Livrable D4.2 of the PERSEE project : Représentation et codage 3D - Rapport intermédiaire - Définitions des softs et architecture

Marcus Barkowsky, Emilie Bosc, Marco Cagnazzo, Josselin Gautier, Christine Guillemot, Vincent Jantet, Olivier Le Meur, Luce Morin, Fabien Racapé, Vincent Ricordel

Projet PERSEE
Texture analysis and synthesis
n° ANR-09-BLAN-0170

Livrable **D4.2** *1/11/2011*

Del 4.2: Représentation et codage 3D - Rapport intermédiaire - Définitions des softs et architecture

---

Marcus	BARKOWSKY	IRCCYN
Emilie	BOSC	INSA
Marco	CAGNAZZO	Telecom Paristech
Josselin	GAUTIER	INSA
Christine	GUILLEMOT	INRIA
Vincent	JANTET	INRIA
Olivier	LE MEUR	INRIA
Luce	MORIN	INSA
Fabien	RACAPE	INSA/INRIA
Vincent	RICORDEL	IRCCYN

Contents

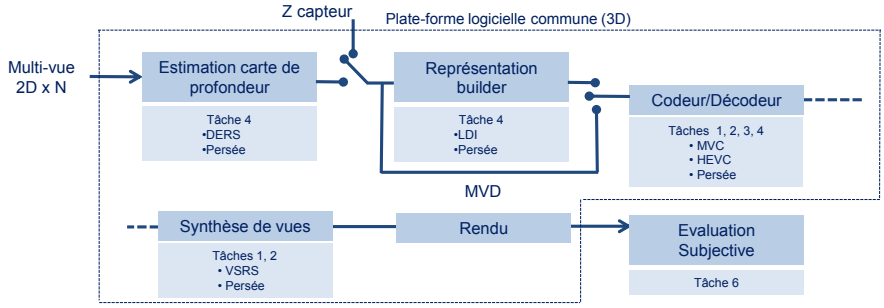


Figure 1: 3D video coding architecture.

1 Architecture for 3D representation and coding

The general architecture of the 3D video coding scheme is described in figure 1. The goal of this architecture is to evaluate the methods and algorithms developed within the Persee project, and compare them to state of the art coding schemes. Subjective evaluations will be performed on the tested methods, to evaluate their pertinence with regard to rendered visual quality.

In a first phase, partners will exchange input/output files in order to perform evaluation of methods without the need for software integration.

Several types of experiments are planed:

- build a reference 3Dvideo coding chain, in order to compare the proposed methods to state of the art coding schemes. The state of the art codecs for MVD data are H264 and 3DHTM, which is currently under development by the Mpeg consortium.
- perform objective/subjective evaluation of depth compression algorithms which have been developed within the Persée consortium and which are described in the following sections of the deliverable.
- exploit perceptual data extracted from the video in the coding and rendering units, such as visual attention or saliency. The scheme presented in figure 2 exploits such information for introducing blur in videos, both for rate reduction and visual quality enhancement.

The following sections provide some tools developed by PERSEE contributors, in order to build and evaluate the 3D video coding scheme.

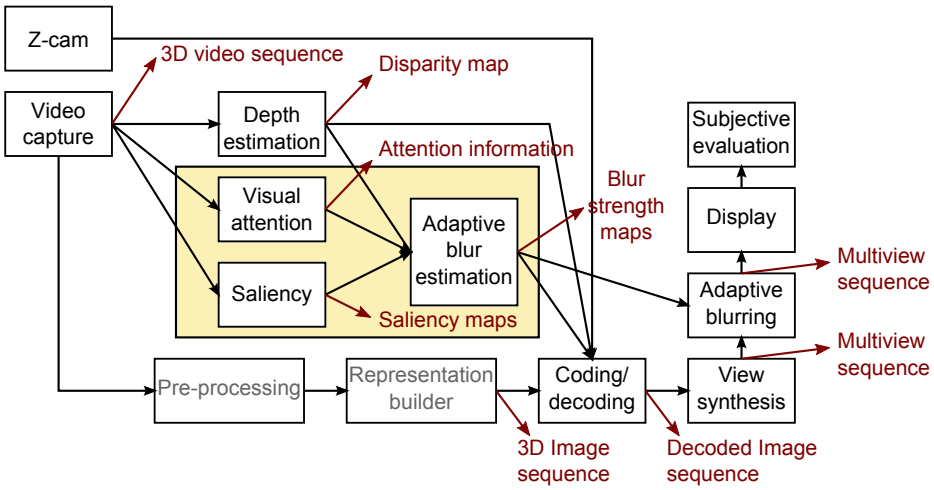


Figure 2: Blur-based example of framework.

2 Disparity map coding by Don't Care Regions

In the image+depth multiview video format, each view $n = 1, \ldots, N$ is represented with a texture and a depth (or, equivalently, disparity) map. Given $\mathbf{v}_n$, $\mathbf{v}_{n+1}$ and $\mathbf{d}_n$, $\mathbf{d}_{n+1}$ the texture and disparity maps at views n and $n+1$, respectively, it is possible to synthesize any intermediate view $\mathbf{v}_k$, $k \in [n, n+1]$ using a depth image-based rendering (DIBR) algorithm. Essentially, any DIBR algorithm synthesizes pixel values in $\mathbf{v}_k$ by properly translating pixels from views n and $n+1$, according to the disparity information contained in the disparity maps, and by possibly resorting to some inpainting technique to fill holes due to occlusions in one of the neighboring views. In this context, the goal of multiview video coding is maximizing the quality of the view synthesized at the decoder, provided that the total number of bits required to encode and transmit $\mathbf{v}_n$, $\mathbf{d}_n$, $\mathbf{v}_{n+1}$ and $\mathbf{d}_{n+1}$ has to be within a predetermined bit budget. This is in general a joint coding problem, since the rate-distortion (RD) performance of texture/disparity at view n will depend on the RD characteristic of the other three signals — assuming that the DIBR algorithm is given.

Instead, in the following discussion we focus on the simpler task of coding disparity maps independently. That is, given uncompressed textures $\mathbf{v}_n$ and $\mathbf{v}_{n+1}$, we want to encode independently $\mathbf{d}_n$ and $\mathbf{d}_{n+1}$ only. This simplified perspective is interesting as it allows as to focus on a basic difference between disparity and conventional video coding. In disparity video coding, the rate to be minimized is the one required to encode $\mathbf{d}_n$ and $\mathbf{d}_{n+1}$. However, coding quality is measured on the *synthesized view* $\mathbf{v}_k$. This leads to two conclusions: 1) the quality metric used in disparity coding should capture the distortion in disparity compensation produced by coarse quantization of disparity maps; and 2) it is possible to give up fidelity and reduce bit rate in the reconstructed disparity maps if this does not affect too much the synthesized view.

2.1 Test of a geometric distortion metric on synthesized views

As mentioned above, at the basis of DIBR there is disparity compensation, i.e., pixels in the view $\mathbf{v}_k$ to be synthesized are obtained by properly displacing pixels in views n and $n+1$. Therefore, coding errors in the disparity maps correspond to displacement errors in the synthesized view. Given the different nature of these two source of distortion, we aim at describing displacement errors through a geometric distortion measure.

We consider the geometric distortion measure (GM) proposed in [11]. This metric has been originally proposed in the watermarking field to quantify the perceptual impact of geometric manipulations aimed at hindering the extraction of a watermark embedded into an image. In a nutshell, this approach consists in projecting the displacement vectors (of a dense displacement field) along the directions parallel to the gradients of the local image structures, with the rationale that only this displacement component is going to be visible to users. The intensity and the direction of local image structures is obtained through Gabor filters. By averaging, for several directions, a combination of local image structure and intensity (in a given direction) of the displacement vectors, it is possible to predict the mean opinion score (MOS) a human

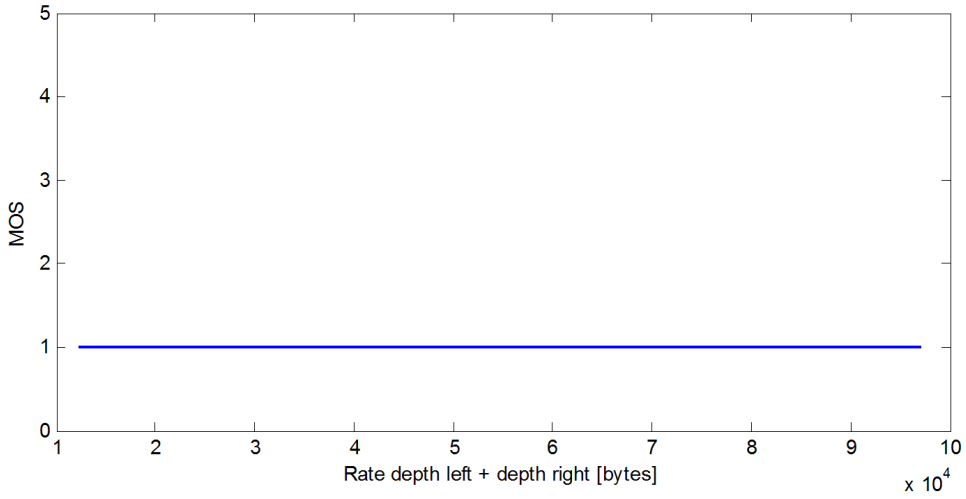


Figure 3: MOS vs. bit rate for coding disparity in the *Teddy* image. The synthesized view $\hat{\mathbf{v}}_4$ is obtained by DIBR on compressed disparities $\hat{\mathbf{d}}_2$ and $\hat{\mathbf{d}}_6$. The rate term here refers to the total bits needed to encoded $\mathbf{d}_2$ and $\mathbf{d}_6$.

observer would give. When we consider the disparity map coding detailed above, a natural question arises: can we use the GM metric to predict the visual impairment introduced by errors in the coding of disparity?

In order to answer this question, we tested the sensitivity of GM to synthesis errors induced by quantization errors in the disparity maps. To this end, we compressed disparity maps $\mathbf{d}_2$ and $\mathbf{d}_6$ of views 2 and 6, respectively, for the *Teddy* image in the Middlebury Dataset [1], using JPEG with different quality factors. We denote with $\hat{\mathbf{d}}_2$ and $\hat{\mathbf{d}}_6$ the reconstructed *noisy* disparities at the decoder. Then, we synthesized $\hat{\mathbf{v}}_4$ using $\hat{\mathbf{d}}_2$ and $\hat{\mathbf{d}}_6$, by means of a simple DIBR algorithm which fills holes in the synthesized view by copying the closest pixel value from either view 2 or 6. In order to single out the contribution to distortion due to compression only, we compared $\hat{\mathbf{v}}_4$ with a groundtruth image $\mathbf{v}_4$ generated using uncompressed $\mathbf{d}_2$ and $\mathbf{d}_6$. We then predict the distortion between $\mathbf{v}_4$ and $\hat{\mathbf{v}}_4$ using GM, which yields a prediction of the MOS. A rate/MOS curve is shown in Figure 3, where it is clearly visible that the GM metric is absolutely not able to provide meaningful prediction of the MOS for the synthesized view. Similar results have been obtained also for the *Doll* image [1].

From these results we can infer that GM is not a suitable metric for synthesis error produced by noisy disparity maps. In fact, even at the maximum JPEG quality, the rounding and truncation entailed in the coding process make the reconstructed disparities being slightly different from the groundtruth. This produces isolated errors in the synthesized view, which however have a very strong impact on the computation of GM. Consider also that GM has been designed, tuned and validated with subjective

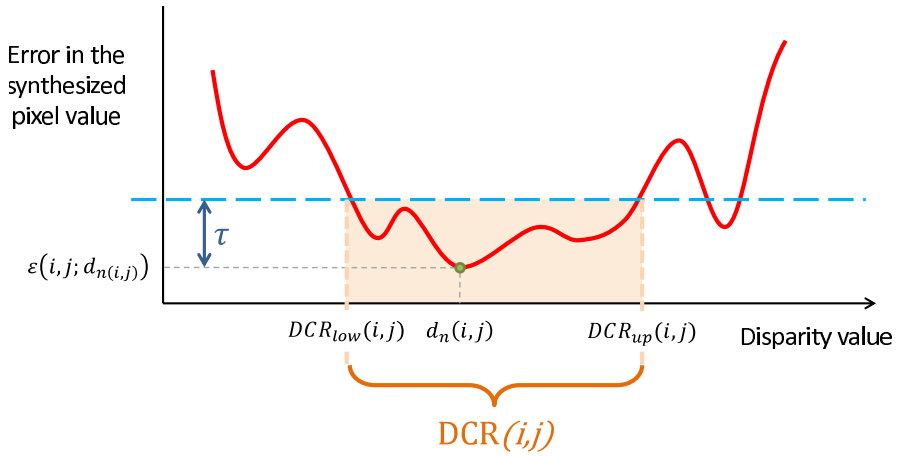


Figure 4: Definition of DCR for a given threshold τ.

tests performed on images corrupted with smooth displacement errors, so there are no guarantees that it can work in the context of view synthesis.

2.2 Increasing coding efficiency of disparity maps through Don't Care Regions

Differently from conventional video coding, where the signal to be compressed is the one that has to be displayed to users, the coding of disparity map video has to be optimized for the display of synthesized views. This introduces a certain degree of freedom in the coding of disparity which is not present in texture coding. For example, in very smooth regions of the frame, having even a relatively large disparity error would not lead necessarily to a large error in the synthesized view. This concept can be formalized and embedded into a state-of-the-art video encoder in order to increase the coding efficiency of disparity maps.

2.2.1 Definition of Don't Care Regions (DCR)

A pixel $v_n(i,j)$ in texture map n, with associated disparity value $d_n(i,j)$, can be mapped into a corresponding pixel of the view $n+1$ through a view synthesis function $s(i,j;d_n(i,j))$. In the simplest case where the views are captured by horizontally aligned cameras, $s(i,j;d_n(i,j))$ corresponds to a displacement in the x direction by $d_n(i,j)$, that is:

$$s(i,j;d_n(i,j)) = v_n(i,j+d_n(i,j)). \tag{1}$$

We define the view synthesis error, $\varepsilon(i,j)$, as the absolute error between the synthesized pixel $s(i,j;d_n(i,j))$ and the actual pixel value $v_{n+1}(i,j)$ in $\mathbf{v}_{n+1}$. That is:

$$\varepsilon(i,j;d_n(i,j)) = \left|s(i,j;d_n(i,j)) - v_{n+1}(i,j)\right|. \tag{2}$$

If $\mathbf{d}_n$ is compressed, the reconstructed value $\tilde{d}_n(i,j)$ employed for view synthesis may differ from $d_n(i,j)$ by an amount $e(i,j) = \tilde{d}_n(i,j) - d_n(i,j)$, resulting in a (generally larger) view synthesis error $\varepsilon(d_n(i,j) + e(i,j))$. We define the Don't Care Region $\mathrm{DCR}(i,j) = [\mathrm{DCR}_{low}(i,j), \mathrm{DCR}_{up}(i,j)]$ as the *smallest* interval of disparity values containing the groundtruth disparity $d_n(i,j)$, such that the view synthesis error for any point of the interval is smaller than $\varepsilon(i,j;d_n(i,j))+\tau$, for a given threshold τ. The definition of DCR is illustrated in Figure 4. Note that DCR intervals are defined *per pixel*, thus giving a very precise information about how much error can be tolerated in the disparity maps.

