Point Cloud Compression using Depth Maps

Arnaud Bletterer*[1,2]*, Frédéric Payan*[1]*, Marc Antonini*[1]*, Anis Meftah*[2]*;
[1] ***Laboratory I3S - University of Nice - Sophia Antipolis and CNRS (UMR 7271), France;*** [2] ***Cintoo3D, France***

Abstract

In this paper we investigate the usage of depth maps as a structure to represent a point cloud.The main idea is that depth maps implicitly define a global manifold structure for the underlying surface of a point cloud. Thus, it is possible to only work on the parameter domain, and modify indirectly the point cloud. We show that this approach simplifies local computations on the point cloud and allows using standard image processing algorithms to interact with the point cloud. We present results of the application of standard image compression algorithms applied on depth maps to compress a point cloud, and compare them with state-of-the-art techniques in point cloud compression. We also present a method to visualize point clouds in a progressive manner, using a multiresolution analysis of depth maps.

Introduction

3D data acquisition has been an active research field over the last years. High resolution devices are able to acquire scenes composed of billions of points. Clouds of points are often transformed in polygon meshes, one of the most popular representation for 3D models. In particular triangle meshes, simple, flexible and widely supported by graphics hardware. With high resolution acquisitions, the resulting meshes are huge. The handling and common processings such as transmission or visualization become complex, or just impossible, even with workstations. Therefore in many industrial domains, visualizing point clouds directly becomes common, and the compression of such data has been an active field of research for one decade.

Our current work focuses on the transmission and the visualization of massive point clouds on different types of devices (computers, tablets, smartphones...). The key problem is to adapt the quantity of information to transmit and/or display, in function of the performances of the user devices and the bandwidth limitations. In this context, data compression is a well-suited tool.

The problem with point clouds is that they are not structured and have no topology. They are difficult to handle, because of the lack of connectivity information. Even the basic notion of neighborhood for a point is problematic. Therefore all the prior methods of compression for point clouds build their own structure before encoding [11, 2, 13, 4, 5, 15, 8].

However, this regularity exists natively in the data generated by multiple 3D acquisition devices, namely *depth maps*. A depth map is a grayscale 2D image, representing the distance from each sample to the scanning device during an acquisition. Every non-black pixel is representing a point in the scene captured (black pixels representing the background).

A point is defined by its 3D coordinates $\{x, y, z\}$. In a depth map, a pixel is defined by one single coordinate $\{d\}$, the distance from the camera. Indeed, the image is sampled on a regular lattice, the coordinates $\{i, j\}$ of the pixel in the image are implicit, and do not need to be stored. To project points from the map to the $3D$ space, one matrix is needed for each depth map (16 real numbers). Its cost is negligible, in comparison with the map cost. Thus, depth maps are naturally more compact than point clouds, as long as $\frac{1}{3}$ of the total number of pixels represent points in $\mathbb{R}^3$.

Depth maps do not provide only geometric information. They also define a parameterization domain and a segmentation related to the different points of view (position and orientation of the system during acquisition). So numerous techniques of image processing can be used to process the associate point clouds. Moreover, their regular lattice enables efficient wavelet filtering, as in the classical 2D image domain.

Considering these nice properties, we choose to investigate the use of depth maps as an alternative representation of point clouds, and aim to develop algorithms based on this specific structure. In this paper, we present our preliminary works in the context of visualization, and of compression.

The rest of the paper is organized as follows. We first present some recent methods of compression for point clouds. The following section explains how a set of depth maps can be used as a parameterization domain for point clouds. Then, we present two scenarios where image-based algorithms can be used to process point clouds easily. First, we show how multiresolution analysis of a set of depth maps enables LOD and view-dependent visualization of point clouds. Second, we explore the use of image coders to compress a point cloud, by compressing its associated set of depth maps. In particular, we compare the RD (Rate Distortion) performances of several image coders and of one state of the art point cloud compression method.

State of the art

As introduced before, a point cloud is only a set of samples distributed in a $3D$ space. It has no structure, no topology. Therefore, in the domain of compression, all the recent methods use a multiscale structure to represent a point cloud before encoding it.

Sphere hierarchy In 2000, the authors of [11] suggest to handle massive datasets by using a bounding sphere hierarchy. Then, each sphere encodes its position relatively to its parent sphere.

