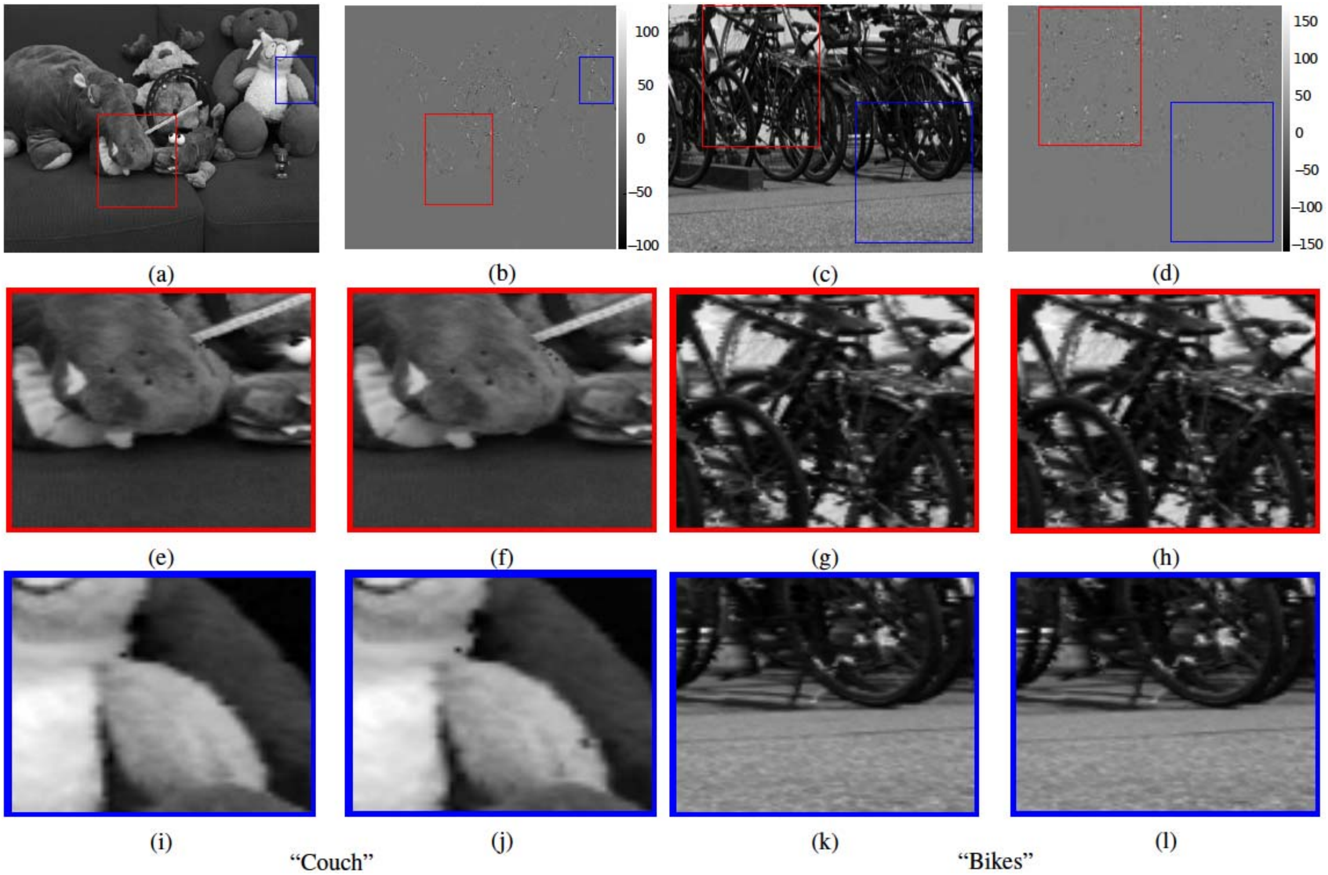


Fig. 7. Comparison of visual artifacts in I_0, the optimally synthesized view, and $I_{0.05}$, synthesized with compressed depth at 0.05 dB PSNR. (a) and (c) are the original images. (b) and (d) are difference images $I_0 - I_{0.05}$. (e), (g), (i) and (k) are some parts of I_0, while (f), (h), (j) and (l) are the same parts of $I_{0.05}$.

exactly as before, except that the "new pixels" of levels greater than 1 are not filled (they are unknown), and are inpainted at the decoder. In that way, the GBR provides the exact amount of information needed for a particular view synthesis, which is efficient in terms of bit-rate as we demonstrate in Section VI.

C. Complexity Discussion

From the view reconstruction and synthesis presented above, the GBR decoding operations that could be bottleneck are those involving pixel-based operations, as well as the inpainting techniques. Pixel-based operations are needed for view projection using the graph information. It is however the case in 3D-HEVC or any other depth-based decoders as well, since views are generally synthesized with a pixel-based DIBR algorithm. Moreover, this computational complexity can be reduced by a smart parallelization of the pixel projections. Finally, inpainting is used only if disocclusion occurs after bidirectional view synthesis, exactly as in the depth-based schemes. Overall, the GBR representation does not involve more computationally complex tasks than traditional depth-based representations with view synthesis at the decoder.