2.3 Use of DCR for increasing coding efficiency of disparity maps

Having a per pixel DCR enables a higher degree of freedom in the encoding of disparity maps, which can be compressed with a larger distortion (i.e., lower rate), without actually compromising too much the quality of the displayed synthesized view. Specifically, DCR can affect three aspects of disparity coding:

- Motion estimation;
- Mode decision;
- Coding of residuals.

Motion estimation During motion estimation, the encoder search for each macroblock or sub-block $\mathcal{B}$ a corresponding region in a reference frame which minimizes the Lagrangian cost function

$$D_{\mathrm{MV}} + \lambda_{\mathrm{MV}} R_{\mathrm{MV}}, \tag{3}$$

where R_{MV} is the bit rate associated to the cost of representing the motion vector and the prediction residual, and λ_{MV} is a Lagrange multiplier. The term D_{MV} is the sum of absolute differences (SAD) between the pixels in the current block and their predictor. We can add more degrees of freedom to motion estimation considering that the term D in (3) can be discounted by the per pixel DCR's. To this end, we manipulate the values of prediction residuals $r(i,j)$ involved in the computation of SAD in (3) according to the following soft-thresholding function:

$$r' = \begin{cases} r(i,j) - (\mathrm{DCR}_{up} - d_n(i,j)) & \text{if } r(i,j) > \mathrm{DCR}_{up} - d_n(i,j), \\ r(i,j) - (\mathrm{DCR}_{low} - d_n(i,j)) & \text{if } r(i,j) < \mathrm{DCR}_{low} - d_n(i,j), \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

The discounted SAD is then computed as

$$D'_{\mathrm{MV}} = \sum_{(i,j)\in\mathcal{B}} r'(i,j)) \tag{5}$$

and substituted to D in (3).

Mode decision In the mode decision phase, the rate-distortion cost of each coding mode (Inter with all possible sub-block partitions, Intra with 16×16, 4×4 or 8×8, Skip) is evaluated in order to select the optimal modes for a given macroblock and for its partitions. In order to find the optimal mode configuration, the encoder minimizes a function similar to (3):

$$D_{\text{MD}} + \lambda_{\text{MD}} R_{\text{MD}}, \tag{6}$$

where D_{MD} is the SAD or SSE (sum of squared differences) distortion entailed by each coding mode, R_{MD} is the associated bit rate and λ_{MD} is a Lagrangian multiplier. As before, for each coding mode we can change the term D_{MD} to D'_{MD} by applying (4) to the residuals obtained with that coding mode.

Coding of prediction residuals Once motion vectors and coding modes have been computed, the prediction residuals are transformed and quantized before being entropy coded and written in the bitstream. According to DCR definition, it is not necessary to reconstruct the true groundtruth disparity at each pixel. Instead, any value that is inside the DCR would be acceptable in terms of view synthesis distortion. However, in this case altering the values of prediction residuals $r(i,j)$ to $r'(i,j)$ using equation (4) is not optimal from the coding point of view, since this soft-thresholding heuristic on prediction residuals in the pixel domain does not give any guarantees of reducing the bit rate for disparity coding. A more correct approach (originally proposed in [9] for the case of still images) would consist in performing transform-domain sparsification of prediction residuals, subject to the DCR constraints in the pixel domain. This is left to future work.

We implemented the modified motion estimation and mode decision in the JM reference software implementation (v. 18) of H.264/AVC. Figure 5 shows the rate-distortion curve obtained by coding 5 frames of the *Kendo* video sequence. With $\tau = 5$ we have a gain in PSNR at low rates of 0.4 dB and a rate saving of almost 60%, while with $\tau = 7$, $\Delta\text{PSNR} = 0.36$ and $\Delta\text{Rate} = 67\%$ using Bjontegaard metric. These results are particularly encouraging since most of the gain is obtained mainly through a higher efficiency in mode decision and motion estimation, with e.g. the number of SKIP macroblocks almost doubling when using our method. We expect even a larger gain when also the transform domain sparsification of prediction residuals will be taken into consideration *jointly* with mode decision and motion estimation.

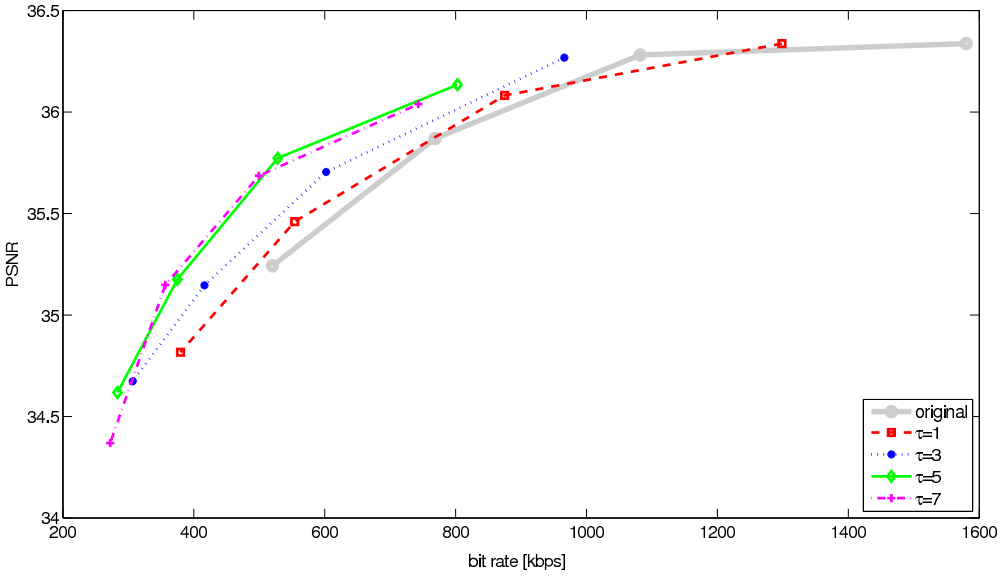


Figure 5: RD performance on 5 frames of the *Kendo* sequence, with a modified H.264/AVC encoder to embed DCR, for several values of τ. The bit rate refers to the total rate required to encode disparity maps of view 3 and 5. The PSNR is computed between the synthesized view $\hat{\mathbf{v}}_4$ and the groundtruth center view $\mathbf{v}_4$.

3 Efficient Depth Map Compression based on Lossless Edge Coding and Diffusion

3.1 Introduction

3DTV and FVV are promising technologies for the next generation of 3D entertainment. To this end, the MultiView plus Depth (MVD) format has been developed. From a collection of multi-view video sequences captured synchronously from different cameras at different locations, a view synthesis or rendering method can generate new viewpoints. The first version of the dedicated MVD sequence encoder, so-called "MVC", was based on block transforms of depth maps. As argued by [36], the deblocking of depth maps is today one of the most important pre- and postprocessing task for the MVD representation.

Depth maps have two main features that must be preserved but can also be relied on for efficient compression.The first one is the sharpness of edges, depicting the depth on the exact borders of the scene objects. Distortions on edges during the encoding step would cause highly visible degradations on the synthesized views, that require depth map post-processing. The second one comes from the general smooth surface properties of objects we are measuring the depth on. Based on these observations, Merkle et al. [23] proposed a "Wedgelet" signal decomposition method (itself based on "Platelet"). The smooth regions of the depth maps are approximated using piecewise-linear functions separated by straight lines. A quadtree decomposition divides the image into variable-size blocks, each of them being approximated by a modeling function. The refinement of the quadtree is then optimized in the rate-distortion sense.

In this context, we also observed that these smooth surfaces can be efficiently approximated by interpolating the luminance values located at their boundaries, instead of using models based on piecewise-linear functions where the coefficients need to be encoded [23]. To this end, we can observe that depth maps share similarity to cartoon-images. Mainberger et al. [20] proposed a dedicated cartoon-image encoder, that -in low bitrate conditions- beats the JPEG-2000 standard. After a Canny edge detection, the edge locations are encoded with a lossless bi-level encoder, and the adjacent edge pixel values are lossy quantized and subsampled. At the decoding stage, an homogeneous diffusion is used to interpolate the inside unknown areas from lossy decoded edges. Indeed, the demonstrated performances -while beating state of the art codecs- reach the limit of 30dB. We revisited this edge-based compression method by proposing improvements to fit the high quality, low bitrate, and specific requirements of depth maps. Finally, we greatly increase the diffusion-based depth map encoding performance, which might be generalized to all kinds of images. In the next section, the encoding process is described. In section III the new decoding and diffusion methods are explained. Results, performances and comparison with state-of-the-art methods are given in Section IV. Conclusions are then drawn in section V.

3.2 Encoding

The encoding is a 3 step process: first is detection of edges, then encoding of the edge location and finally encoding of the edge, border and seed pixel values.

3.2.1 Edge detection

Different operators exist to extract the contour of an image. An optimal edge detector should provide:

- a good detection: the algorithm should find as much real edges as possible.
- a good localization: the edges should be marked as edges as close as possible to the real edges.
- a good robustness: as much as possible, the detector should be insensitive to noise.

In our context of depth map edge coding, several requirements are added. The quality of reconstruction by diffusion should be maximized, while minimizing the number of edges that will be needed for later diffusion. In order to maximize the quality of edges for diffusion, the localization of contours should be quasi-perfect (see section III for explanation). The detection of contours should be good but avoiding an over-detection. Up to a certain limit, weak contours (i.e. with a low gradient) might be useless to the reconstruction and might unnecessarily increase the edge coding cost. Also, noisily detected pixels should be avoided for the same reason.

The Marr-Hildreth edge detector combined with Canny like hysteresis thresholding is used in [20], but suffers from error of localization at curved edges. The widely used Canny edge detector has also been benchmarked. It relies on a 5x5 gradient prefiltering to cope with noise before local maxima edge detection. This prefiltering step also makes this detector vulnerable to contour localization errors, as illustrated in Fig.6(c), where inexact selection of adjacent edge pixels lead to improper diffusion. Oppositely Sobel has the advantage of an accurate contour localization as shown in Fig.6(d) at the cost of noisy detection. Pixels with a bi-dimensional gradient amplitude larger than a threshold λ are extracted. To cope with the edge noise, contours c shorter than a certain value ($c < 14$) are excluded. Used with sharp depth maps, this gives well-localised contours.

3.2.2 Encoding the contour location

As in [20], a bi-level edge image containing the exact location of previously detected edges is first encoded using the *JBIG (Joint Bi-level Image Experts Group)* standard. This is a context-based arithmetic encoder enabling lossless compression of bi-level images. We use the JBIG-Kit [2], a free C implementation of the JBIG encoder and decoder. The progressive mode is disabled to reduce the required bitrate.

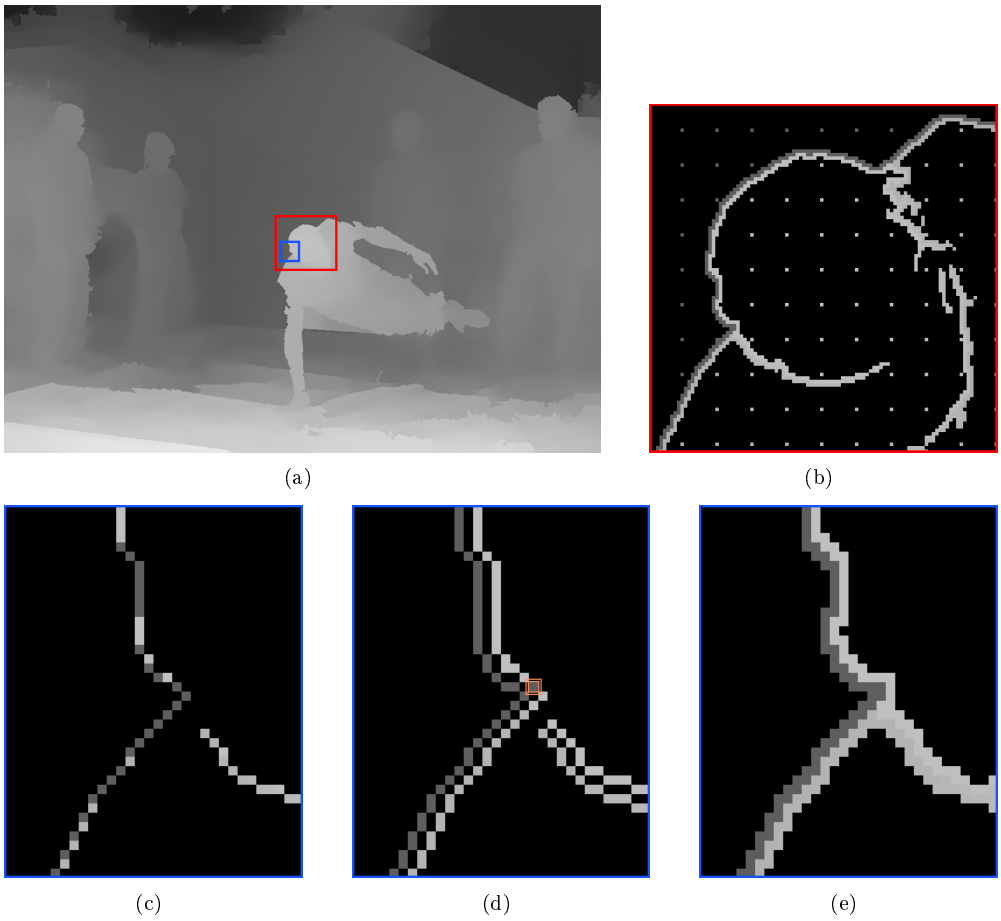


Figure 6: (a) A "Breakdancer" depth map, (b) the encoded and decoded Sobel edge and seed pixels (red selection on (a)), (c) the Canny edges (blue selection), (d) the selection of pixel values adjacent to Canny edges (c) as in [20], with an intruder edge pixel in orange that will lead to bad diffusion, (e) the proposed Sobel selection of edge pixel values, exactly located from both side of the frontier edge

3.2.3 Encoding the contour values

Once the edge pixel locations have been encoded, the pixel luminance values have also to be losslessly encoded following our initial requirements. The authors in [20] proposed to store the pixel values on both sides of the edge, instead of the pixel values lying on the edge itself. Indeed, for blurry contours, this might be valuable to interpolate the inner part of the edge and code the luminance values on both sides. However, with sharp depth maps, the pixel values lying directly on an edge, as illustrated in Fig.6(b), alternate between one side or another from this edge and couldn't be interpolated correctly.