Octree-based methods In 2002, the authors of [2] propose to use a high resolution voxel grid, and to progressively coarsen it by merging 8 ($2x2x2$) blocks. To reconstruct the initial resolution grid, one byte is used to select which ones of the 8 merged blocks contain points. Later, [13] and [4] present similar methods that embedd the point cloud in an octree. The bounding box of a point

cloud is recursively subdivided, and each cell is encoded with a prediction scheme depending on the configuration of the children cells.

Incremental simplification In 2004, a progressive decimation scheme for point clouds is proposed by [15]. Pairs of points are merged recursively, and a prediction scheme encodes the position of merged points.

Segmentation-based methods The same year, an original approach is proposed in [8]. The authors develop a method that segments a point cloud in several subsets, such as each subset can be parameterized by a height-field. Hence, any still image compression scheme can be used to encode these 2D data.

Theoretical foundations

The link between a point cloud defining a surface in $3D$ space, and a depth map is given by a projection matrix M. For each point $v \in \mathbb{R}^3$, it computes $p \in \mathbb{R}^2$, the position of the projection of v in a depth map. This transformation is invertible (M is an upper triangular matrix, with non-zero coefficients on the diagonal), and enables the embedding of 2D pixels in $3D$ space. The surface constructed by the embedding of every point of a depth map is called a *range surface*.

This projection procedure is equivalent to the notion of *parameterization* in differential geometry. A *range surface* S_k corresponds to a *patch* $C_k \in \mathbb{R}^3$ and a projection matrix M_k represents a parameterization function φ_k, linking a patch C_k to a parameterization domain $\Omega_k \in \mathbb{R}^2$ (representing a depth map). *Transition functions* $\tau_{\varphi_i \to \varphi_j}$ define the link between parametric coordinates of points belonging to a common intersection between two patches C_i and C_j :

$$\forall v \in C_i \cap C_j, \quad \varphi_j(v) = \tau_{\varphi_i \to \varphi_j}(\varphi_i(v)), \tag{1}$$

which is equivalent to :

$$\forall v \in S_i \cap S_j, \quad M_j * v = M_j * M_i^{-1}(M_i * v) \tag{2}$$

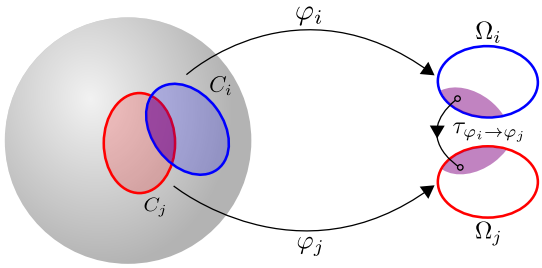


Figure 1. *Set of overlapping parameterizations (φ_i and φ_j) connected by transition functions $\tau_{\varphi_i \to \varphi_j}$[10]*

Given this link, we can consider a point cloud as simply being an embedding of a set of depth maps and storing only these ones in memory. This allows working in the parameterized domain to interact with point clouds, as done in [8] for compression, or [9] for spectral processing.

Visualization

As introduced before, visualizing a point cloud using depth maps is not more complex than visualizing the raw point cloud. When rendering a model, points are projected on the screen and become pixels. This is done by applying a matrix transform to all points of the model, which will compute their position in the rendered view. When rendering a point cloud from a set of depth maps, only one additional step is needed to rebuild the point cloud: the embedding of the pixels in $\mathbb{R}^3$. Considering that, we only keep depth maps and projection matrices in memory, instead of a set of 3D positions.

In this part we show how we can represent a point cloud progressively by applying a multiresolution analysis on each depth map.

Multiresolution analysis

The wavelet transform [7] is a time-frequency transformation, decomposing a signal λ_0 in

- a "low-frequency" signal λ_{-1}, approximating the original signal (average subsampling);
- a set of wavelet coefficients γ_{-1}, representing the "high-frequency" details removed to get the "low-frequency" signal.

This decomposition step is called *analysis*. It is an invertible operation, allowing the perfect reconstruction of the original signal. The inverse operation is called *synthesis*. In the case of images, a dyadic decomposition is often applied: only the low-frequency signal is decomposed at each level of resolution. This representation is relevant for transmitting an image progressively: a coarse version is first sent, and then the sets of coefficients are sent to reconstruct the original image progressively.