VI. Geometry Coding Experiments

In this section, we present extensive experiments to evaluate the performance of our representation for coding the geometry in multiview images. We use several datasets summarized in Table I. They all provide a set of multiple rectified views, for which color and raw depth information are available. One view is used as reference (*i.e.*, View 1) and both its color and depth information are used to synthesize the other views. We evaluate the trade-off between the geometry bit rate and the quality of synthesis views that are constructed using decoded geometry information. Such tests are classical in the geometry compression literature; they provide a quantifiable and meaningful evaluation of geometry compression algorithms. In all the results below, we only consider the bit rate required for geometry, and we assume that the color information of the reference view is available losslessly to all the algorithms we compare.

TABLE I
The Rectified Multiview Plus Depth Image Set Used in Our Experiments. Sources : *a*. Middlebury University; *b*. Disney Research

Sequence	Resolution	views	Geometry complexity
Venus[a]	383×435	$1 \dots 5$	low
Sawtooth[a]	380×434	$1 \dots 5$	low
Tsukuba[a]	252×348	$1 \dots 5$	medium
Couch[b]	268×402	$25, 29, \dots, 50$	medium
Statue[b]	172×263	$25, 29, \dots, 50$	medium-high
Bikes[b]	176×268	$25, 29, \dots, 50$	medium-high
Church[b]	263×401	$25, 29, \dots, 50$	high
Mansion[b]	345×549	$25, 29, \dots, 50$	very high

In detail, we synthesize multiple views from one reference view that is given with uncompressed color information and with compressed geometry information. We compare our scheme with three standard depth compression schemes: JPEG2000 [37], HEVC Intra [11] and an edge-adaptive wavelet (EAW) transform based coder [24]. For the three schemes we compare against, we code the depth map of the reference view with different quantization levels, and evaluate a rate-distortion performance curve. Using the decoded depth image, we predict the target views using a classical depth-image based rendering technique [38]. The expansion and disocclusion holes are filled using a simple horizontal interpolation algorithm, that estimates every missing pixel with a linear combination between the two closest known pixel values of the same row. In our GBR, instead of a depth image, we represent the geometry information with a graph whose connections link pixels in the target views and in reference view. More precisely, we build a GBR between the reference view and one target view in the set of target viewpoints. The GBR has a sub-pixel precision ε. The graph is coded using the proposed algorithm introduced in Sec. IV. We then synthesize all viewpoints with the geometry information provided by the GBR, as described in Section V. Since "new" pixels in the target views are absent in the GBR, they are interpolated at the reconstruction process, using the same technique as in the depth-based prediction schemes. We measure the PSNR (in dB) of the synthesized views with respect to the original ground-truth view. The best PSNR for the synthesized view would be the one corresponding to the use of non-compressed depth for view synthesis. This will be called "optimal" quality in what follows. Note that this optimal quality is not infinite because of: i) input geometry inaccuracies and ii) the existence of disoccluded regions for which the actual pixel values are not available and need to be interpolated.

We first consider the scenario where the goal is to reach nearly optimal synthesis quality with a rate that is as low as possible. In Fig. 7, we show some visual examples of synthesized views that are within just 0.05 dB of the optimal quality. We first observe in Fig. 7(b) and (d), that while the overall loss is small (0.05 dB) errors are localized and can be large, *e.g.*, error magnitudes of around 150 for some pixels. High error pixels (absolute error greater than 30) correspond however to only $0.2-0.6\%$ of the total number of pixels. Figure 7(e)-(l) shows that the quality is still acceptable. We show in Figs. 8 and 9, similar visual results when the synthesized quality is at 0.05 and 0.2 dB from the optimal quality. We clearly observe in Figs 8 and 9 (f),(i), that a loss of 0.2 dB leads to significant visual artifacts. The error is still localized but the number of erroneous pixels (with an error >30) achieves $1.2-2.0\%$ of the image size, which becomes highly noticeable. We note here that a small PSNR loss (even as low as 0.2 dB) can substantially decrease the visual quality. In Table II, we show the minimum rate required to obtain a synthesized view within 0.2 and 0.05 dB of optimal for each of the geometry compression schemes. Since the performance of GBR is only one point in the RD space, the GBR rates are only reported for the scenario of -0.05 dB. We have shown previously that a loss of 0.2 dB is a range for which the errors are visually too large, so we present the rate comparison between GBR and the baseline methods for the case of -0.05 dB. Results are reported in Table III.

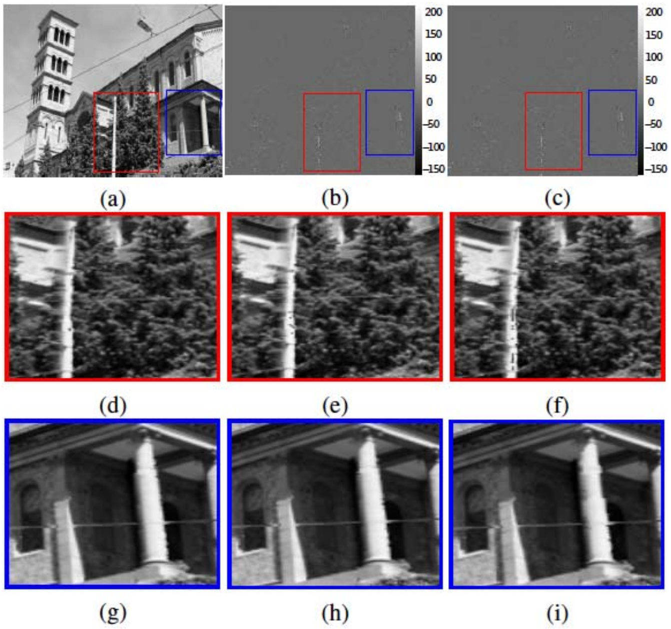


Fig. 8. Visual artifacts between I_0, the optimally synthesized view, $I_{0.05}$ and $I_{0.2}$, the ones synthesized with compressed depth (HEVC) and whose PSNR are respectively at 0.05 and 0.2 dB from the optimal quality, for the "Church" data set. (a) is the original image. (b) and (c) are the difference images, $I_0 - I_{0.05}$ and $I_0 - I_{0.2}$. (d) and (g) are some parts of I_0, while (e), (h) are the same parts of the synthesized view $I_{0.05}$ and (f), (i) are the same parts of the synthesized view $I_{0.2}$.