With the Sobel edge detection not thinned to a single edge pixel, we ensure to retain at least one pixel value from each side of the frontier edge (Fig.6(d)).

We keep the idea of storing the pixel values by their order of occurrence along the edge to minimize signal entropy. A path with fix directional priorities (E, S, W, N, NE, NE SE, SW and NW) is used. As the intrinsic properties of pixels along an edge or "isophote" are their small luminance variation, then we propose to compute the differential values of edge pixels in a Differential Pulse Code Modulation (DPCM) way. From this optimized path encoding method, the stream of DPCM values is then encoded with an arithmetic coder.

Additionally to these edges we also encode two kinds of information. The pixel values from the image border are stored to initiate the diffusion-based filling from borders. Inspired by the work of [4] on "dithering" for finding optimal data for interpolation, we propose to sparsely deploy, at regular intervals, some seeds of original depth pixels as shown in Fig.6(e). While having low overhead, we discovered that this helps accurate reconstruction by initializing the diffusion in large missing areas. As we will see, it additionally greatly speeds up the diffusion process.

Thus, these extra border and seed pixels are coded in DPCM and added to the differential edge values. The resulting file is thus composed of the payload of the JBIG data and of the arithmetic encoded bitstream of the DPCM edge, border, and seed pixel values.

3.3 Decoding and diffusion

3.3.1 Decoding contour location and pixel values

Once the edge location from JBIG payload is decoded, the edge pixel values are decoded and positioned following the same order in which they were encoded: the path along contour location respecting directional priorities.The border and seed values are also re-positioned following a predefined location.

3.3.2 Reconstructing the missing values by diffusion

We now have a sparse depth map containing only the edge, border and seed pixel values. An homogeneous diffusion-based inpainting approach is used to interpolate the missing data. This method is the simplest of the partial differential equations

(PDEs) diffusion method, and has the advantage of low computational complexity. It directly stems from the heat equation:

$$\begin{cases} I_{t=0} = \widetilde{I} \\ \frac{\delta I}{\delta t} = \Delta I \end{cases}$$

where $\widetilde{I}$ is the decoded edge image before diffusion that will constitute the Dirichlet boundaries of the equation. The diffused data then satisfies the Laplace equation $\Delta I = 0$. The diffusion process is run in a hierarchical manner, each diffusion step being in addition helped with seeds and appropriate initialization. These three improvements have been introduced in the classical diffusion approach to limit the number of iterations required to converge, hence to speed up the entire process:

Hierarchical diffusion A Gaussian pyramid is built from $\widetilde{I}$. The diffusion process is first performed on a lower level of the pyramid and the diffused values are then propagated to a higher level (3 levels are used and shown good performance). The propagation of the blurred version of the diffused pixel values from an lower level to an upper one helps to initialize the diffusion in unknown areas.

Middle range initialization On the highest level, instead of starting from unknown value of $\widetilde{I}$ set at 0, we propose to initialize unknown values to the half of the possible range: 128 for an 8 bit depth map. This facilitates and speeds up the process of diffusion by limiting the number of required iterations to converge.

Seeding As explained in section 3.2.3, some seeds are chosen from a regular pattern both to accelerate the diffusion process and to provide accurately initialized values in large unknown areas. Indeed, this definitely achieves a fast and accurate diffusion -with a gain of 10 dB- for a quasi-exact reconstruction of the depth map.

3.4 Experiments

3.4.1 Conditions

The performances of the proposed compression method are evaluated on an original resolution depth map from a Multiview Video-plus-Depth (MVD) sequence "Breakdancers" from Microsoft [45]. The depth maps were estimated through a color segmentation algorithm. The choice of this sequence is motivated by the presence of sharp edges of objects on depth maps. Most other raw MVD sequences might be suitable once they would be "sharpened" i.e. post-processed with a bilateral filter, as it is often required in practice [36].

3.4.2 Depth map quality evaluation

The reconstruction quality of our PDE-based method is investigated and compared with the JPEG and JPEG2000 compressed versions. First, to illustrate the difference

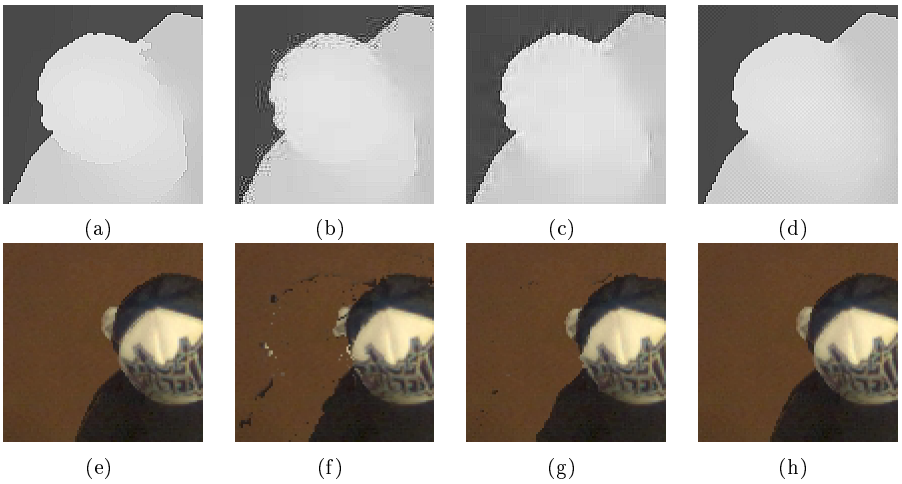


Figure 7: Upper row: zoom on the head of a dancer on original View#3 (V_3) depth map (a) highlights by comparison the ringing artifact on JPEG (b) and JPEG2000 (c). Our method (d) based on exact edges and homogeneous diffusion prevent this effect. (Depth maps have been sharpened for distortion visibility). Lower row: zoom on synthesized view V_4 without (e) or with JPEG (f), JPEG2000 (g) and our method (h) compressions at equal PSNR (43dB) referenced to (e).

of quality reconstruction on edges, the three methods are compared at equal *Peak-Signal-to-Noise-Ratio (PSNR)*,(46 dB, JPEG with a Quality factor Q=75, JPEG2000-Q=25). A zoom on the head of a character presenting initially sharp edges, highlights the difference of edge quality depending on the compression type (Fig.7). While at high PSNR, the JPEG (a) and JPEG2000 (b) versions of the depth map tend to blur the edges. This is commonly referred to as ringing artifacts. It appears on JPEG because of the lossy quantization of discrete cosine transform coefficients within 8x8 pixels blocks. The same but less pronounced ringing effect appears on JPEG2000 at equal PSNR, due to lossy quantization following wavelet transformation. In addition, JPEG and JPEG2000 cannot efficiently reconstitute the smooth gradient on uniform areas. At the opposite, our proposed approach stores the exact edges and interpolated regions between these edges, resulting in a smooth gradient restitution on slanted surfaces and non distorted edges.

Thus we evaluate the global depth-map rate-distortion performances of the three encoding methods. Fig.8 shows that our approach outperforms JPEG2000 except in very low or high bitrate conditions. An adjustment of the number and location of seeds in these conditions might however improve the performance.

3.4.3 View synthesis quality evaluation

The impact of depth compression methods on rendering is measured by calculating the PSNR of a synthesized view (from a pair of uncompressed textures and compressed

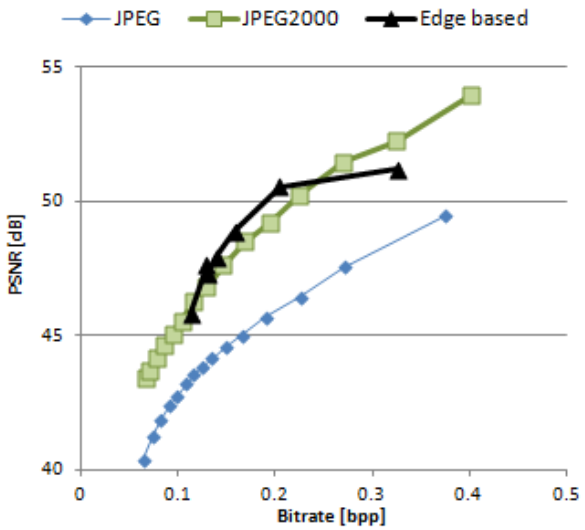


Figure 8: Rate-Distortion performance of the V_3 depth map for different quality factors of JPEG and JPEG2000, and different Sobel detection thresholds of our method.

depth maps), with respect to an original synthesized view (from a pair of uncompressed textures and depth maps). The corresponding synthesized view from two original depth maps is then the reference. VSRS 3.0 [41] is used for view interpolation from this 2-view dataset. The R-D synthesis performance, illustrated in Fig.9, justifies the edge-coding approach: undistorted edges permits an accurate and efficient view coding and rendering. Again, the PSNR measure shows its limitation of objective evaluation on perceived quality. Even at equal PSNR, our synthesized view (Fig.7h) not only outperform in term of bitrate the existing methods (Fig.7e, f), but also improved the view rendering perceived quality.

3.5 Conclusion

We proposed a new method for lossless-edge depth map coding based on optimized path and fast homogeneous diffusion. Our method, combined with a Sobel edge detection, provides a simple but efficient compression of edges that enables perfect restoration of the contours in depth maps. Also, it outperforms the state of the art JPEG and JPEG2000 compression methods. Thanks to careful edge selection and seeding, we also manage to increase the quality reconstruction of previous works based on edge image coding. Also, this lossless edge coding method could be locally applied to color image compression, especially on uniform areas. In this case the edge detection method should probably be optimized depending on edge smoothness. Finally, a depth map video encoder is in our scope for future research.

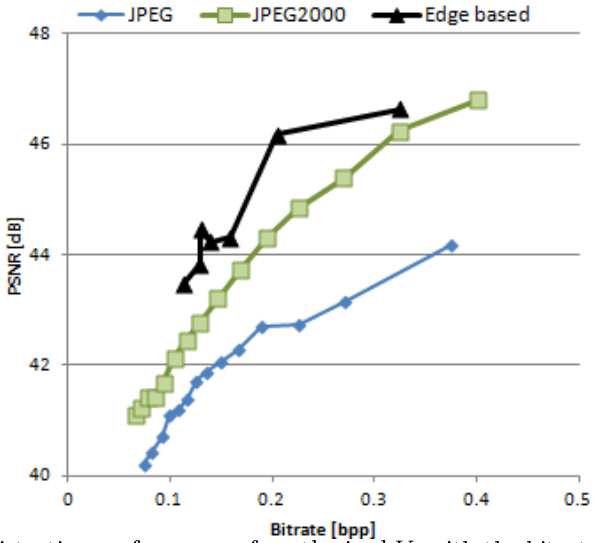


Figure 9: Rate-Distortion performance of synthesized V_4 with the bitrate of V_3, for different quality factors of JPEG and JPEG2000, and different Sobel detection thresholds of our method.

4 Layered Depth Image Representations for improved virtual view synthesis in a rate-constrained context.

4.1 Introduction

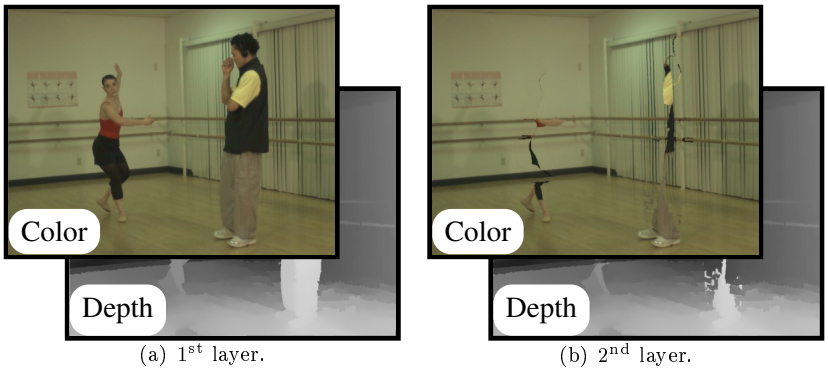
A multi-view video is a collection of video sequences captured for the same scene, synchronously by many cameras at different locations. Associated with a view synthesis method, a multi-view video allows the generation of virtual views of the scene from any viewpoint [7, 45]. This property can be used in a large diversity of applications [38], including Three-Dimensional TV (3DTV), Free Viewpoint Video (FTV), security monitoring, tracking and 3D reconstruction. The huge amount of data contained in a multi-view sequence needs an efficient compression [26].

However, the compression algorithm is strongly linked to the data representation as well as to the view synthesis methods. View synthesis approaches can be classified in two classes. Geometry-Based Rendering (GBR) approaches use a detailed 3D model of the scene. These methods are useful with synthetic video data but they become inadequate with real multi-view videos, where 3D models are difficult to estimate. Image-Based Rendering (IBR) approaches are an attractive alternative to GBR. Using the acquisition videos accompanied by some low-detailed geometric information, they allow the generation of photo-realistic virtual views.

The Layer Depth Image (LDI) representation [34,43] is one of these IBR approaches. In this representation, pixels are no more composed by a single color and a single depth value, but can contain several colors and associated depth values. It extends the 2D+Z representation, but instead of representing the scene with an array of depth pixels (pixel color with associated depth values), each position in the array may store several depth pixels, organised in layers. This representation is shown in Figure 10. I-LDI Layers turn out to be more compact, with a less spread pixel distribution, and thus easier to compress than LDI, while enabling visual rendering of similar quality as with classical LDI. It efficiently reduces the multi-view video bitrate while enabling photo-realistic rendering, even with complex scene geometry.

This representation reduces efficiently the multi-view video size, and offers a fast photo-realistic rendering, even with complex scene geometry. Various approaches to LDI compression have been proposed [14, 43, 44], based on classical LDI's layers constructions [8, 43]. The problem is that layers generated are still correlated, and some pixels are redundant between layers.