To perform a multiresolution analysis, we use a lifting scheme, which is an efficient implementation of the wavelet transform [14]. The lifting scheme is composed of 3 different steps:

- **Separation**. A signal λ_i is decomposed in two subsets λ_{i-1} and γ_{i-1} gathering respectively the even and odd samples (pixels in the case of an image) of the original signal;
- **Predict**. Odd samples values γ_{i-1} are predicted using an operator P from the values of λ_{i-1}. Prediction errors represent wavelet coefficients and are stored in γ_{i-1};
- **Update**. Values of λ_{i-1} are updated using an operator U based on the wavelet coefficients γ_{i-1} obtained.

One advantage of the lifting scheme is to obtain the inverse transform just by applying the preceding steps in the inverse order. This process is generally repeated on the low-frequency signal λ_{i-1} until a given level of resolution. In the context of 2D images, the lifting scheme is applied in a separable manner: it consists in applying the scheme on every line and then on every column, producing one low-frequency subband and three subbands of wavelet coefficients (Figure 2).

Progressive point cloud representation

We now use a lifting scheme applied to a set of depth maps defining a point cloud to get a lower resolution of it. The update operator U preserves global properties of a signal through the different levels of decomposition (average value for example). In our

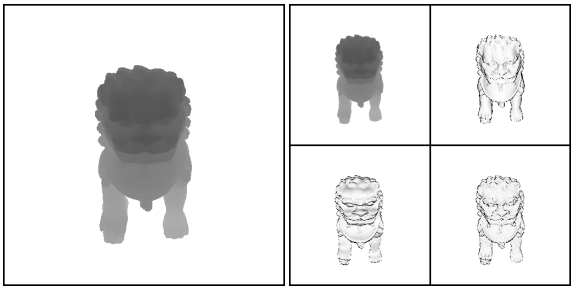


Figure 2. *Original depth map (left) and its wavelet transform (right).*

context, this operator may alter the shape of the point cloud in 3D space, by shifting severely the points of the lower resolution. We chose to disable this step. Hence, the lower resolution of a point cloud is inevitably a subset of the original points, and the fidelity to the initial shape is preserved.

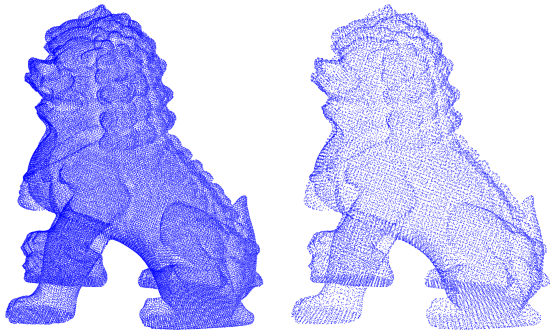


Figure 3. *Original point (left) and a subsampled version after one level of decomposition (right).*

Levels of details (LOD)

This structure allows LOD and progressive transmission: a first set of low-resolution depth maps is sent (via a network eventually) to the GPU, and then more or less details are sent in function of the device capabilities (and of the network limitations, if transmitted).

View-dependent visualization

The structure chosen is really suited for partial data visualization. As the point cloud is represented through different segmentations, one can choose to visualize a subset of segmentations only, depending on the current point of view during the visualization process. This decimates the point cloud really fast, and can achieve excellent results. For example, for a specific point of view, it is possible to visualize the same information with only some points (Figure 4). Given that the multiresolution analysis is applied independently on each depth map, it also possible to refine a point cloud partially, by decomposing each depth map at a different resolution. It is also possible to display only a part of the object, depending on the current point of view during the visualization process, as shown in Figure 4). The structure of depth maps enables a really fast decimation of a point cloud, and achieve excellent results.

Compression

To visualize a point cloud, it is necessary to have a file containing all the information concerning it. The size of this file becomes huge when reaching hundreds of million points.

Depth maps are grayscale images, thus they can be compressed similarly to other kind of images. Using our structure, compressing a point cloud consists in compressing each depth map independently. Considering that, we decide to investigate the use of different image coders to compress point clouds.

JPEG2000 [3] is a well-known image coding algorithm based on wavelets. It has some useful properties, like progressive decompression (by reconstructing only some levels during the multiresolution synthesis), lossless compression up to 16 bits per pixel, and offer a good tradeoff between bitrate and quantization error.