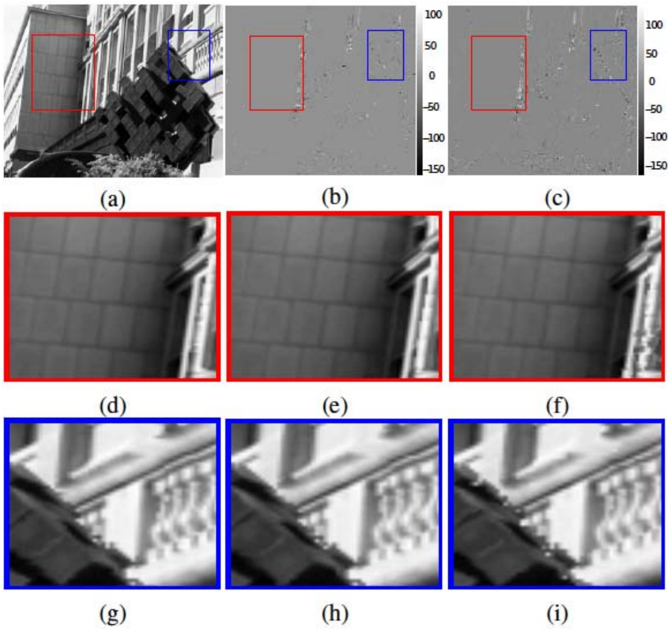
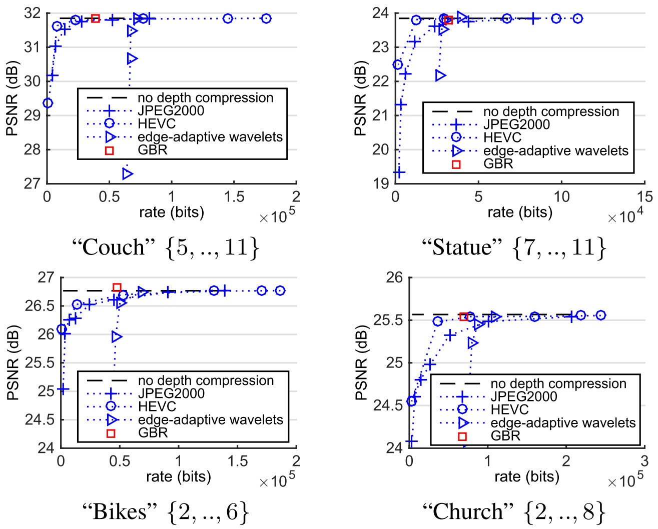


Fig. 9. Visual artifacts between I_0, the optimally synthesized view, $I_{0.05}$ and $I_{0.2}$, the ones synthesized with compressed depth (JPEG2000) and whose PSNR are respectively at 0.05 and 0.2 dB from the optimal quality, for the "Statue" data set. (a) is the original image. (b) and (c) are the difference images $I_0 - I_{0.05}$ and $I_0 - I_{0.2}$. (d) and (g) are some parts of I_0, while (e), (h) are the same parts of the synthesized view $I_{0.05}$ and (f), (i) are the same parts of the synthesized view $I_{0.2}$.



“Couch” $\{5,..,11\}$ “Statue” $\{7,..,11\}$

“Bikes” $\{2,..,6\}$ “Church” $\{2,..,8\}$

Fig. 10. Reference View 1 is used for the synthesis of multiple views, after the coding of geometry information with multiple techniques.

TABLE II

Minimal Rate (kb) to Achieve a View Synthesis Quality at 0.2 and 0.05 dB From the One Obtained With Lossless Depth. GBR Has Only One Value at −0.05 dB

	−0.2 dB			−0.05 dB			
	JP2K	HEVC	EAW	JP2K	HEVC	EAW	GBR
“Couch”	20.6	**10.1**	70.1	49.0	40.0	73.4	**39.2**
“Statue”	28.8	**11.1**	28.9	56.5	**12.2**	29.85	31.9
“Bikes”	39.8	**37.5**	51.9	80.2	65.2	64.7	**48.2**
“Church”	63.7	**31.7**	84.5	127.5	**60.0**	100.5	70.02

TABLE III

Rate Comparison Between GBR and Baselines Compression Methods With Synthesized Views at 0.05 dB From the Optimal Quality, Based on Results Obtained in Table II

	GBR vs JP2K	GBR vs HEVC	GBR vs EAW
“Couch”	−20.0%	−2.0%	−46.6%
“Statue”	−43.5%	+161%	−6.87%
“Bikes”	−39.9%	−26.0%	−25.5%
“Church”	−45.0%	+16.7%	−30.3%

Fig. 10 shows the rate-distortion curves of the three baseline depth compression schemes, and the rate-distortion point of our GBR technique. The PSNR values shown in these curves are the average PSNRs over all the synthesized views. The rates are those needed for geometry representation.