This paper addresses the problem of LDI construction aiming at improving the trade-off between compactness or compression efficiency and original or virtual view rendering quality. The first construction method is incremental, hence the name incremental LDI (I-LDI). The iterative construction procedure allows reducing the inter-layer correlation. The number of layers is significantly reduced for an equivalent final rendering quality. Techniques are then described to overcome visual artifacts, like sampling holes and ghosting artifacts [8, 31, 45]. The I-LDi construction method allows reducing the correlation between layers. However, artifacts remain which result from depth discontinuities, in particular after depth map compression.



(a) 1st layer. (b) 2nd layer.

Figure 10: Two first layers (color + depth map) of a classical LDI.
Synthesized from "Ballet" [45], views 4–3–5 at $t=0$, with incremental method [17].

In order to further improve the trade-off between the compactness of the representation and the quality of the view synthesis, a novel object-based LDI representation is then proposed. This representation organises LDI pixels into two separate layers (foreground and background) to enhance depth continuity. The number of layers inside a LDI is not the same for each pixel position. Some positions may contain only one layer, whereas some other positions may contain many layers (or depth pixels). If several depth pixels are located at the same position, the closest belongs to the foreground, visible from the reference viewpoint, whereas the farthest is assumed to belong to the background. If there is only one pixel at a position, it is a visible background pixel, or a foreground pixel in front of an unknown background. The construction method makes use of a background-foreground segmentation method based on a region growing algorithm. Once the foreground/background classification is done, the background layer is most of the time not complete. Some areas of the background may not be visible from any input view. To reconstruct the corresponding missing background texture, the construction method then uses inpainting algorithms on both texture and depth map images. The advantage is that the costly inpainting algorithm is processed once, during the LDI classification, and not during the synthesis of each view. The resulting object-based LDI representation leads to good compression efficiency due to the fact that depth pixels from a real 3D object belong to the same layer, increasing the compression efficiency thanks to higher spatial correlation, hence effective spatial prediction of texture and depth map. Moreover, these continuous layers can be rendered efficiently (in terms of both speed and reduced artifacts) by using mesh-based rendering techniques.

4.2 LDI representations: Background

The concept of LDI has first been introduced in [34], for complex geometries. An LDI contains potentially multiple depth pixels at each discrete location in the image.

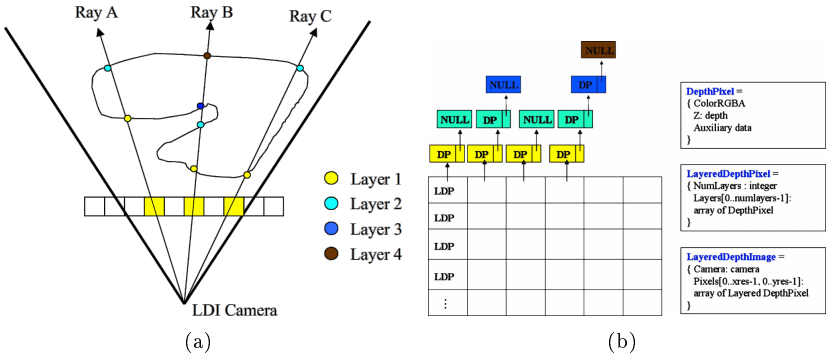


Figure 11: Structure of a Layer Depth Image (LDI).

Instead of a 2D array of depth pixels (a pixel with associated depth information), an LDI is a 2D array of layered depth pixels. A layered depth pixel stores a set of depth pixels along one line of sight sorted from front to back order. The front element in the layered depth pixel samples the first surface seen along that line of sight; the next pixel in the layered depth pixel samples the next surface seen along that line of sight, etc. Figure 11(a) presents the LDI representation as defined in [34].

Various methods have been proposed in the literature to construct LDI representations. They principally differ on the type of input data they require to operate which can be either a 3D model, a set of images, or a set of multiple video plus depth sequences. Here we consider multi video plus depth input data sets. An LDI can be generated from real multi-view + depth video sequences by using a warping algorithm. The algorithm uses a view and the associated depth map to generate a new viewpoint of the scene.

More precisely, given a set of viewpoints and one depth map per view, the classical algorithm for constructing a LDI [8, 43] proceeds in three steps, as summarized in Figure 12. First, an arbitrary viewpoint is chosen as the reference viewpoint. This reference viewpoint is usually chosen among input viewpoints, but this is not an obligation. Then, each input view is warped onto this reference viewpoint using a DIBR method. Finally, all these warped views are merged into a single LDI model, where each pixel position may contain many layered depth pixels. There are many merging policies depending on the application. Keeping all depth pixels results in unnecessarily highly redundant layers. It is preferable to keep at each pixel location, only pixels whose depth value significantly differs from that of the others. We use a threshold Δ_d on the depth value to eliminate pixels with very similar depth values.

The size of the representation grows only linearly with the observed depth complexity in the scene. The number of layers is not limited by the definition, and depends on the complexity of the scene. In practice, an LDI is often limited to a small number of layers. The first three layers of such a LDI are shown in Figure 13. Layered pixels are ordered according to their depth value. The first layer is composed of pixels with smallest depth, the second layer contains pixels with second smallest depth, and so

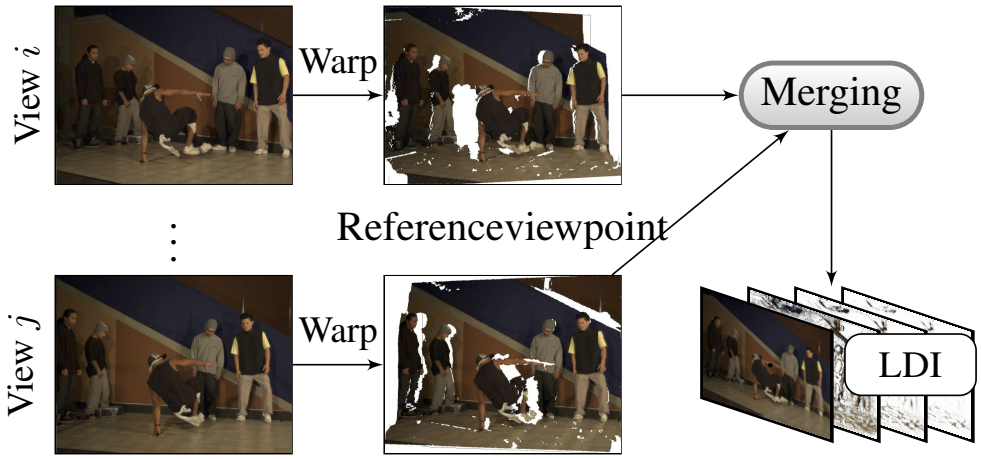


Figure 12: Classical LDI construction scheme.

(a) 1st layer (b) 2nd layer (c) 3rd layer

Figure 13: First layers of an LDI frame. 8 inputs views are used for the generation. ($\Delta_d = 0.1$)

on. We observe that, except for the first one, layers are partially empty, but non-empty pixels are sparsely distributed all over the layer. Furthermore, many pixels are redundant between the layers. These characteristics make it difficult to efficiently compress the LDI.

In [31], the authors propose to project the reference view onto each additional viewpoint, in order to recover disoccluded textures, so called residual data. These disoccluded depth-pixels can either be preserved onto their original viewpoint, or projected onto the reference viewpoint to be included into the LDI. The temporal extension of LDI is called LDV, for Layered Depth Video.

An alternative construction algorithm, that we call I-LDI (for Incremental LDI), is described in Section 4.3.1. This incremental algorithm reduces the number of pixels inside an LDI while enhancing the final rendering quality. A layer reorganization algorithm is proposed in Section 4.5, improving synthesized virtual views quality, in a

rate-constrained context. Pixels from each layer are reorganized to enhance depth continuity. A compression method is also proposed which exploits temporal correlations ant inter-layer correlations.

4.3 Incremental Layer Depth Image (I-LDI) Representation

This section introduces the incremental construction scheme, based on extra information extraction [31]. The algorithm is incremental, which means that input views are treated sequentially, in a fixed order. Figure 14 illustrates the incremental construction scheme.

4.3.1 I-LDI construction algorithm

The algorithm starts with an empty LDI, for which the reference viewpoint is chosen freely. The reference view is used to create an I-LDI with only one layer (the view itself). The reference viewpoint is often one of the input viewpoints, but is sometimes an intermediate viewpoint between two input views. Then, this I-LDI is warped iteratively on every other viewpoint (in a fixed order). The synthesized view is then compared with the original acquired view, and the discovered information is isolated, using a logical exclusion difference between the real view and the warped I-LDI to compute the residual information. This discovered information is warped back into the reference viewpoint and inserted in the I-LDI layers. By this method, only required residual information from side views is inserted, and no pixels from already defined areas are added to the L-LDI. On the other hand, all the information present in the MVD data is not inserted in the I-LDI.

The first three layers of such an I-LDI are presented in Figure 15. Compared to LDI layers, I-LDI layers contain fewer pixels, and these pixels are grouped in connected clusters. Indeed, with this method, only required extra (or residual) information from side views is inserted, and no pixels from already defined areas are added to the I-LDI. On the other hand, all the information present in the MVD data is not inserted in the I-LDI, reducing the correlation between layers.

The ordering of input views and the rendering method used to synthesize virtual views from the I-LDI are factors which have an important impact on the compactness of the representation and on the quality of the synthesized views. The first input view to be used in the I-LDI construction is the one which will contribute the most to the final I-LDI. This view should thus be the one for which the camera viewpoint is the nearest to the reference viewpoint of the I-LDI. The reference viewpoint is very often chosen as one of the input cameras, in this case, the first layer of the I-LDI is almost identical to the reference view. The only difference is located around depth discontinuities, where the JPF method uses a confidence scoring to remove ghosting artifacts. The other views can then be ordered arbitrarily, but local rendering artifacts may appear, depending on views insertion order. Best results are obtained when input views are sorted by increasing distance compared to the reference camera. In that case, each input view contributes to the LDI construction by bringing a little band of pixels

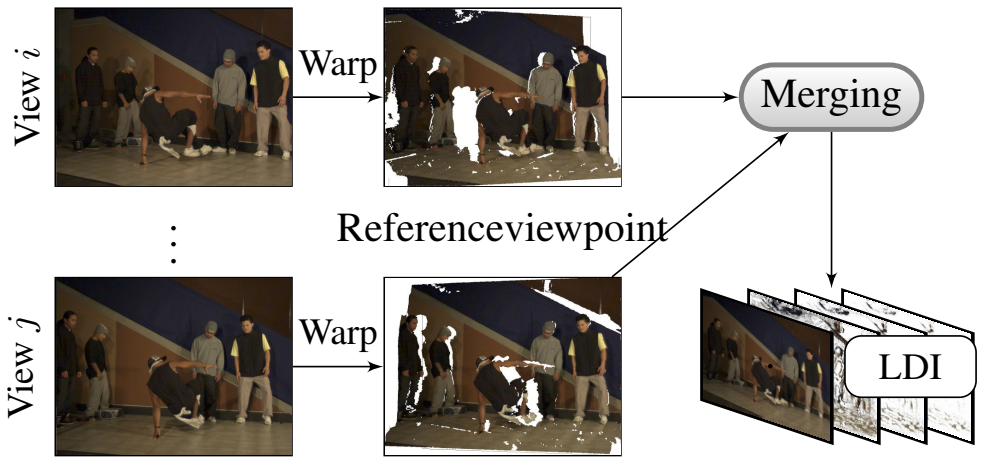
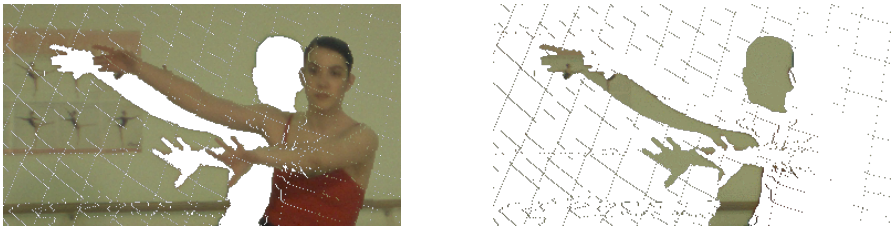


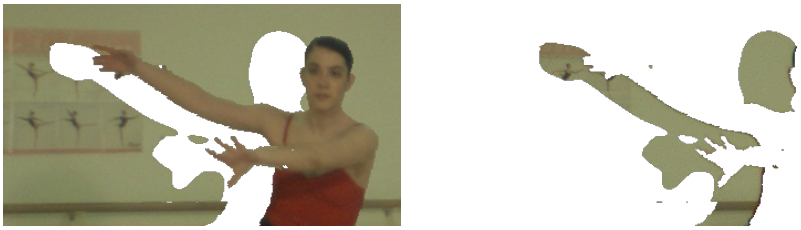
Figure 14: Step of I-LDI construction for view i, with residual information extraction.

(a) 1st layer (b) 2nd layer (c) 3rd layer

Figure 15: First layers of an I-LDI frame. All 8 inputs views are used for the generation, in a B-hierarchical order.



(a) Point-based view synthesis.



(b) The JPF method.

Figure 16: Residual information obtained with 16(a) a point based view synthesis and 16(b) the JPF method. The first column shows the rendered view of an intermediate I-LDI and the second column shows the corresponding residual information which should be inserted into the I-LDI.

The rendering method used at each iteration to synthesize virtual views from I-LDI, should verify some properties. A classical point based projection generates three kinds of artifacts: disocclusions, cracks and ghosting artifacts. We use the JPF method to handle cracks without the need of an additional filtering step. The JPF method also fills in disocclusions.

4.3.2 Warping

In this section, we explicit the equations used in the warping process. Let $(X, Y, Z, 1)$ be a 3D point in homogeneous coordinates, which is projected onto pixel $p_1 = (x_1, y_1, 1)$ in view V_1 and pixel $p_2 = (x_2, y_2, 1)$ in view V_2. Pixel coordinates p_i in view V_i are derived from the projection equations:

$$\omega_i \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} C^i_{1,1} & C^i_{1,2} & C^i_{1,3} & C^i_{1,4} \\ C^i_{2,1} & C^i_{2,2} & C^i_{2,3} & C^i_{2,4} \\ C^i_{3,1} & C^i_{3,2} & C^i_{3,3} & C^i_{3,4} \end{bmatrix} \times \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{7}$$

where C^i is the 3×4 projection matrix depending on the viewpoint i, and ω_i is an arbitrary scale factor. Knowing both camera parameters and the depth map Z_{p_1} associated to view V_1, the warping equations provide p_2 coordinates as a function of p_1 and Z_{p_1}. Warping algorithm works in two steps. The first step uses p_1 and Z_{p_1} to

estimate the 3D point (X, Y, Z_{p_1}). The second step uses this estimated 3D point to evaluate the pixel position p_2 in the new viewpoint image.