BPG [1] is a new image coder based on a subset of HEVC (intra-frame encoding). It has already shown some interesting results, producing files smaller than JPEG for a similar quality. This algorithm also offers the possibility to compress images losslessly, up to 12 bits per pixel.

Lossless compression

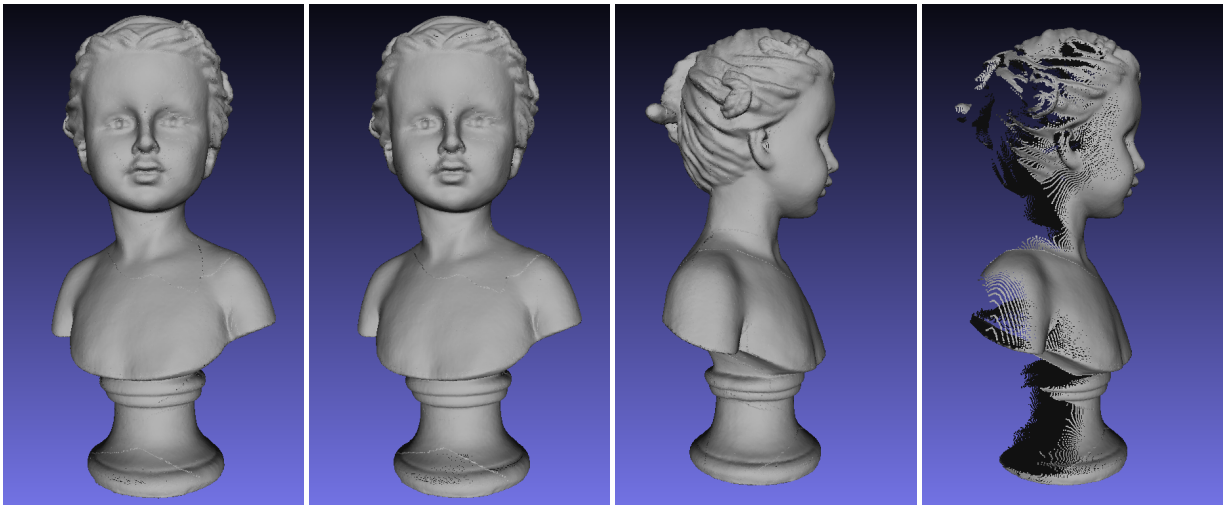
When restructuring point clouds, state-of-the-art methods quantize the data, choosing representatives of points contained in the leaf nodes. So even if a compressed point cloud can contain the same number of points as the original data (in other words, when each leaf contains 1 point only), the position of each point is not conserved, introducing some error. Instead, we consider the original data provided at the end of an acquisition as an input to our algorithm, where pixel intensities are quantized to 16 bits. We compare the bitrate obtained using different standard image compression algorithms. Concerning BPG, the actual implementation is not able to encode more than 12 bits per pixel. Thus we express the bitrate obtained with respect to the quantization error introduced (quantizing from 16 to 12 bits per pixel).

	Bits per point	
Model	*JPEG2000*	*BPG*
Bunny	6.236	3.275 ($7.225e^{-9}$)
Buste	6.603	3.469 ($7.744e^{-9}$)
Chinese	8.429	4.118 ($7.569e^{-9}$)
Dragon	7.537	4.009 ($8.649e^{-9}$)
Ramsesses	6.651	3.06 ($8.281e^{-9}$)
Sphere	3.693	2.364 ($5.041e^{-9}$)

Rates obtained after lossless compression of different models. BPG quantization error is indicated between parentheses (with respect to the bounding box of the original point cloud).

Lossy compression

In terms of lossy compression, we cannot compare our results on the same data used in state-of-the-art works. We consider depth maps at the original data instead of point cloud. Even if we can generate depth maps from 3D models using a rendering process, we will not have the same samplings as other methods (comparing MSE is meaningless in that case). Nonetheless, we can compare our approach with the one of [5], where the code has been provided through the Point Cloud Library [12]. The loss of quality is measured using *peak signal to noise ratio* (PSNR in dB) as a function of the rate (in bits per point). To evaluate the error, with use a similar measure as the other papers of the domain



(a) 860 932 points (b) 281 166 points (c) 860 932 points (d) 281 166 points

Figure 4. *View-dependent visualization of a point cloud. Selecting only a subset of depth maps reduces the number of points to display, for a similar visual result. For a specific point of view, visualizing every depth maps (a) or only a subset of them (b) is nearly the same in terms of quality. (d) shows the point cloud from a side view*

of point cloud compression. We measure the *root mean square error* $RMS(S,\hat{S})$ between the original point cloud S and the decompressed point cloud $\hat{S}$. The PSNR can be computed as :

$$20log_{10}\frac{dB}{max(RMS(S,\hat{S}),RMS(\hat{S},S))},$$

where dB is the bounding box diagonal of S. $RMS(S,\hat{S})$ is computed by finding, for each point v of S, its nearest neighbor $\hat{v}$ in $\hat{S}$.