We now analyze the results of Fig. 10 and Tables II and III, from three different perspectives: 1) comparison between GBR and edge-aware encoder (EAW), 2) comparison between GBR and high performance standard codecs (HEVC/JPEG2000), 3) a specific analysis of GBR’s behavior.

A. GBR Versus EAW

Comparing GBR and EAW in Fig. 10, Tables II and III, we see that GBR performs overall better. This is due to two main reasons. Firstly, the EAW method spends a significant bit rate in coding the geometry (*i.e.*, preserving the edges in the depth image). In Fig. 10, we can see that all the EAW curves are lower-bounded by the rate needed for edge transmission. Secondly, preserving edge information is not always the best strategy. To prove this statement, we compare the simplified geometry provided by GBR with the edges that EAW, as well as many other traditional depth coding algorithms [23], [24], [26], aim at preserving. In Fig. 11, we show the depth maps, the edges detected by gradient-based algorithms (used in EAW-based coder), and the positions of our connections in the GBR. We can see how GBR differs from traditional methods by considering two cases. Firstly, when smooth depth gradients exist, our GBR may introduce geometry information (*i.e.*, a graph connection) in the middle of the gradient, representing a disparity change. In contrast, depth-based approaches compress this smooth region very crudely, allocating most of the bit budget to coding the object edges. Secondly, some depth edges do not lead to disparity jumps in our GBR, as we can see for example in the “Couch” example. This is clearly visible in the compressed geometry images shown in Fig. 12. These observations outline the fact that the proposed GBR builds a geometry signal that is adapted to the geometry of the scene and to the complexity of the view prediction: it preserves every useful piece of geometry information and not only the edges as EAW.

B. GBR Versus JPEG2000 / HEVC

First, we observe in Figs 8 and 9, that scaling the rate is not without effect and might be useless in the view synthesis context, in the sense that a small PSNR loss (0.05 dB) can already have a large visual impact on the quality of the synthesized view. Second, we observe that our GBR approach almost always outperforms the JPEG2000 approach. Although it is true that JPEG2000 is not the highest performance image coder, this result provides a good reference for the performance of our GBR technique. Fig. 10 also shows that GBR and HEVC are not far from each other in terms of performance and in Table II, and III we see that GBR outperforms HEVC for two out of the four sequences, indicating GBR based coding can be competitive with even the most recent and powerful compression standards. We additionally note that for “Statue,” the difference between GBR and HEVC performances is significant. If we look at Fig. 10, we notice that this is mainly due an exceptionally good performance of HEVC for this sequence. Note that GBR is quite consistent in its relative comparison with EAW and JPEG2000, and we see that the curve of HEVC at low bitrate reaches the optimal PSNR very quickly. The “Statue” sequence has a specific geometry structure, made of several objects parallel to the foreground, which leads to a piecewise constant depth image. Images that are piecewise constant are particularly easy to encode with HEVC (given intra prediction) and this could explain such a gap between HEVC and the other methods that do not perform intra prediction. Conversely, for images where depth has more complex characteristics (e.g., slopes) HEVC’s intra prediction is not efficient and rate-distortion comparisons tend to favor GBR.

C. GBR’s Behavior Analysis

We finally show that GBR adapts its geometry representation to the task at hand. In particular, the geometry information

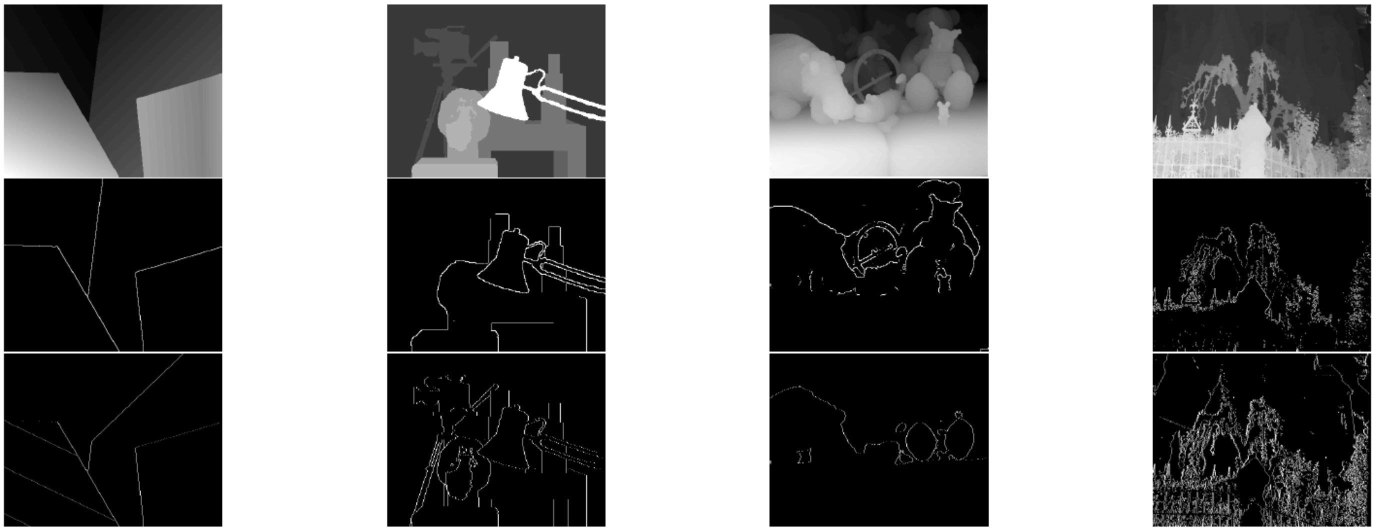


Fig. 11. Illustration of depth images (first row), depth edges preserved by depth coding algorithm (second row), and GBR connection positions (third row), for the "Venus," "Tsukuba," "Couch" and "Mansion" multiview datasets of Table I.