To solve the first step, we need to inverse the projection equation (7). Let (L_1), (L_2) and (L_3) be the three linear equations corresponding to the matrix notation (7) which are combined as follows:

$$\begin{array}{rr} & (C^1_{2,2} \cdot 1 - C^1_{3,2} \cdot y_1) \cdot (L_1) \\ + & (C^1_{3,2} \cdot x_1 - C^1_{1,2} \cdot 1) \cdot (L_2) \\ + & (C^1_{1,2} \cdot y_1 - C^1_{2,2} \cdot x_1) \cdot (L_3) \end{array} \tag{8}$$

Unknown parameters Y and ω_1 can then be eliminated by simplifying equation (8) giving:

$$\begin{array}{rr} & X \cdot \det\left(\left[C^1_{.,1} \middle| C^1_{.,2} \middle| p_1 \right] \right) \\ + & Z_{p_1} \cdot \det\left(\left[C^1_{.,3} \middle| C^1_{.,2} \middle| p_1 \right] \right) \\ + & \det\left(\left[C^1_{.,4} \middle| C^1_{.,2} \middle| p_1 \right] \right) \end{array} = 0 \tag{9}$$

where $C^1_{.,i}$ is the i^{th} column of the C^1 matrix. A direct form for the point's abscissa X is given by equation (9), and the same kind of equation could be written to estimate Y.

Compared to a classical matrix inversion, some coefficients of the equation (9) only depend on p_1 and do not change during warping of all layered depth pixels at a same pixel location. By implementing this optimization, we reduce by 49% the number of multiplications needed to warp a full LDI, and almost as much for it's time consumption.

Each pixel is warped independently of the others. To avoid the use of a depth buffer, we implemented the McMillan's priority order list algorithm [21]. Warping results are shown in figure 17(a).

4.3.3 Holes filling by inpainting

Directly applying warping equations may cause some visual artifacts, due mostly to disocclusion and sampling [8, 31, 45]. This section describes our simple inpainting method to fill sampling holes, visible in figure 17(a).

Let V_p be the pixel color at the p position, and W_p a neighborhood around p. Undefined pixels p can be interpolated as:

$$V'_p = \frac{1}{k} \sum_{q \in W_p} V_q \tag{10}$$

where k is the number of defined pixels within W_p.

This inpainting solution is used both during the rendering stage, and during the I-LDI construction. During the rendering stage, it improves the visual quality by interpolating all missing pixels. During the I-LDI construction, it is used carefully to fill only the sampling holes, and to leave disocclusion areas unchanged. Results are shown in figure 17(b). If a pixel is undefined due to a sampling effect, it should be surrounded by many defined pixels, which mean a high k value. If the pixel is undefined due to a large disocclusion area, it should be surrounded by many undefined pixels, which mean a low k value. The classification is done by comparing k with a threshold Δ_k.

(a) Basic Warping (b) Sampling holes filled

Figure 17: View warping and sampling holes filling

4.4 Ghosting artifacts removal

In real pictures, pixels along object boundaries receive the contribution from both foreground and background colors. Using these blended pixels during the rendering stage results in ghosting artifacts (visible in figure 18(a)). We remove these blended pixels from the reference view before I-LDI construction. Their color is thus imported by side cameras during the I-LDI construction.

Blended pixels in a view can be identified by an edge detector performed on the associated depth map. Let p be a pixel position, we estimate the depth mean $\bar{d}_p$ and depth variance v_p within a neighborhood W_p around p. Pixels near a boundary have a high variance, but among these pixels, only those from the background side of a boundary may cause a visible artifact. We then remove pixels p such $v_p > \Delta_v$ and $d_p > \bar{d}_p$ where Δ_v is a threshold.

The result of our ghosting removal method is visible in figure 18. The silhouette behind the person is erased.

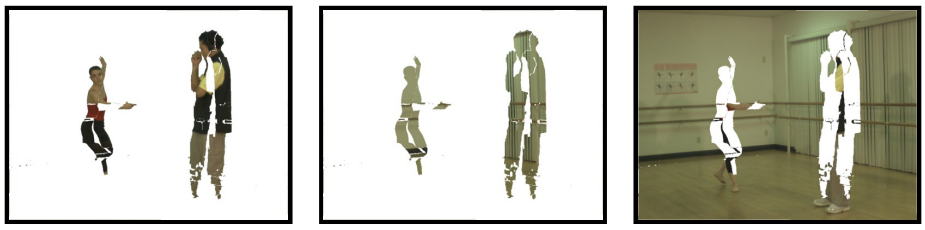
4.5 Onject-based Layer Depth Image (O-LDI) Representation

In order to overcome artifacts which result from depth discontinuities, in particular after depth map compression, a novel object-based LDI representation is proposed. This representation organises LDI pixels into two separate layers (foreground and background) to enhance depth continuity. If depth pixels from a real 3D object belong to the same layer, then compression is more efficient thanks to higher spatial correlation which improves effective spatial prediction of texture and depth map. Moreover, these continuous layers can be rendered efficiently (in terms of both speed and reduced artifacts) by using mesh-based rendering techniques.

The number of layers inside a LDI is not the same for each pixel position. Some positions may contain only one layer, whereas some other positions may contain many layers (or depth pixels). If several depth pixels are located at the same position,

(a) Without boundaries detection. (b) With boundaries detection.

Figure 18: Ghosting artifacts removal results from rendering view. (W_p is a 11×11 window)



(a) Foreground. (b) Background. (c) Unclassified.

Figure 19: Initialising state of the region growing algorithm.

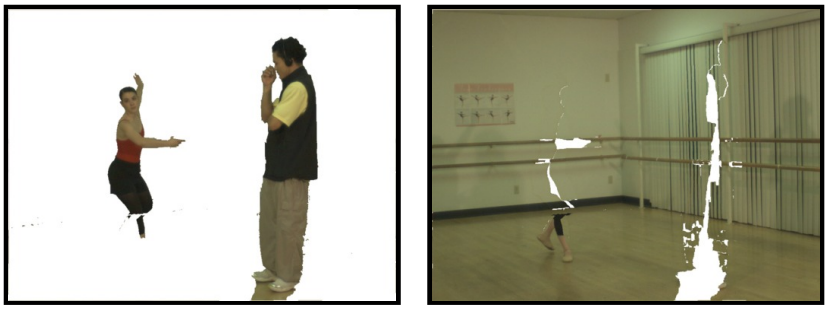
the closest belongs to the foreground, visible from the reference viewpoint, whereas the farthest is assumed to belong to the background. If there is only one pixel at a position, it is a visible background pixel, or a foreground pixel in front of an unknown background.

4.5.1 Background-Foreground segmentation

This section presents a background-foreground segmentation method based on a region growing algorithm, which allows organising LDI's pixels into two object-based layers.

First, all positions p containing several layers are selected from the input LDI. They define a region R, shown in Figure 19, where foreground and background pixels are easily identified. Z_p^{FG} denotes foreground depth, and Z_p^{BG} denotes background depth at position p. For each position q outside the region R, the pixel P_q has to be classified as a foreground or background pixel.

The classified region grows pixel by pixel, until the whole image is classified, as shown in Figure 20. For each couple of adjacent positions (p, q) around the border of region R such that p is inside R and q is outside R, the region R is expanded to q by classifying the pixel P_q according to its depth Z_q. For classification, Z_q is compared to



(a) Foreground. (b) Background.

Figure 20: Final layer organisation with the region growing classification method.

background and foreground depths at position p. An extra depth value is then given to position q, so that q is associated with both a foreground and a background depth value.

$$P_q \in \begin{cases} \text{foreground} & \text{if } (Z_p^{BG} - Z_q) > (Z_q - Z_p^{FG}) \\ & \text{so } Z_q^{FG} = Z_q \text{ and } Z_q^{BG} = Z_p^{BG} \\ \text{background} & \text{if } (Z_p^{BG} - Z_q) < (Z_q - Z_p^{FG}) \\ & \text{so } Z_q^{FG} = Z_p^{FG} \text{ and } Z_q^{BG} = Z_q \end{cases}$$

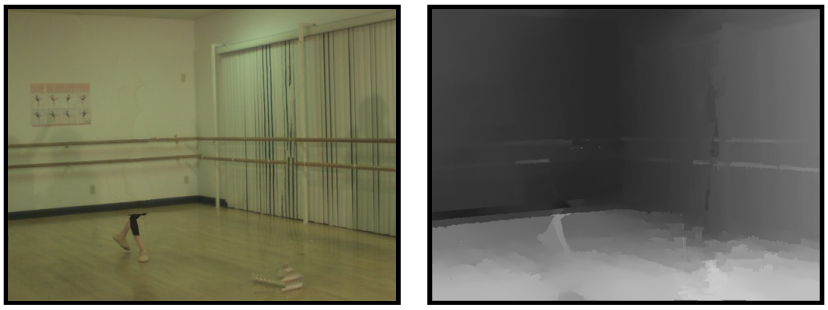
Figure 20 shows the result of classification by region growing.

4.6 Background filling by inpainting

Once the foreground/background classification is done, the background layer is most of the time not complete (see Figure 20(b)). Some areas of the background may not be visible from any input view. To reconstruct the corresponding missing background texture, one has to use inpainting algorithms on both texture and depth map images. The costly inpainting algorithm is processed once, during the LDI classification, and not during each view synthesis. Figure 21 shows the inpainted background with Criminisi's method [10].

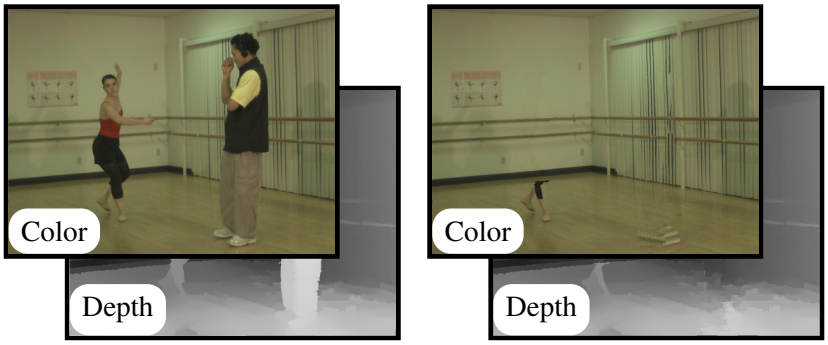
4.7 Compression

Both classical LDI, I-LDI and object-based LDI are compressed using the Multi-view Video Codec (MVC) [40], both for texture layers, and for depth layers. The MVC codec, an amendment to H.264/MPEG-4 AVC video compression standard, is DCT-based and exploits temporal, spatial and inter-layer correlations. However, MVC does not deal with undefined regions on LDI layers. To produce complete layers, each layer is filled in with pixels from the other layer, at the same position, as shown in Figure 22. This duplicated information is detected by the MVC algorithm, so that



(a) Texture. (b) Depth map.

Figure 21: Background layer obtained after texture and depth map inpainting with the Criminisi's method [10].



(a) Foreground. (b) Object-based background.

Figure 22: Final layers of an object-based LDI.

it is not encoded into the output data flow and it can be easily removed during the decoding stage.

4.8 View Synthesis

There exists a number of algorithms to perform view rendering from a LDI. This section briefly presents the two methods which have been implemented, focusing respectively on efficiency and quality.

The fastest method transforms each continuous layer into a mesh, which is rendered with a 3D engine, as shown in Figure 23. The foreground mesh is transparent on background region in order to avoid stretching around objects boundaries. Our first experiments, with this method, have shown the feasibility of real time rendering for an eight-views auto-stereoscopic display.

The second method improves the visual quality of synthesized views by using a point-

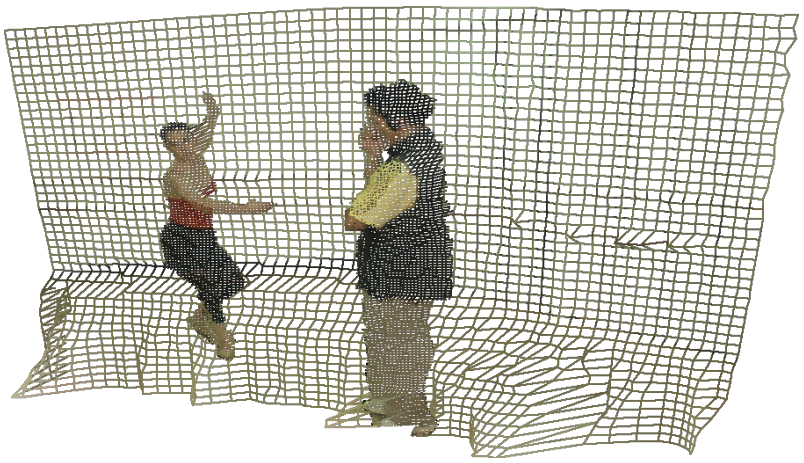


Figure 23: Fast 3D rendering of a high detailed foreground mesh, onto a low detailed background mesh.

based projection. It combines the painter's algorithm proposed by McMillan [21], and diffusion-based inpainting constrained by epipolar geometry. Remaining disocclusions areas are filled in with background texture. Figure 24 presents rendering results for both classical and object-based LDI.

4.9 Experimental results

Experiments have been conducted on Breakdancers and Ballet data sets from MSR [45]. Parameters of the 8 acquisition cameras and all associated depth maps are already estimated and provided within the data. The viewpoint number 4 is considered as the reference viewpoint. Only frames for time $t = 0$ are considered.

For the LDI construction, all 8 acquired views are warped into the reference viewpoint. A small merging threshold value $\Delta_d = 0.1$ is used in following comparisons. For the I-LDI construction, views are inserted in a B-hierarchical order (4; 0; 7; 2; 6; 1; 5; 3). Thresholds are set by experiments: $\Delta_v = 20$ for boundary detection and $\Delta_k = 60\% \cdot N$ for inpainting, where N is the number of pixels within the W_p window. Both inpainting and boundary detection are done within W_p a 11×11 window.

All 8 input views are used, but all pixels from each view are not inserted in the LDI. Because of the depth threshold in the LDI construction, and of the exclusion difference in I-LDI construction, some pixels from side views are ignored. Figure 25 presents the ratio of pixels from each view which is really inserted in the LDI. We can observe that few pixels are inserted from view 5 to 8, means these views become almost useless with the I-LDI construction scheme. Using only a subset of acquired views (the reference and the extreme views) provides almost the same I-LDI layers.