We can see that our image-based method has much better results than the octree method of [5]. In this article, the authors targeted a real-time compression scheme, to visualize a dynamic point cloud. This method compresses a point cloud first, and then only sends the differences between two point clouds, to reduce the amount of data transmitted.

When comparing the different image-based coders, we can see that BPG has better results than JPEG2000 at low bitrates (Figure 5). On the other hand, JPEG2000 is able to reconstruct point clouds much more precisely, being really close to the original data (Figure **??**). This is mostly due to the fact that BPG cannot reconstruct perfectly the data, since it quantized the pixel values to 12 bits.

Progressive decompression

Concerning JPEG2000, we are really interested in the progressive decompression, as it allows using the same algorithm to compress data, and to obtain a progressive representation of point clouds. However this algorithm uses specific wavelets :

- biorthogonal Cohen-Daubechies-Fauveau 5/3 (CDF 5/3) for lossless compression,
- biorthogonal CDF 9/7 for lossy compression,

both updating the values of the low-resolution subband at each decomposition level. For depth maps this is inconvenient,

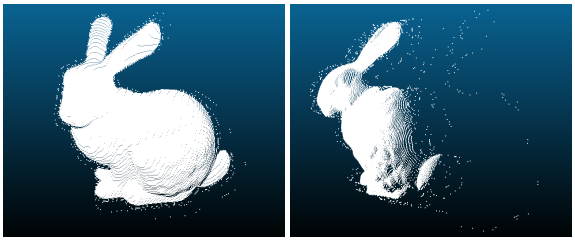


Figure 7. *Noise appears around the object when decompressing only until the penultimate level. Points near the silhouette of the model are moved due to the smoothing of the low-frequency subband.*

because they are never seen as images, but as their embedding in $\mathbb{R}^3$. Modifying these values alters the geometry of the point cloud making it noisy (Figure 7). Those schemes smooth the values in the low-frequency subbands (Figure 8).

It means that it is not possible to use the progressive aspect of JPEG2000 in this scenario, even in the lossless case. However this problem is only related to the fact that we were not reconstructing using all the wavelet coefficients. In the case of a full reconstruction, those problems only arise with high compression rates, where the effect of the quantization is too strong.

Figure 8. *Difference between a lifting scheme with (left) and without (right) using the update operator U. The update operator has a tendency to smooth regions having an important difference of intensities between neighbor pixels.*