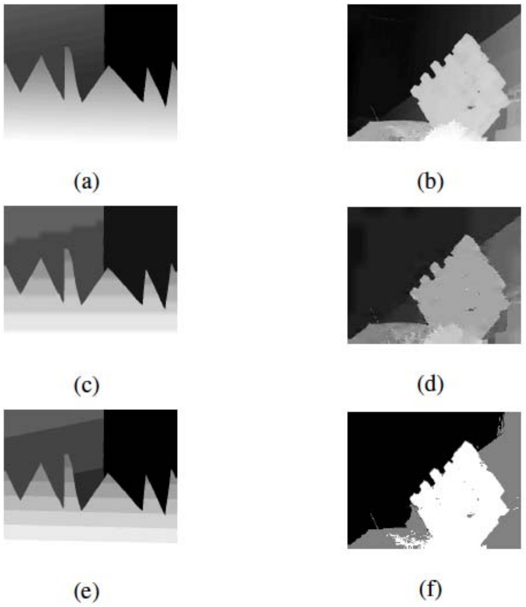


Fig. 12. Geometry images for "Sawtooth" (left) and "Statue" (right) sequences. Subfigures (a) and (b) are the original depth maps. Subfigures (c) and (d) are the depth maps coded with edge-adaptive wavelet (EAW) based coder [24], while (e) and (f) are geometry images extracted from our GBR. In these visual examples, the geometry coding rate of EAW is equal to the rate of our GBR (30 kb for "Sawtooth" and 10 kb for "Statue").

is no longer carried out with compressed depth maps, but with connections between particular pixels positions. We show here the shape and the behavior of the geometry signal provided by the GBR. For different datasets, summarized in Table I, we build the GBR for two views (a reference and a predicted one). We deduce the disparity maps from the GBR connections using Algorithm 3. The experiments are repeated for two different distances between the views (*i.e.*, different prediction complexity). The results are shown in Fig. 13 for four datasets. We see that the GBR geometry provides a simplified version of the depth map. For example, the gradient of depth in the "Venus" depth image becomes a set of steps parallel to the camera plane. The results also show that GBR adapts its level of precision to the complexity of the prediction. For example, the "Sawtooth" geometry is represented by six values while the geometry is only described by two values for the "Couch" image set. Moreover, looking at the geometry images for the prediction of View 2 and View 3, we observe that when the distance increases, GBR increases the amount of geometry information used in order to guarantee a good prediction quality.

In Fig. 14, we show the performance of the geometry compression scheme as a function of the distance δ between the reference and synthesized views. For every distance δ, we plot the RD comparison (in the format of Fig. 10), and we pick the minimum rate for each method to achieve a PSNR that is closer than 0.05 dB to the optimal one. The evolution of these rates is summarized in Fig. 14. We see that for low distance, our GBR is not that competitive with the standards methods. This is due to the fact that the error artifacts introduced in the compressed depth maps have a very small influence on the synthesized view quality. However, when distance increases, these coding artifacts become very important (they are indeed increasing with the baseline distance). At large distances, GBR that precisely controls the error of its projection becomes more efficient than the baseline depth compression methods.

In this experimental section, we have demonstrated that our GBR solution describes the geometry in a compact form, that competes with the most recent standards. In addition, it presents some interesting properties, such as adapting its level of precision to the complexity of the view synthesis to be done, which overcomes one of the main limitations of depth-based compression methods.

VII. Conclusion

In this paper, we have proposed an alternative method to depth for multiview geometry representation. Using graphs to describe connections between pixels of different views, our method represents the true geometry in the scene and avoids the inter-view redundancies. At the same time, it increases the control on geometry compression artifacts in the

