FAST POINT CLOUD COMPRESSION VIA REVERSIBLE CELLULAR AUTOMATA BLOCK TRANSFORM

S. Milani

University of Padova, Dept. of Information Engineering,
e-mail: simone.milani@dei.unipd.it

ABSTRACT

Augmented and mixed reality applications require efficient tools permitting the compression and the visualization of 3D object at a limited computational cost. To this purpose, 3D point cloud representations have been widely used, together with an octree-based hierarchical organization of data that enables a multi-resolution visualization. This paper presents a voxel coding strategy based on a hierarchical Cellular Automata block reversible transform which permits obtaining a multi-resolution representation of the input volume and a higher compression gain with respect to the state-of-the-art octree strategies. The proposed solution also proves to be more flexible in defining multiple layers and more effective in preserving 3D volume quality when the stream is partially decoded.

Index Terms— 3D volumes, point clouds, voxels, octree, multi-resolution

1. INTRODUCTION

Virtual and augmented reality applications have recently highlighted the need for effective and lightweight strategies that are capable of transmitting and handling live 3D dynamic content that can be viewed from any viewpoint.

Such contents have been so far represented using either polygonal meshes [1] or point clouds, along with their associated colour information. Meshes represent surfaces very well, but they are not robust to noise and other artifacts typically found in live captures. Point clouds are instead less sensitive to noise but provide a sparse model of 3D volumes. Starting from these two possibilities, the need for real time acquisition and visualization have led researchers to represent point clouds of 3D data into hierarchically organized voxel structures. The volume to be coded is progressively approximated by recursively partitioning 3D space into finer volumes enclosing the model to be coded. Such process can be represented by the coding tree reported in Fig. 2.

First proposals date back to the 80's when Sparse Voxel Octrees (SVOs) were introduced. More recently, SVOs have proved highly efficient for real time encoding at high frame rates of both geometry [2] and color attributes [3]. The limited requirements in terms of computational complexity have made them quite suitable for the efficient compression [4] and rendering [5,6] of dynamic point clouds. As a matter of fact, octree is nowadays the compression strategy adopted in the PCL library [7].

Despite being lightweight and flexible, octree representation proves to be inefficient in terms bit occupancy and scalability. In order to obtain a lower resolution representation from a coded oct tree stream, it is possible to terminate the reconstruction procedure at a lower depth level of the coding tree. This leads to an upper bound approximation of the coded volume which could produce unpleasant artifacts when the depth reconstruction is very low. This paper proposes an alternative to octree partitioning based on a 3D Cellular Automata (CA) reversible transform, which enables to decompose hierarchically voxel volumes leading to a better coding and representation efficiency. The proposed transform operates on $2 \times 2 \times 2$ binary voxel grids, and it can be designed in order to optimize both the visual rendering and the compression efficiency. As a result, visual appearance significantly improves with partial decoding of the bitstream (see Fig. 1), together with the overall compression efficiency.

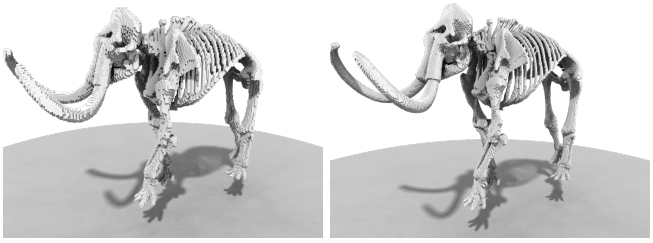

Fig. 1. Reconstructed model with rate 120 kbit for Octree coder (left) and for the proposed solution (right).

In the following, Section 2 overviews the main recently published works on voxel coding, while Section 3 shows how voxel volumes can be modelled using CA. Section 4 describes the proposed coding solution, whose performance is measured by the experimental results reported in Section 5. Section 6 draws the final conclusions.

2. RELATED WORKS

The widespread of augmented reality applications have highlighted the need to coding and editing tools processing a volumetric representation of 3D data, such as lighfield, meshgrid, or point clouds. Most of the previous 3D coding solutions were intended to a panoramic representation of data, which proved to be suitable for 3DTV or virtual reality but limited for AR applications. Among the different 3D data representations, point clouds have proved to be effective in terms of robustness to noise and effortless processability [8]. As a matter of fact, the compression of such data have recently raised a particular interest in coding experts. A few initial solutions adopted a 2D wavelet transform based scheme [9] or a multi-resolution decomposition of the 3D points [10].

The work has been supported by the 3D CloudVision projects, funded by the University of Padova, Italy.

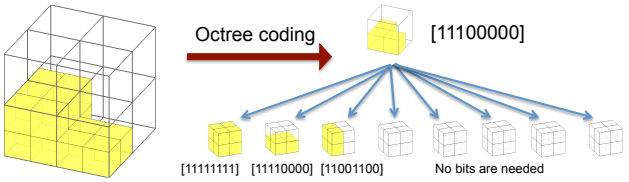

Fig. 2. Octree coding of of a $4 \times 4 \times 4$ voxel block.