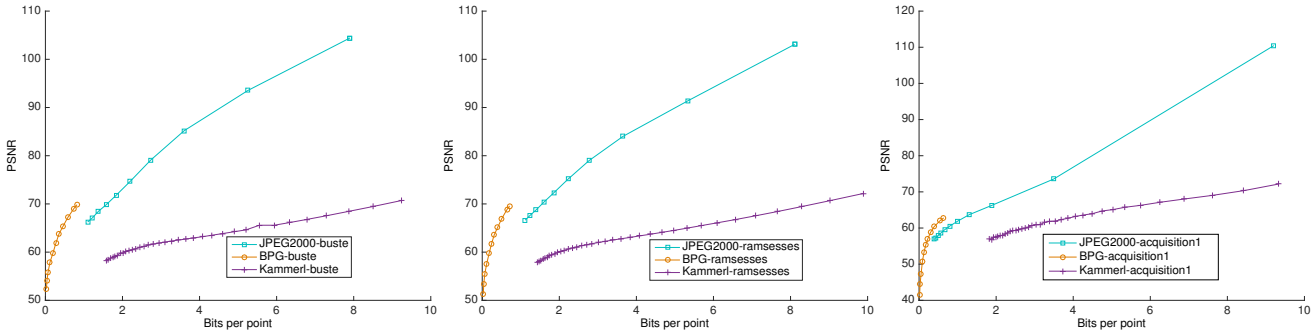
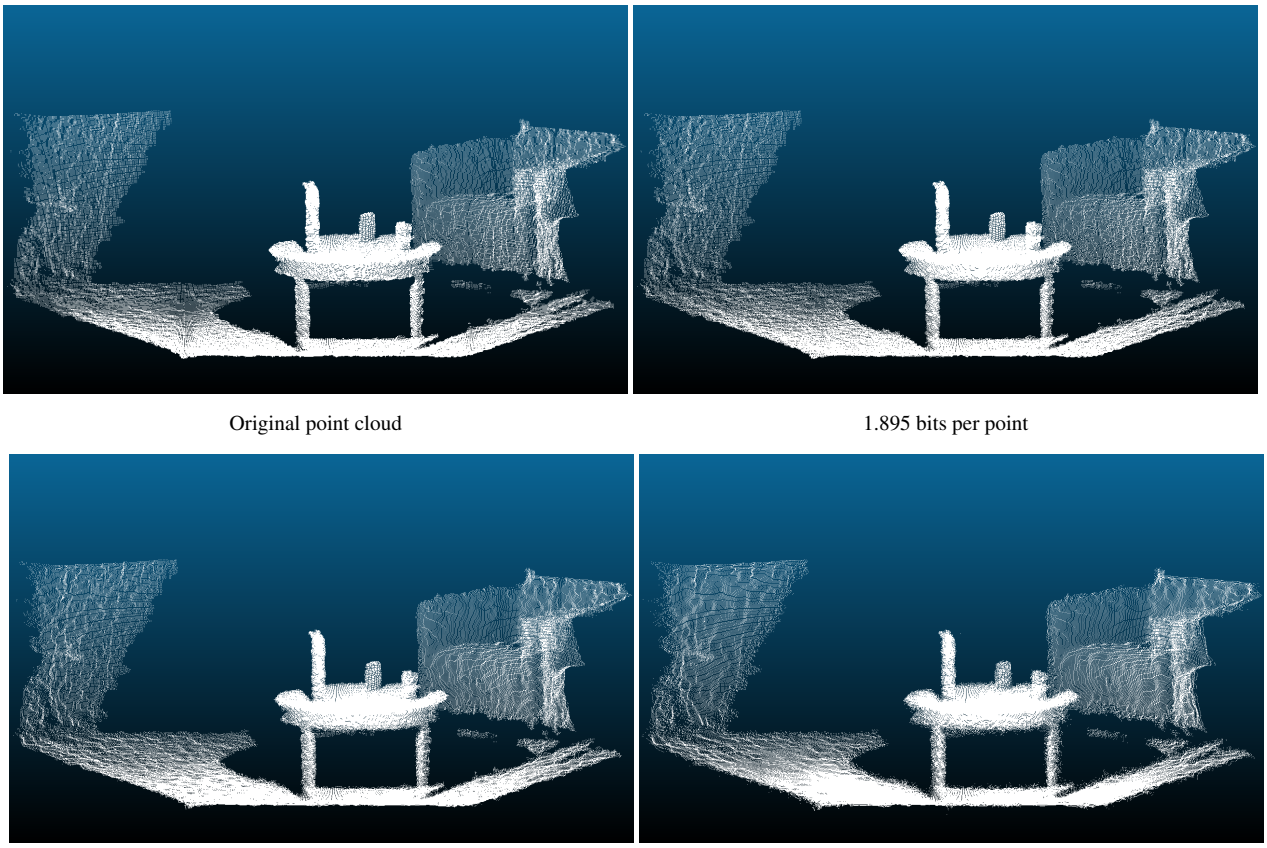


Figure 5. *Evolution of the PSNR as a function of the bitrate. JPEG2000 is able to reconstruct higher quality point clouds, but at equivalent quality, BPG has a much lower bitrate. In comparison to the work of [5], for a similar error, our approach is able to compress point clouds at a much higher rate. The right-side curves show how the error evolves on a real data (Kinect), using a depth map provided by [6].*



Original point cloud

1.895 bits per point

0.9872 bits per point

0.6644 bits per point

Figure 6. *Visualization of a point cloud (top-left) compressed at different rates (expressed in bits per point) using lossy JPEG2000.*

Conclusions

We have presented a novel representation of point clouds using a set of depth maps and projection matrices. This structure allows using standard image processing algorithms to interact with point clouds.