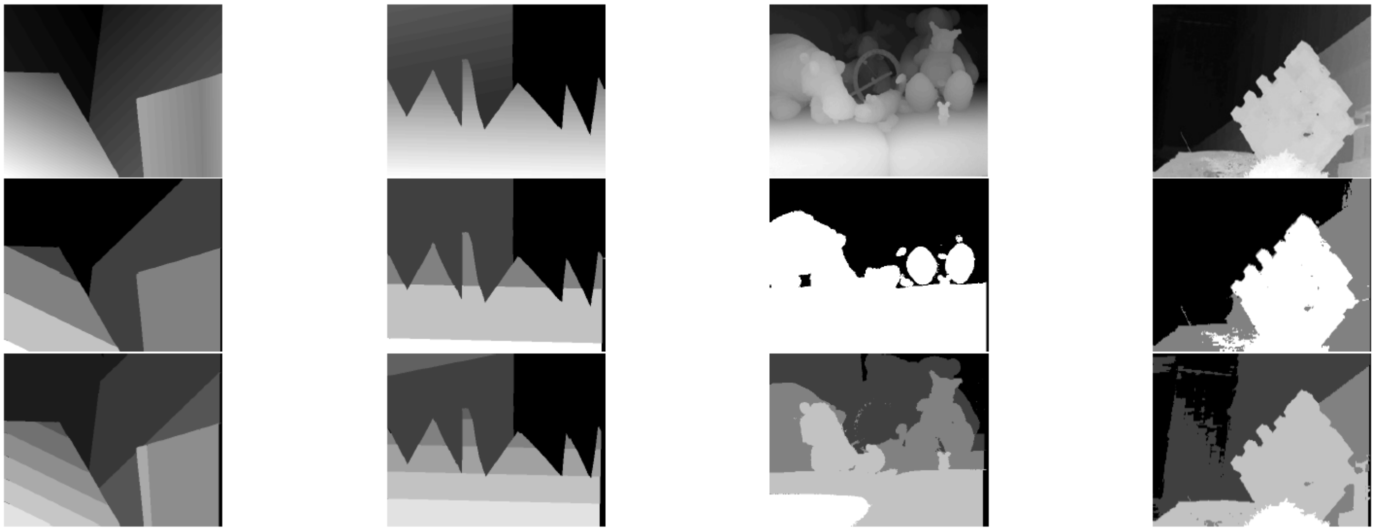
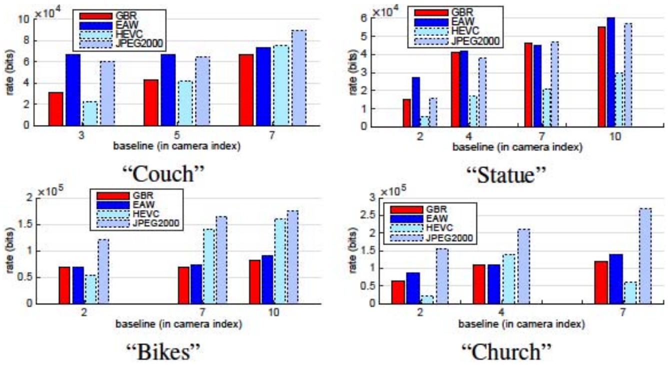


Fig. 13. Illustration of depth images (first row), GBR geometry for view 2 prediction (second row), and GBR geometry for view 3 prediction (third row), for the “Venus,” “Sawtooth,” “Couch” and “Statue” multiview datasets of Table I.



“Couch”

“Statue”

“Bikes”

“Church”

Fig. 14. Geometry rate to achieve a view synthesis quality at 0.05 dB from the optimal PSNR, as a function of the distance δ between reference and the synthesized view (in camera index).

reconstructed views. We have additionally proposed an efficient geometry coding algorithm based on this new graph-based representation and illustrated its potential in rate-distortion view synthesis performance compared to standard depth compression schemes. Future works will focus on the development of more effective coding strategies for both color and geometry coding in order to extend the performance of this promising GBR of multiview images. Moreover, introducing a variable rate control block might also be an interesting direction in order to enable the trade-off between quality and bit-rate for more flexibility.

Acknowledgment

The authors would like to thank Wei Hu, Gene Cheung and Vladan Velisavljevic for their help in the experimental comparisons with the edge-adaptive wavelet coding scheme.

References

[1] S. Foix, G. Alenya, and C. Torras, “Lock-in time-of-flight (ToF) cameras: A survey,” *IEEE Sensors J.*, vol. 11, no. 9, pp. 1917–1926, Sep. 2011.

[2] H.-Y. Shum, S. B. Kang, and S.-C. Chan, “Survey of image-based representations and compression techniques,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 11, pp. 1020–1037, Nov. 2003.

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

[4] J. Salvador and J. R. Casas, “Multi-view video representation based on fast Monte Carlo surface reconstruction,” *IEEE Trans. Image Process.*, vol. 22, no. 9, pp. 3342–3352, Sep. 2013.

[5] *Joint Multiview Video Model (JMVM)*, document ISO/IEC MPEG & ITU-T VCEG, Marrakesh, Morocco, Jan. 2007.

[6] J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum, “Plenoptic sampling,” in *Proc. 27th Annu. Conf. Comput. Graph. Interact. Techn.*, 2000, pp. 307–318.

[7] S.-Y. Kim and Y.-S. Ho, “Mesh-based depth coding for 3D video using hierarchical decomposition of depth maps,” in *Proc. IEEE Int. Conf. Image Process.*, San Antonio, TX, USA, Sep./Oct. 2007, pp. V-117–V-120.

[8] P. Merkle, A. Smolic, K. Müller, and T. Wiegand, “Multi-view video plus depth representation and coding,” in *Proc. IEEE Int. Conf. Image Process.*, San Antonio, TX, USA, Sep./Oct. 2007, pp. I-201–I-204.

[9] S. Yea and A. Vetro, “View synthesis prediction for multiview video coding,” *EURASIP J. Signal Process., Image Commun.*, vol. 24, nos. 1–2, pp. 89–100, 2009.

[10] D. Tian, P.-L. Lai, P. Lopez, and C. Gomila, “View synthesis techniques for 3D video,” *Proc. SPIE*, vol. 7443, p. 74430T, Sep. 2009.

[11] K. Müller *et al.*, “3D high-efficiency video coding for multi-view video and depth data,” *IEEE Trans. Image Process.*, vol. 22, no. 9, pp. 3366–3378, Sep. 2013.