Figure 26 shows the ratio of defined pixels in each layers for both LDI and I-LDI construction schemes. For both constructions, the first layer contains 100% of it's

(a) Classical LDI. (b) Object-based LDI.

Figure 24: Rendering comparison between classical and object-based LDI.

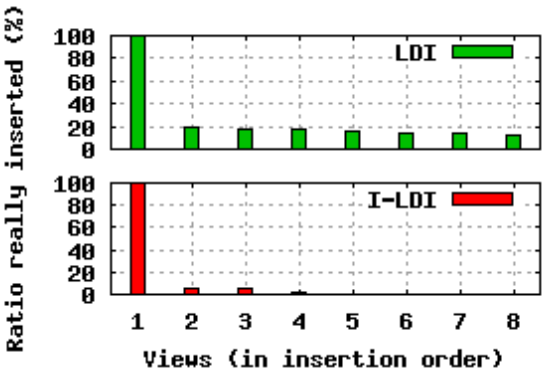


Figure 25: Utilization rate of acquired views during layers construction

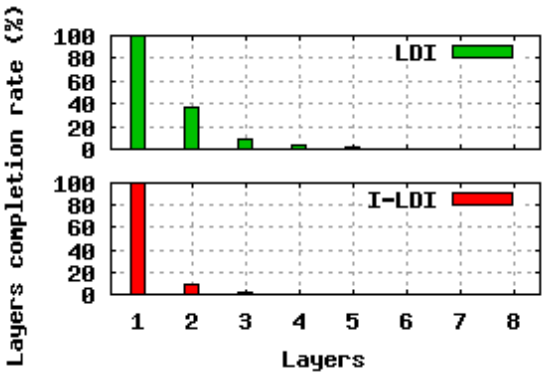


Figure 26: Layers completion rate for LDI and I-LDI

pixels, and differences appear for extra layers. For the LDI, extra layers represent more than 50% of the size (in number of pixels) of the first layer, whereas for the I-LDI, extra layers represent less than 10%. Layers beyond the 3rd one are quite empty and can be ignored. The visual rendering is of similar quality with both LDI and I-LDI construction scheme. Local rendering artifacts may appear, depending on views insertion order.

The rendered quality of object-based LDI is compared with classical LDI on one side, and state-of-the-art MPEG compression techniques on the other side. Images are taken from "Ballet" data sets, provided by MSR [45]. Only frames for time $t = 0$ are considered.

In the first place, a LDI restricted to two layers, is constructed from three input views: the reference view 4 and side views 3 and 5 alternatively. To deal with unrectified camera sets and reduce correlation between layers, we use the Incremental LDI construction algorithm described in [17]. The corresponding object-based LDI is obtained by applying our region growing classification method on the classical LDI.

Classical LDI and object-based LDI are compressed using the MVC algorithm, as explained in section 4.7. Several quantization parameters were used, from QP=18 to QP=54, producing compressed output data flows with bit-rates going from 1 Mbit/s to 25 Mbit/s. These compressed data flows are used to synthesize virtual views onto viewpoint 6, using the pixel-based projection method.

In the second place, the state-of-the-art method for multi-view video coding is used with the same input data. Views 1, 3, 5 and 7 are coded with the MVC algorithm with various quantization parameters, then the compressed views 5 and 7 are used to synthesize virtual views onto viewpoint 6, using the MPEG/VSRS software [40].

Finally, all synthesized views are compared to the original view 6, using the SSIM comparison metrics. Figure 27 presents all the results as three rate distortion curves. For each quantization parameter, object-based LDI can be better compressed than classical LDI, resulting in a smaller bitrate. The rendering quality is also better,

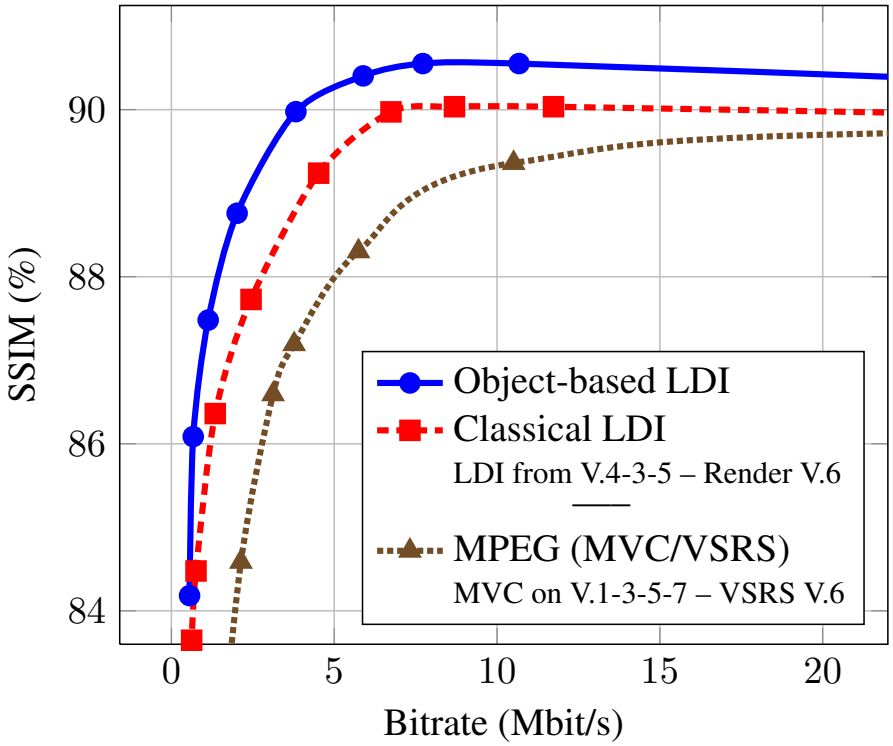


Figure 27: Rate distortion curves firstly for LDI (object-based or not) compressed by MVC and rendered by our point-based projection, and secondly for multi-view video compressed by MVC and rendered with VSRS algorithm.

resulting in a higher SSIM for the same quantization parameter. Combining these two advantages, the rate distortion curve for the object-based LDI is higher than the one for classical LDI, for every bitrate.

4.10 Conclusions and future work

This paper has presented two methods for constructing layered depth image representations from natural multi-view images. The first method is an incremental procedure which uses iterative warping. The minimum information to fill disocclusion areas is inserted into LDI layers which makes layers easier to compress. They contain 80% less pixels, and with a more compact distribution. To overcome visible artifacts, some simple solutions have been proposed. The sampling holes filling by pixel interpolation provides good results. The ghosting removal by depth discontinuity detection may cause some luminosity discontinuity between textures from different views.

In future work, we will investigate an improved disocclusion detection to insert into

the I-LDI all occluded textures. An alpha merging approach will be used to reduce luminosity discontinuity artifacts. Finally, the compression stage will be investigated with a full video sequence.

5 A content based method for perceptually driven joint color/depth compression

5.1 Introduction

3D Video applications [37], such as 3D Television (3DTV) or Free Viewpoint Television (FTV), require the use of several conventional video sequences to ensure depth sensation, or to offer novel views of a scene. For these purposes, the use of color and geometry information of the scene is the key. MVD data refer to a specific representation of an observed scene and meet this need. They consist in a set of conventional color video sequences and an associated set of depth video sequences, all acquired at slightly different viewpoints.

A first issue refers to the need for an efficient MVD compression method, considering the huge amount of data to be processed. Up to now, there is no standardized compression method for MVD sequences. Most of the proposed compression methods rely on the extension of state-of-the-art 2D codecs. The most popular is H264/AVC [39] whose 3D extension (standardized for Multi-View-Video representation, MVV), namely H.264/MVC for Multi-view Video Coding [24], has been the subject of many adaptations for MVD compression [30]. However, the exploitations of the spatial inter-view redundancies in both types of data turn out to be insufficient in particular cases. For instance, Merkle *et al.* [27] observed that in case of large disparity between the different views of multi-view sequences, the predictions structures did not result in an improved coding efficiency.

A second issue refers to the synthesis of novel views from decoded data. New intermediate viewpoints can be generated from depth and color data through Depth-Image-Based-Rendering [15] (DIBR) methods. Previous studies already pointed out the impact of depth encoding on the synthesized frames. Compression-related artifacts that may be imperceptible in depth maps cause important distortion during the synthesis process [22].

Many methods have been proposed recently in order to address the aforementioned issues. Various encoding strategies are possible to achieve depth map compression. Several studies have proposed bit-rate-control methods [12,29] relying on the objective quality of the resulting synthesized views, or on a distortion model [19]. A popular and efficient strategy is the post-processing of depth maps after decoding [13]. Depth-adapted encoding methods [16, 28, 33] have also been proposed. Section 5.2 gives a review of these methods. Our work is in line with the depth-adapted encoding strategy since the method proposed in this paper relies on a content-based representation of the depth map.

The main purpose of this novel framework is to preserve the consistency between color and depth data. Our strategy is motivated by previous studies [22] of artifacts occurring in synthesized views: most annoying distortions are located around strong depth discontinuities [6] and these distortions are due to misalignment of depth and color edges in decoded images. Thus the method is meant to preserve edges and to ensure consistent localization of color edges and depth edges. It is based on a 2D

still image codec, namely LAR [32] (Locally adapted Resolution). The LAR codec is based on a quad-tree representation of the images. In this quad-tree, the smaller the blocks, the higher the probability of the presence of a depth discontinuity. Analogously, big blocks correspond to smooth areas. The quad-tree representation contributes in the preservation of depth transitions when target bit-rate decreases. Another original contribution of the proposed method relies on the use of the decoded color data as an anchor for the enhancement of the associated decoded depth, together with information provided by the quad-tree structure. This is meant to ensure consistency in both types of data after decoding.

This paper is organized as follows: Section 5.2 introduces the compression issues for MVD data. Section 5.3 presents the proposed method. Section 5.4 defines the experimental protocol and gives the results. Finally Section 5.5 concludes the paper.

5.2 Compression of mvd sequences and quality of synthesized views

This section presents the main issues related to MVD compression and a review of the proposed methods addressing these problems in the literature.

Most of the proposed compression methods for MVD data rely on the extension of state-of-the-art 2D codecs. Sets of color and depth sequences can be separately encoded through existing 2D methods. This is an evident encoding strategy because depth maps, being monochromatic signals, are considered as conventional sequences. However, depth maps are not natural images. They provide structural information of the scene: large and smooth regions often belong to the same depth plane. The closer the depth plane is from the acquiring camera, the lighter the region. This leads to smooth areas with sharp edges. The edges correspond to depth transitions.

Previous studies [5, 25, 42] have shown that coding artifacts on depth data can dramatically influence the quality of the synthesized view. Particularly, the sharp edges of the depth maps are prone to synthesis errors even when depth maps are uncompressed. As pointed out in a recent study [6], the synthesis process, with DIBR methods, induces specific artifacts located around the edges of objects. These errors are notably due to depth map inaccuracy, numerical rounding, hole filling method in DIBR, or both. Consequently, errors occurring in these specific critical areas of the depth maps are enhanced by coarse compression. The impacts of depth compression on visual quality of synthesized views can be explained by the fact that 2D codecs are optimized for human visual perception of color images. Thus, artifacts, that may be imperceptible when visualizing the depth map, produce distortions because during the synthesis, the warping process relies on wrong depth values. The impacts of depth compression was observed in different studies [5, 22].

Consequently, efforts have been directed in order to propose depth compression methods more adapted to the special features of depth maps. Morvan *et al.* [28] proposed to represent the depth map thanks to platelets (piecewise linear functions). The depth map is first divided through quad-tree decomposition and each block is approximated by a platelet. The platelet-based compression outperformed JPEG2000 in the study. An additional interesting comparison would be that against H.264/AVC.

Moreover, in this study, the gain is evaluated with respect to the depth distortion (in PSNR). This protocol of validation is questionable because since the artifacts in the two compared methods are different, their impact on the synthesis may also be different. Yet, the quality of the synthesized views generated from the decoded depth maps is not presented. Graziosi *et al.* [16] also proposed a block partioning method associated to a least-square prediction for depth map compression. In this method, the validation is also achieved by comparing the depth map distortion from different compression scheme (JPEG2000 and H.264 intra). The method includes the use of a dictionary, containing concatenations of scaled versions of previously encoded image blocks. Sarkis *et al.* [33] proposed a depth compression method based on a subsampling in the frequency domain followed by a reconstruction using a non-linear conjugate gradient minimization scheme. The method also meant to preserve the particular features of the depth map. The method outperformed JPEG and JPEG2000.

As Morvan *et al.* and Graziosi *et al.*, our method relies on a block partioning. Contrary to the aforementioned methods, we choose to evaluate the performances of our method against H.264, whose artifacts in the depth map induce less annoying distortions than JPEG2000 (Gibbs effect in the depth map induce disastrous distortions in the synthesized view). The quantization used in our method is also different since we choose to modify the block partioning of the depth map according to the target bit-rate. The next section will present our proposed method that is meant to preserve object edges to maximize the synthesized view quality.

5.3 Proposed method

Since color and depth sequences can be encoded through two different schemes, the proposed method enables the use of any compression method for the color sequences. In our case, we use H.264 for color sequence encoding since it has proved its efficiency for conventional color media compression. Moreover, encoding color with a standard codec enables backward compatibility with classical 2D video. This section presents the depth map encoding strategy.

5.3.1 Depth map encoding method

To address the first constraint regarding the preservation of the depth map edges, a content-based representation and encoding is required. We choose to base our method on the LAR method because the quad-tree representation of this method matches the characteristics of the depth maps. The LAR method is based on the assumption that an image can be considered as the combination of two components: the global information and the details, which are respectively the flat image and the local texture. The flat image and the local texture both rely on the same quad-tree representation. Each pixel in the flat image is assigned the mean value of the pixels of the block it belongs to. Each pixel in the local texture is then assigned the compensated error. Figure 28 depicts this principle.

Since depth maps do not contain high frequency areas, the local texture (that is to say the details) is not essential and represents an avoidable additional cost of compres-

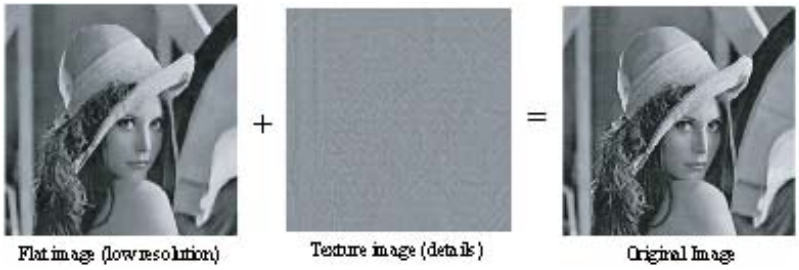


Figure 28: Assumption of LAR method.