We have shown that a progressive representation of a point cloud can be built from a multiresolution analysis of depth maps.

Our results show that it is really interesting to use image-based coder to compress a point cloud. Especially knowing that any image-based coder can be used to encode depth-maps. But they also demonstrate that it is not possible to use some specificities of the coders (like progressive decompression of JPEG2000) due to the nature of their implementation. In our case, it would be interesting to have a wavelet coder not updating the values of the different low-frequency subbands. Thus it would be possible to visualize a point cloud resulting of a partially decoded file. Especially knowing that some parts of the point cloud are not visible, or too far from the camera.

Currently there is no control of the quality depending on a geometric error. Image-based coder like JPEG2000 minimize an

error for a specific target bitrate, but this error is computed in the image domain. But this is not optimal, because depth-maps are never seen as images, but only as their embedding in $\mathbb{R}^3$. It would be much more interesting to minimize a geometric error in an image-based coder.

Acknowledgments

This work is supported by a grant from *RÉGION PROVENCE ALPES CÔTE D'AZUR*.

References

[1] F. Bellard. Better portable graphics. `http://bellard.org/bpg/`.

[2] M. Botsch, A. Wiratanaya, and L. Kobbelt. Efficient High Quality Rendering of Point Sampled Geometry. Thirteeth Eurographics Workshop on Rendering, page 12, 2002.

[3] M. J. Gormish, D. Lee, and M. W. Marcellin. Jpeg 2000: overview, architecture and applications. In In Proc. of ICIP'2000, pages 29–32, 2000.

[4] Y. Huang, J. Peng, C. C. J. Kuo, and M. Gopi. A generic scheme for progressive point cloud coding. In IEEE Transactions on Visualization and Computer Graphics, volume 14, pages 440–453, 2008.

[5] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach. Real-time compression of point cloud streams. In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 778–785. IEEE, 2012.

[6] K. Lai, L. Bo, and D. Fox. Unsupervised feature learning for 3d scene labeling. In IEEE International Conference on on Robotics and Automation, 2014.

[7] S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 11(7):674–693, 1989.

[8] T. Ochotta and D. Saupe. Compression of point-based 3D models by shape-adaptive wavelet coding of multi-height fields. In Proc. Symposium on Point-Based Graphics, pages 103–112, 2004.

[9] M. Pauly and M. Gross. Spectral Processing of Point-Sampled Geometry. Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01, pages 379–386, 2001.

[10] N. Ray, W. C. Li, B. Lévy, A. Sheffer, and P. Alliez. Periodic global parameterization. ACM Trans. Graph., 25(4):1460–1485, Oct. 2006.

[11] S. Rusinkiewicz and M. Levoy. QSplat. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00, pages 343–352, 2000.

[12] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, May 9-13 2011.

[13] R. Schnabel and R. Klein. Octree-based Point-Cloud Compression. SPBG, 2006.

[14] W. Sweldens. Lifting scheme: a new philosophy in biorthogonal wavelet constructions. In SPIE's 1995 International Symposium on Optical Science, Engineering, and Instrumentation, 1995.

[15] M. Waschbüsch, M. Gross, F. Eberhard, E. Lamboray, and S. Würmlin. Progressive Compression of Point-Sampled Models. Eurographics Symposium on Point-Based Graphics, 2004.

Author Biography

Arnaud Bletterer received his M.Sc. degree in Computer Science in June 2014 from the University of Strasbourg (France). He is now doing his Ph.D in signal and image processing since October 2014 at he University of Nice-Sophia Antipolis (UNS, France), in the I3S/CNRS Laboratory. His research interests include geometry processing and rendering, in particular visualization and compression of point clouds.

Frédéric Payan received the Ph.D degree in signal and image processing in December 2004 from the University of Nice-Sophia Antipolis (UNS, France). He is now assistant professor at the UNS, in the I3S/CNRS Laboratory. His research interests include geometry processing, in particular compression, remeshing, and sampling of surfaces.

Marc Antonini received the Ph.D degree in electrical engineering from the University of Nice-Sophia Antipolis (France) in 1991. He joined the CNRS in 1993 at the I3S laboratory where he is "Directeur de Recherche" since 2004. His research interests include image, video and geometry coding using wavelet analysis. He is the author of more than 200 papers and 10 patents. He is a co-founder of Cintoo3D, a Start-Up specialized in 3D streaming solutions.