[12] W.-S. Kim, A. Ortega, P. Lai, D. Tian, and C. Gomila, “Depth map distortion analysis for view rendering and depth coding,” in *Proc. 16th IEEE Int. Conf. Image Process.*, Cairo, Egypt, Nov. 2009, pp. 721–724.

[13] H. Yuan, S. Kwong, J. Liu, and J. Sun, “A novel distortion model and Lagrangian multiplier for depth maps coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 3, pp. 443–451, Mar. 2014.

[14] A. Secker and D. Taubman, “Highly scalable video compression with scalable motion coding,” *IEEE Trans. Image Process.*, vol. 13, no. 8, pp. 1029–1041, Aug. 2004.

[15] M. A. Agostini-Vautard, M. Cagnazzo, M. Antonini, G. Laroche, and J. Jung, “A new coding mode for hybrid video coders based on quantized motion vectors,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 7, pp. 946–956, Jul. 2011.

[16] Y. Morvan, D. Farin, and P. H. N. de With, “Joint depth/texture bit-allocation for multi-view video compression,” in *Proc. Picture Coding Symp. (PCS)*, 2007, pp. 265–268.

[17] Q. Wang, X. Ji, Q. Dai, and N. Zhang, “Free viewpoint video coding with rate-distortion analysis,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 6, pp. 875–889, Jun. 2012.

[18] G. Cheung, V. Velisavlevic, and A. Ortega, “On dependent bit allocation for multiview image coding with depth-image-based rendering,” *IEEE Trans. Image Process.*, vol. 20, no. 11, pp. 3179–3194, Nov. 2011.

[19] B. Rajei, T. Maugey, H.-R. Pourreza, and P. Frossard, “Rate-distortion analysis of multiview coding in a DIBR framework,” *Ann. Telecommun.*, vol. 68, nos. 11–12, pp. 627–640, 2013.

[20] K. Müller, P. Merkle, G. Tech, and T. Wiegand, "3D video coding with depth modeling modes and view synthesis optimization," in *Proc. Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC)*, Hollywood, CA, USA, Dec. 2012, pp. 1–4.

[21] T. Maugey, Y.-H. Chao, A. Gadde, A. Ortega, and P. Frossard, "Luminance coding in graph-based representation of multiview images," in *Proc. IEEE Int. Conf. Image Process.*, Paris, France, Oct. 2014, pp. 130–134.

[22] P. Merkle, A. Smolic, K. Müller, and T. Wiegand, "Efficient prediction structures for multiview video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 11, pp. 1461–1473, Nov. 2007.

[23] Y. Morvan, D. Farin, and P. H. N. de With, "Depth-image compression based on an R-D optimized quadtree decomposition for the transmission of multiview images," in *Proc. IEEE Int. Conf. Image Process.*, San Antonio, TX, USA, Sep./Oct. 2007, pp. V-105–V-108.

[24] M. Maitre and M. N. Do, "Depth and depth–color coding using shape-adaptive wavelets," *J. Vis. Commun. Image Represent.*, vol. 21, nos. 5–6, pp. 513–522, Jul./Aug. 2008.

[25] S. Liu, P. Lai, D. Tian, and C. W. Chen, "New depth coding techniques with utilization of corresponding video," *IEEE Trans. Broadcast.*, vol. 57, no. 2, pp. 551–561, Jun. 2011.

[26] G. Cheung, W.-S. Kim, A. Ortega, J. Ishida, and A. Kubota, "Depth map coding using graph based transform and transform domain sparsification," in *Proc. IEEE Int. Workshop Multimedia Signal Process.*, Hangzhou, China, Oct. 2011, pp. 1–6.

[27] I. Daribo, G. Cheung, and D. Florencio, "Arithmetic edge coding for arbitrarily shaped sub-block motion prediction in depth video compression," in *Proc. 19th IEEE Int. Conf. Image Process.*, Orlando, FL, USA, Sep. 2012, pp. 1541–1544.

[28] A. Gelman, P. L. Dragotti, and V. Velisavljević, "Multiview image coding using depth layers and an optimized bit allocation," *IEEE Trans. Image Process.*, vol. 21, no. 9, pp. 4092–4105, Sep. 2012.

[29] U. Takyar, T. Maugey, and P. Frossard, "Extended layered depth image representation in multiview navigation," *IEEE Signal Process. Lett.*, vol. 21, no. 1, pp. 22–25, Jan. 2014.

[30] F. Maninchedda, M. Pollefeys, and A. Fogel, "Efficient stereo video encoding for mobile applications using the 3D+F codec," in *Proc. IEEE 3DTV-Conf.*, Zürich, Switzerland, Oct. 2012, pp. 1–4.

[31] T. Maugey, A. Ortega, and P. Frossard, "Graph-based representation and coding of multiview geometry," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Vancouver, BC, Canada, May 2013, pp. 1325–1329.

[32] T. Maugey, A. Ortega, and P. Frossard, "Multiview image coding using graph-based approach," in *Proc. IEEE 11th Workshop 3D Image/Video Technol. Appl. (IVMSP)*, Seoul, Korea, Jun. 2013, pp. 1–4.

[33] F. Ono, W. Rucklidge, R. Arps, and C. Constantinescu, "JBIG2-the ultimate bi-level image coding standard," in *Proc. Int. Conf. Image Process.*, Vancouver, BC, Canada, Sep. 2000, pp. 140–143.