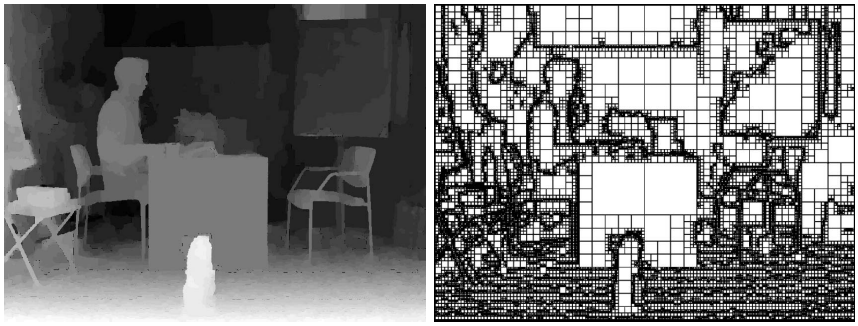


Figure 29: Quad-tree decomposition (Book Arrival).

sion. Thus, only the flat image is considered and encoded in the method we propose.

Quad-tree decomposition The quad-tree decomposition is dependent on the local gradient of the depth image. Given a threshold Y for the local gradient, the image is split into blocks: the higher the local activity, the more splits. This leads to small blocks around object edges and bigger ones in continuous areas.

We denote $P^{[N_{max}...N_{min}]}$ the quad-tree partition with N_{max} and N_{min} the maximal and minimal allowed block sizes, expressed as power of 2. Let I be an image and $I(x,y)$ a pixel of I with coordinates(x,y). The block $b^N(i,j)$ in I is noted $I(b^N(i,j))$, expressed as:

$$b^N(i,j) = \begin{cases} (x,y) \in N_x \times N_y \\ |\, N \times i \leq x < N \times (i+1)\,, \\ N \times j \leq y < N \times (j+1)\,, \\ N \in [N_{min} \dots N_{max}] \end{cases} \tag{11}$$

As explained before, the quad-tree partition $P^{[N_{max}...N_{min}]}$relies on the analysis of the local gradient. Then, the decomposition can be expressed as:

$$\forall I(b^N(i,j)), N = \begin{cases} N' \in [N_{max} \dots N_{min}[\text{ if } \max(I(b^{N'}(i,j)) - \min(I(b^{N'}(i,j)) \leq Y, \\ \text{and if } \exists (k,m) \in 0,1^2 \\ |\max(I\left(b^{\frac{N'}{2}}\ (i+k, j+m)\right) - \min(I(b^{\frac{N'}{2}}\ (i+k, j+m)) > Y \\ N_{min} \text{ otherwise} \end{cases} \tag{12}$$

The value of the threshold Y strongly influences the final representation of the image. Figure 29 gives an example of quad-tree decomposition for the first frame of *Book Arrival* sequence.

Compression scheme The compression scheme in the LAR method is based on a pyramidal decomposition [3]. The pyramid, built from I, consists of a set of images, noted as $\{L_l\}_{l=0}^{l=l_{max}}$, as a multi-resolution representation of the image, where l_{max} is the top of the pyramid and $l = 0$ is the lowest level, i.e. the full resolution image. At each level, the image is expressed by:

$$\begin{cases} l = 0,\ L_0\,(x,y) = I(x,y) \\ l > 0,\ L_l(x,y) = \left\lfloor \frac{L_{l-1}(2x,2y) + L_{l-1}(2x+1,2y+1)}{2} \right\rfloor \end{cases} \tag{13}$$

The LAR method allows the prediction of each level of the pyramid, from top to bottom. For each level, the associated image of errors, also relying on the quad-tree decomposition, can be transmitted to compensate the prediction errors. At the decoder side, from the top to the bottom, the image is reconstructed.

Compression cost is mainly due to the encoding of small blocks. This is why in our proposed method small blocks are not transmitted (those are blocks whose size is such as $N = N_{min}$). This is achievable thanks to the pyramidal decomposition. The encoding of small blocks is related to the image of errors corresponding to the lowest level, i.e. L_0. The lowest level is not encoded in the method we propose, and the image will be refined at the decoder side thanks to the analysis of the values of the nearest neighbor blocks whose size is such as $N > N_{min}$: they will be predicted, depending on the values of their closest larger blocks. This allows bit-rate savings. The pseudo code of this prediction is given in Algorithm 1.

5.3.2 Rate control in depth map

Pasteau *et al.* [32] suggested applying a quantization step depending on the block sizes, in the case of conventional images. Our experiments revealed that in the case of depth map compression, this was not an adequate strategy because the smaller the blocks, the coarser was the quantization (this allowed bit rate savings because small block are costly). Yet, small blocks correspond to strong depth discontinuities and errors occurring in these areas may have disastrous effect at the synthesis step. Figure 30 shows the impact of the quantization as suggested in Pasteau *et al.* [32] (first column) at 0.06 bpp. Depth transitions are highly degraded and will result in errors in the synthesized frame (third column, crumbling artifacts around the head and around

Algorithm 1 Prediction of lowest level of the pyramidal decomposition

Require: $\tilde{L}_l$ is the estimated representation of the image at the decoder side, for level l, $P^{[N_{max} \dots N_{min}]}$ is the quad-tree partition.
G.init($\mathbf{x}_{init}$)
for $l = l_{max} \dots l_1$ **do**
Estimate $\tilde{L}_l$ as in the LAR method
end for
for each block of P such as $N = N_{max} \dots N_{min}$ **do**
Given $P^{[}N_{max} \dots N_{min}]$, then $\tilde{L}_0(b^N(i,j)) = \tilde{L}_1$
end for
for each block of P such as $N = N_{min}$ **do**
$\tilde{L}_0(b^{(}N_{min})(i,j))$ = Mean value of the closest block b^N of P such as $N > N_{min}$
end for
return $\tilde{L}_0$

the legs of the chair). The synthesized frames obtained in Figure 30 are generated from original color data and decoded depth maps in order to visually assess only the impact of depth quantization (i.e. not the combined effect of both color and depth compression).

Thus, we propose a quantization achieved through the evolution of the quad-tree representation of the image. Small blocks are costly and a way to reduce the bit-rate is to reduce the number of small blocks. This implies that the quad-tree representation can change according to the target bit-rate. The number of small blocks is directly related to the value of the threshold Y. Thus, an increasing threshold Y decreases the bit-rate, so that the representation of the image contains larger blocks. This corresponds to a spatial quantization that affects the values of the depth. It results in assigning the same depth to object that were not formerly in the same depth plane. The dynamic range of the depth is reduced but the global structure is preserved. Figure 30 shows that the proposed method (second column) renders sharp depth transitions. The synthesized frame in Figure 30, fourth column, shows improvements compared to the previous strategy, third column. Figure 31 gives the quad-tree representations and the resulting depth maps using two different thresholds for the quad-tree decomposition. It shows that the semantic information of the image is preserved.

In this study, we empirically determined a model allowing the choice of Y depending on the target bit-rate. Based on the analysis of the synthesized view quality scores obtained for various values of Y, and according to the corresponding bit-rate R of the encoded frames, we opted to an exponential model such as:

$$Y = ae^{bR} \tag{14}$$

R is the target bit-rate, a and b are two constants. Previous experiments showed that $a = 30$ and $b = -12$ gave good results for the tested sequence. Note that these values will differ from one MVD sequence to another, since the representation of the quad-tree depends on the structure of the sequence, thus Y model depends on the

]

]Decoded depth when applying quantization suggested in Pasteau *et al.*

Decoded depth when applying our proposed strategy.

Frame synthesized from decoded depth when applying quantization suggested in Pasteau *et al.*

Frame synthesized from decoded depth when applying our proposed strategy.

Figure 30: Comparison of two decoded depth maps at 0.06bpp, using the LAR method or the proposed method of rate control.

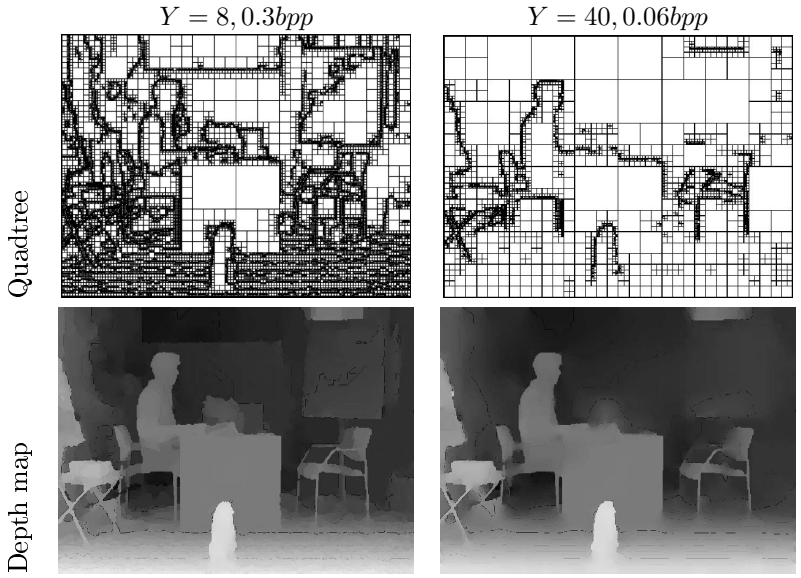


Figure 31: Quantization of the depth map.

depth structure of the sequence.

5.3.3 Depth reconstruction at decoder side

To address the second constraint regarding the consistency between color and depth edges, reconstruction step is included at the depth map decoder side, right after the first estimation of the smallest blocks, as explained in Section 5.3.1. The additional step described in this section can be considered as a second pass of the depth reconstruction. It consists of a multi-lateral filtering aided by the quad-tree representation whose principles are partially based on the description of *Lai et al.* method [18]. In our proposed method, the decoded associated color image is used to enhance only the blocks smaller or equal to 4×4 in the depth map. Small blocks are likely to be located around depth discontinuities. Thus, it is believed that improving the accuracy in these regions, according to the decoded associated color, will ensure consistency between color and depth edges. Let $\tilde{C}$ be the decoded associated color image, and $\tilde{L}_0$ the lowest level image of the depth pyramidal decomposition. Let Ω be the set of pixels such as:

$$\Omega = \left. \tilde{L}_0(x,y) \right| \tilde{L}_0\left(x,y\right) \in \tilde{L}_0\left(b^N\left(i,j\right)\right), \quad N \in [N_{min} \dots 4] \tag{15}$$

The reconstruction, noted $\tilde{L}_{0r}\left(x,y\right)$, of any pixel belonging to Ω is expressed as:

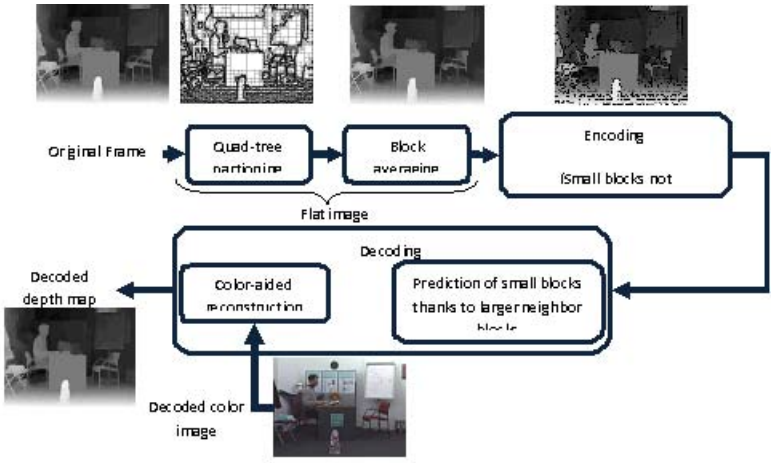


Figure 32: Overview of the proposed method.

$\forall\ \tilde{L}_0(x,y) \in \Omega,$

$$\tilde{L}_{0r}(x,y) = \tilde{L}_{0r}(p) = \frac{1}{K}\sum_{q\in\Gamma} \tilde{L}_0(p) e^{-\frac{\|p-q\|}{2\sigma_d}} e^{-\frac{\|\tilde{L}_0(p)-\tilde{L}_0(q)\|}{2\sigma_s}} e^{-\frac{\|Luma(p)-Luma(q)\|}{2\sigma_c}} \tag{16}$$

$$K = \sum_{q\in\Gamma} e^{-\frac{\|p-q\|}{2\sigma_d}} e^{-\frac{\|\tilde{L}_0(p)-\tilde{L}_0(q)\|}{2\sigma_s}} e^{-\frac{\|Luma(p)-Luma(q)\|}{2\sigma_c}} \tag{17}$$

$\Gamma$is the pixel window used for the calculation; $Luma$ is the luminance component of the decoded color image; $Luma(p)$ and $Luma(q)$ are pixels of the luminance component of the decoded color image; σ_d, σ_s, $\sigma_c$are standard deviations related to the spatial domain, the depth range domain (similarity of depth values), and the color range domain, respectively.

Figure 32 gives the overview of the proposed method. In this figure, at the encoding step, black blocks correspond to non transmitted blocks.

5.4 Experiments

5.4.1 Protocol

The proposed method is compared to state-of-the-art codec H.264 in intra mode. As preliminary studies, the experiments concern only still images. First frames of views 6 and 10 from *Book Arrival* were encoded through both encoding methods. Afterwards, decoded color and depth maps were used to compute the intermediate view 8, through

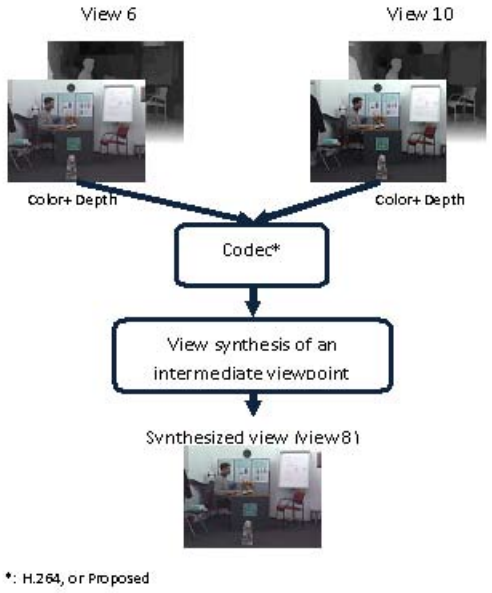


Figure 33: Overview of the experimental protocol

the reference software, VSRS 3.5 [40]. Since view 8 is among the originally acquired views, it is considered as a ground truth for the quality assessments. In this paper, the quad-tree decomposition parameters are $N_{min} = 1$ and $N_{max} = 12$. In Equation 14, $a = 30$ and $b = -12$. Finally, in Equation 7, $\sigma_d = 4$, $\sigma_s = 10$, $\sigma_c = 3$. The color images are encoded with a QP varying from 0 to 50. Figure 33 gives an overview of the experimental protocol.

5.4.2 Results

For the performance comparisons, a pixel-based metric (PSNR) and a more perception-oriented metric VIF (Visual Information Fidelity [35]) are considered. Figure 34 depicts the rate-distortion curve obtained by computing the PSNR scores, and the VIF scores of the synthesized views, with respect to the original acquired view. At high bit-rates (higher than 2bpp), the proposed method obtains better PSNR scores. However, under 2bpp, H.264 performs better.

The curve based on VIF scores shows that H.264 and the proposed method give similar results at high bit-rates (higher than 2bpp). However, contrary to the curve based on PSNR, the curve based on the perception-oriented VIF shows that the proposed method performs better at low bit-rates.

A visual appreciation is also useful to evaluate the methods. Figure 35 gives snapshots of the obtained synthesized views for 0.1bpp and 0.9bpp. Ghosting effect is

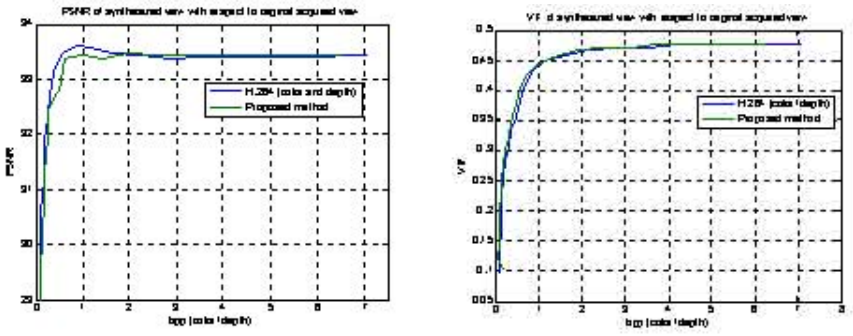


Figure 34: Performance comparisons, in terms of PSNR and VIF, between the original view and the synthesized view.

perceptible with both methods behind the head of the man. However, the quad-tree based method preserves better the vertical edges: the vertical dark lines of the posters are better rendered with the data encoded with the proposed method. At low bit-rate (0.1bpp), Figure 35 gives snapshots of the synthesized views. Although, PSNR score shows lower performances for the proposed method at low bit-rate, the observation of Figure 35 shows improvements around the edges of the synthesized objects. The ghosting effect around the head of the man is less strong with the proposed method. The crumbling artifacts occurring around the leg of the chair at 0.1bpp with H.264 are no longer perceptible with the proposed method.

5.5 Conclusion

We proposed a novel framework whose main purpose is to preserve consistency between color and depth edges. Depth encoding is based on a 2D still image codec, namely LAR (Locally Adapted Resolution). It consists in a quad-tree representation of the images. The quad-tree representation contributes in the preservation of edges in depth data. The originality of the proposed method relies on the proposed quantization method and the use of the decoded color data as an anchor for the associated depth enhancement at the decoder side. The proposed method showed visual performances similar to H.264 at high bit-rates and some improvements at lower bit-rates because it preserves better the object edges. Future work should focus on the use of a more perception-oriented criterion for the quad-tree decomposition. A method to choose automatically the Y model, for any sequence, should also be investigated. Finally, the method should be extended to exploit temporal redundancies in the whole sequence.

Figure 35: Snapshots of synthesized views from data encoded with H.264 and from data encoded with the proposed method.

References

[1] "Middlebury stereo vision dataset," http://vision.middlebury.edu/stereo/.

[2] "Joint bi-level image experts group: Information technology - progressive lossy-lossless coding of bi-level images," in *ISO/IEC JTC1 11544, ITU-T Rec. T.82, Final Committee Draft 11544*, 1993.

[3] M. Babel, O. Deforges, and J. Ronsin, "Interleaved s+ p pyramidal decomposition with refined prediction model," in *IEEE International Conference on Image Processing, 2005. ICIP 2005*, vol. 2, 2005.

[4] Z. Belhachmi, D. Bucur, B. Burgeth, J. Weickert *et al.*, "How to choose interpolation data in images," *SIAM Journal on Applied Mathematics*, vol. 70, no. 1, pp. 333–352, 2009.

[5] E. Bosc, V. Jantet, M. Pressigout, L. Morin, and C. Guillemot, "Bit-rate allocation for multi-view video plus depth," in *Proc. of 3DTV Conference 2011*, Turkey, 2011.

[6] E. Bosc, R. Pepion, P. Le Callet, M. Koppel, P. Ndjiki-Nya, M. Pressigout, and L. Morin, "Towards a new quality metric for 3D synthesized view assessment," *IEEE Journal of Selected Topics*, 2011.

[7] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "Unstructured lumigraph rendering," in *Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH), Proceedings of.* New York, NY, USA: ACM, 2001, pp. 425–432.

[8] X. Cheng, L. Sun, and S. Yang, "Generation of layered depth images from multi-view video," *IEEE International Conference on Image Processing (ICIP)*, vol. 5, pp. 225–228, Oct. 2007. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4379806

[9] G. Cheung, A. Kubota, and A. Ortega, "Sparse representation of depth maps for efficient transform coding," in *Proc. Picture Coding Symposium*, Nagoya, Japan, December 2010.

[10] A. Criminisi, P. Pérez, and K. Toyama, "Object removal by exemplar-based inpainting," in *Computer Vision and Pattern Recognition (CVPR)*, vol. 2. IEEE Computer Society, Jun. 2003, pp. 721–728.

[11] A. D'Angelo, L. Zhaoping, and M. Barni, "A full-reference quality metric for geometrically distorted images," *IEEE Trans. Image Processing*, vol. 19, no. 4, pp. 867–881, 2010.

[12] I. Daribo, C. Tillier, and B. Pesquet-Popescu, "Motion vector sharing and bit-rate allocation for 3D Video-plus-Depth coding," *EURASIP JASP Special Issue on 3DTV*, p. 258920, 2009.

[13] D. De Silva, W. Fernando, H. Kodikaraarachchi, S. Worrall, and A. Kondoz, "A depth map Post-Processing framework for 3D-TV systems based on compression artifact analysis," *IEEE Journal of Selected Topics in Signal Processing*, vol. PP, no. 99, pp. 1–1.

[14] J. Duan and J. Li, "Compression of the layered depth image," *Image Processing, IEEE Transactions on*, vol. 12, no. 3, pp. 365–372, Mar. 2003.

[15] C. Fehn, "Depth-image-based rendering (DIBR), compression and transmission for a new approach on 3D-TV," in *Proceedings of SPIE Stereoscopic Displays and Virtual Reality Systems XI*, vol. 5291, 2004, pp. 93–104.

[16] D. B. Graziosi, N. M. M. Rodrigues, C. L. Pagliari, S. M. M. de Faria, E. A. B. da Silva, and M. B. De Carvalho, "Compressing depth maps using multiscale recurrent pattern image coding," *Electronics letters*, vol. 46, no. 5, pp. 340–341, 2010.

[17] V. Jantet, L. Morin, and C. Guillemot, "Incremental-ldi for multi-view coding," in *The True Vision, 3DTV Conf.*, Potsdam, Germany, May 2009, pp. 1–4.

[18] P. Lai, D. Tian, and P. Lopez, "Depth map processing with iterative joint multilteral filtering," *PCS 2010*, 2010.

[19] Y. Liu, Q. Huang, S. Ma, D. Zhao, and W. Gao, "Joint video/depth rate allocation for 3D video coding based on view synthesis distortion model," *Signal Processing: Image Communication*, vol. 24, no. 8, pp. 666–681, Sep. 2009. [Online]. Available: http://www.sciencedirect.com/science/article/B6V08-4WJHB3J-1/2/df2bb7c1aea8e8c10d3ef1d69f64dd04

[20] M. Mainberger and J. Weickert, "Edge-based image compression with homogeneous diffusion," in *Computer Analysis of Images and Patterns*. Springer, 2009, pp. 476–483.

[21] L. McMillan, "A list-priority rendering algorithm for redisplaying projected surfaces," University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, Tech. Rep. 95-005, 1995. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.1759

[22] P. Merkle, Y. Morvan, A. Smolic, D. Farin, K. Muller, and T. Wiegand, "The effects of multiview depth video compression on multiview rendering," *Singal Processing: Image Communication*, vol. 24, no. 1-2, pp. 73–88, Jan. 2009.

[23] P. Merkle, Y. Morvan, A. Smolic, D. Farin, K. Müller, T. Wiegand *et al.*, "The effects of multiview depth video compression on multiview rendering," *Signal Processing: Image Communication*, vol. 24, no. 1-2, pp. 73–88, 2009.

[24] P. Merkle, K. Muller, A. Smolic, and T. Wiegand, "Efficient compression of multiview video exploiting inter-view dependencies based on h. 264/MPEG4-AVC," in *Proc. ICME*, 2006, pp. 9–12.

[25] P. Merkle, A. Smolic, K. Mueller, and T. Wiegand, "Multi-view video plus depth representation and coding," in *Proceedings of ICIP*, 2007, pp. 201–204.

[26] P. Merkle, A. Smolic, K. Müller, and T. Wiegand, "Efficient prediction structures for multiview video coding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, no. 11, pp. 1461–1473, Nov. 2007. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4358661

[27] P. Merkle, A. Smolic, K. Muller, and T. Wiegand, "Efficient prediction structures for multiview video coding," *IEEE Transactions on circuits and systems for video technology*, vol. 17, no. 11, pp. 1461–1473, 2007.

[28] Y. Morvan, P. de With, and D. Farin, "Platelet-based coding of depth maps for the transmission of multiview images," in *Proceedings of SPIE, Stereoscopic Displays and Applications*, vol. 6055, 2006, pp. 93–100.

[29] Y. Morvan, D. Farin, and P. de With, "Joint depth/texture bit-allocation for multi-view video compression," in *Proceedings of Picture Coding Symposium (PCS 2007)*, vol. 10, Lisboa, Portugal, Nov. 2007, p. 4349.

[30] K. Muller, A. Smolic, K. Dix, P. Merkle, and T. Wiegand, "Coding and intermediate view synthesis of multiview video plus depth," Nov. 2009, pp. 741–744.

[31] K. Müller, A. Smolic, K. Dix, P. Kauff, and T. Wiegand, "Reliability-based generation and view synthesis in layered depth video," *Multimedia Signal Processing (MMSP), IEEE International 10th Workshop on*, pp. 34–39, Oct. 2008.

[32] F. Pasteau, M. Babel, O. DÃ©forges, C. Strauss, L. BÃ©dat *et al.*, "Locally adaptive resolution (LAR) codec," *Recent Advances in Signal Processing*, pp. 37–48, 2010.

[33] M. Sarkis and K. Diepold, "Depth map compression via compressed sensing," in *Proceedings of the 16th IEEE international conference on Image processing*, 2009, pp. 737–740.

[34] J. Shade, S. Gortler, L. He, and R. Szeliski, "Layered depth images," in *Computer graphics and interactive techniques, SIGGRAPH*. New York, NY, USA: ACM, 1998, pp. 231–242. [Online]. Available: http://grail.cs.washington.edu/projects/ldi/

[35] H. R. Sheikh and A. C. Bovik, "Image information and visual quality," *Image Processing, IEEE Transactions on*, vol. 15, no. 2, pp. 430–444, 2006.

[36] S. Smirnov, A. Gotchev, S. Sen, G. Tech, and H. Brust, "3d video processing algorithms–part i," 2010.

[37] A. Smolic, K. Mueller, P. Merkle, C. Fehn, P. Kauff, P. Eisert, and T. Wiegand, "3D video and free viewpoint Video Technologies, applications and MPEG standards," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME06)*, 2006, pp. 2161–2164.

[38] A. Smolic, K. Müller, N. Stefanoski, J. Ostermann, A. Gotchev, G. Akar, G. Triantafyllidis, and A. Koz, "Coding algorithms for 3dtv - a survey," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, no. 11, pp. 1606–1621, Nov. 2007. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4373329

[39] G. J. Sullivan, P. Topiwala, and A. Luthra, "The h. 264/AVC advanced video coding standard: Overview and introduction to the fidelity range extensions," in *Presented at the SPIE Conference on Applications of Digital Image Processing XXVII Paper No*, vol. 5558, 2004, p. 53.

[40] M. Tanimoto, T. Fujii, K. Suzuki, N. Fukushima, and Y. Mori, "Reference softwares for depth estimation and view synthesis," Apr. 2008.

[41] ——, "Reference softwares for depth estimation and view synthesis, iso/iec jtc1/sc29/wg1," in *Archamps, France, Tech. Rep. M15377*, 2008.

[42] A. Vetro, S. Yea, and A. Smolic, "Towards a 3D video format for auto-stereoscopic displays," *Proceedings of the SPIE: Applications of Digital Image Processing XXXI, San Diego, CA, USA*, 2008.

[43] S.-U. Yoon, E.-K. Lee, S.-Y. Kim, and Y.-S. Ho, "A framework for representation and processing of multi-view video using the concept of layered depth image," *Journal of VLSI Signal Processing Systems for Signal Image and Video Technology*, vol. 46, pp. 87–102, Mar. 2007. [Online]. Available: http://vclab.gist.ac.kr/papers/01/2007/1_2007_SUYOON_A_Framework_for_Representation_and_Processing_of_Multi-view_Video_Using_the_Concept_of_Layered_Depth_Image.pdf

[44] S.-U. Yoon, E.-K. Lee, S.-Y. Kim, Y.-S. Ho, K. Yun, S. Cho, and N. Hur, "Coding of layered depth images representing multiple viewpoint video," *Picture Coding Symposium (PCS)*, vol. SS3-2, pp. 1–6, Apr. 2006. [Online]. Available: http://vclab.gist.ac.kr/papers/03/2006/3_2006_SUYOON_Coding_of_Layered_Depth_Images_Representing_Multiple_Viewpoint_Video.pdf

[45] C.-L. Zitnick, S.-B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 600–608, 2004. [Online]. Available: http://research.microsoft.com/~larryz/videoviewinterpolation.htm