[34] J.-R. Ohm, D. Rusanovsky, A. Vetro, and K. Müller, "Joint collaborative team on 3D video coding extension develoment," Tech. Rep. JCT3V-B1006, Oct. 2012.

[35] K. Müller, A. Smolic, K. Dix, P. Merkle, and T. Wiegand, "Coding and intermediate view synthesis of multiview video plus depth," in *Proc. 16th IEEE Int. Conf. Image Process.*, Cairo, Egypt, Nov. 2009, pp. 741–744.

[36] M. Tanimoto, "FTV: Free-viewpoint television," *Signal Process., Image Commun.*, vol. 27, no. 6, pp. 555–570, Jul. 2012.

[37] *JPEG 2000 Final Committe Draft Version 1.0*, document ISO/IEC FCD 15444-1, 2000.

[38] C. Fehn, "Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV," *Proc. SPIE*, vol. 5291, pp. 93–104, May 2004.

Thomas Maugey (S'09–M'11) received the degree from the École Supérieure d'Electricité, Supélec, Gif-sur-Yvette, France, in 2007, the M.Sc. degree in fundamental and applied mathematics from Supélec and Universit Paul Verlaine, Metz, France, in 2007, and the Ph.D. degree in image and signal processing from TELECOM ParisTech, Paris, France, in 2010. From 2010 to 2014, he was a Post-Doctoral Researcher with the Signal Processing Laboratory, Swiss Federal Institute of Technology, Lausanne, Switzerland. Since 2015, he has been a Research Scientist in the team-project SIROCCO with INRIA, Rennes, France.

His research interests include monoview and multiview distributed video coding, 3D video communication, data representation, video compression, network coding, and view synthesis.

Antonio Ortega (F'07) received the Telecommunications Engineering degree from the Universidad Politecnica de Madrid, Madrid, Spain, in 1989, and the Ph.D. degree in electrical engineering from Columbia University, New York, NY, USA, in 1994.

He joined the Electrical Engineering Department, University of Southern California (USC), in 1994, where he is currently a Professor. At Columbia University, he was supported by a Fulbright scholarship. He has served as an Associate Chair of EE-Systems and the Director of the Signal and Image Processing Institute at USC. His recent work is focusing on distributed compression, multiview coding, error tolerant compression, wavelet-based signal analysis, information representation in wireless sensor networks, and graph signal processing. His work at USC is being funded by agencies, such as NSF, NASA, and DOE, and companies, such as HP, Samsung, and Chevron or Texas Instruments. Almost 40 Ph.D. students have completed their Ph.D. thesis under his supervision at USC. His work has led to over 300 publications in international conferences and journals, and several patents. His research interests are in the areas of multimedia compression, communications, and signal analysis. He is a member of the Association for Computing Machinery and the Asia Pacific Signal and Information Processing Association. He was a recipient of the NSF CAREER Award, the 1997 IEEE Communications Society Leonard G. Abraham Prize Paper Award, the IEEE Signal Processing Society 1999 Magazine Award, the 2006 *EURASIP Journal of Advances in Signal Processing* Best Paper Award, the 2011 ICIP Best Paper Award, and a best paper award at 2012 Globecom. He has been the Chair of the Image and Multidimensional Signal Processing Technical Committee and a member of the Board of Governors of the IEEE Signal Processing Society (SPS). He was the Technical Program Co-Chair of MMSP 1998, ICME 2002, ICIP 2008, and PCS 2013. He has been an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE SIGNAL PROCESSING LETTERS, the *EURASIP Journal on Advances in Signal Processing*, and the *IEEE Signal Processing Magazine*. He is an Inaugural Editor-in-Chief of the *APSIPA Transactions on Signal and Information Processing* (APSIPA and Cambridge University Press, 2012). He is the Chair of the SPS Big Data Special Interest Group. He was a plenary speaker at ICIP 2013.

Pascal Frossard (S'96–M'01–SM'04) received the M.S. and Ph.D. degrees in electrical engineering from the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, in 1997 and 2000, respectively. From 2001 to 2003, he was a member of the Research Staff with the IBM T. J. Watson Research Center, Yorktown Heights, NY, USA, where he was involved in media coding and streaming technologies. Since 2003, he has been a Faculty Member at EPFL, where he is currently the Head of the Signal Processing Laboratory. His research interests include image representation and coding, visual information analysis, distributed image processing and communications, and media streaming systems.

He was a recipient of the Swiss NSF Professorship Award in 2003, the IBM Faculty Award in 2005, the IBM Exploratory Stream Analytics Innovation Award in 2008, and the IEEE TRANSACTIONS ON MULTIMEDIA Best Paper Award in 2011. He was the General Chair of the IEEE ICME 2002 and Packet Video 2007. He was the Technical Program Chair of the IEEE ICIP 2014 and EUSIPCO 2008, and a member of the Organizing or Technical Program Committees of numerous conferences. He was an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING (2010–2013), the IEEE TRANSACTIONS ON MULTIMEDIA (2004–2012), and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (2006–2011). He was the Chair of the IEEE Image, Video and Multidimensional Signal Processing Technical Committee (2014–2015). He has been an elected member of the IEEE Visual Signal Processing and Communications Technical Committee since 2006 and the IEEE Multimedia Systems and Applications Technical Committee since 2005. He served as the Steering Committee Chair (2012–2014) and the Vice Chair (2004–2006) of the IEEE Multimedia Communications Technical Committee, and a member of the IEEE Multimedia Signal Processing Technical Committee (2004–2007).