Following works have instead relied on a decomposition of the 3D space an octree hierarchical division. The initial point cloud is voxelized into a cube of voxels, which is then recursively divided into 8 equally-sized subvolumes. A flag bit signals for every subvolume whether it contains some points to be coded. In case, no points can be found, recursion stops. An example is reported in Fig. 2. In the decoding phase, it is possible to reconstruct the object at a lower resolution by terminating the decoding at one of the upper levels of the coding tree. This partitioning is very simple to obtain, requires a limited computational effort [6], defines a hierarchical representation of the objects, and can be employed to code different types of data and attributes, like color information [3, 11]. Moreover, octree decomposition has proved to be effective when dealing with large amount of points [12] thanks to the possibility of reconstructing the model at different resolutions. Some approaches have also tried to improved the octree decomposition by adopting a dynamic depth adaptation whenever no further subdivision are required [13]. Other works combine the octree solution with graph-based transforms [11, 14] in order to decorrelate the signal. In [15], a region-adaptive transform is introduced in order to effectively compress colour information. When point clouds are spatially and temporally redundant, it is possible to exploit block based prediction to reduce the amount of information to be written in the stream [11,16]. The solution in [17] adopts the motion estimation prior to voxelization in order to minimize the amount of prediction residual to be coded.

Despite octree enables a hierarchical representation of the input data, it is possible to notice that the lower resolution representations resulting from a partial decoding of the bitstream define an upper-bound representation of the volume (see the example of Fig. 2). Moreover, the amount of possible rate-accuracy points that can be obtained are quite limited. A more effective solution can be found by introducing a hierarchical transform which permits specifying different quality levels together with ensuring a high compression gain. So far, transform-based compressor [11, 14, 15] have been targeting colour information since traditional transforms work well on integer or real data. When dealing with binary values, the possibility of energy concentration are largerly reduced and compression becomes harder. In this work, a geometry-coding transform modelling voxel cube as a lattice of Cellular Automata is proposed. Like previous 2D solutions [18], the transform enables a multi-layer representation of the input data, as well as higher coding gains with respect to state-of-the-art solutions.

3. CELLULAR AUTOMATA BLOCK TRANSFORMS

3.1. Reversible Block Cellular Automata

The proposed coding strategy relies on modelling the voxel space as a block cellular automata (CA) structure, where each voxel is a cellular automaton that can assume two possible states (0 or 1). The lattice of CA is divided into non-overlapping blocks of size $2 \times 2 \times 2$ (Necker neighbourhood); therefore, it is possible to define the configuration of each block $\mathbf{b} = [s_0 \dots s_7]$ (where s_i is the state of the i-th automaton/voxel in the block) using an 8-bits integer. As a matter of fact, the configurations $\mathbf{b}$ span in the range $[0, 255]$.

Within each block, the state of a single automaton evolves according to the value of the states of the other automata within the same block. This evolution can be defined by a transform $\mathbf{b}^P = P(\mathbf{b})$, which maps 8-bits states $\mathbf{b}$ into other 8-bits state $\mathbf{b}^P$ (see Fig. 3) and operates independently on each neighbourhood. In simpler terms, the CA-transform operates an invertible-remapping of vectors $\mathbf{b} \in \{0, 1\}^8$ which allows to organize the data more efficiently.

Similarly to other block CA (like Critters [19]), after an initial transform step, corner automata within each block are grouped together into new voxel cubes with halved dimensions. After this reordering, the transform can be operated again on this new smaller volume. This operation is depicted in Fig. 3, where corner automata in each block are grouped together by the reordering operations. Following a wavelet-like hierarchical decomposition, the transform and reordering process can be operated multiple times on the so called DC sub-volume (i.e., the sub-volume defined by the upper-left automaton/voxel of the first plane) until a single CA block remains.

Since reversibility must be ensured in transform design (see [20]), final transformation results in a permutation of the states which aims at achieving a more compact representation of the voxels, i.e., minimizing the number of automata/voxels with state equal to 1. This purpose can be achieved designing a transform so that the evolution of the states mimick the energy-concentration behaviour of a Discrete Wavelet Transform (DWT).

In order to compute the energy of each CA block, the state of each automata can be associated to an Ising model (spin *up* or *down*). The resulting energy can be generically modelled by Potts energy equation

$$H_p = -J_p \sum_{<i,j>} \delta(s_i, s_j) \tag{1}$$

where s_i and s_j are two adjacent cells (i.e., belonging to the same block), and $\delta(\cdot)$ is the Kronecker delta function. The parameter J_p is a coupling constant. In simpler words, the resulting block energy depends on the number of CA with the same state. As a matter of fact, a DWT-like CA transform would keep the energy approximately unaltered, while minimizing the number of states equal to 1 (the spins are moved from 1 to 0).

Assuming that $n_1(\mathbf{b})$ and $n_0(\mathbf{b})$ respectively refer to the number of states equal to 1 and 0 for $\mathbf{b}$, an efficient transform would make $n_0(\mathbf{b}^P)$ increase. This imply that the number of states equal to 1 should be converted into 0s, while state 0s should be unaltered. This sort of state inversion is only approximate since block configurations with minimum number of states equal to 1 are limited.

3.2. Transform design

According to the principles described in the previous subsection, it is possible to design the permutation $P(\cdot)$ so that some properties are verified:

a) $P(0) = 0$;

b) $P(255) = 1$;

c) highly-probable blocks $\mathbf{b}$ with $n_1(\mathbf{b}) > 4$ should be converted to $\mathbf{b}^P$ s.t. $n_0(\mathbf{b}^P) = n_1(\mathbf{b})$;

d) less probable blocks $\mathbf{b}$ with $n_0(\mathbf{b}) > 4$ should be converted to $\mathbf{b}^P$ s.t. $n_1(\mathbf{b}^P) = n_0(\mathbf{b})$;

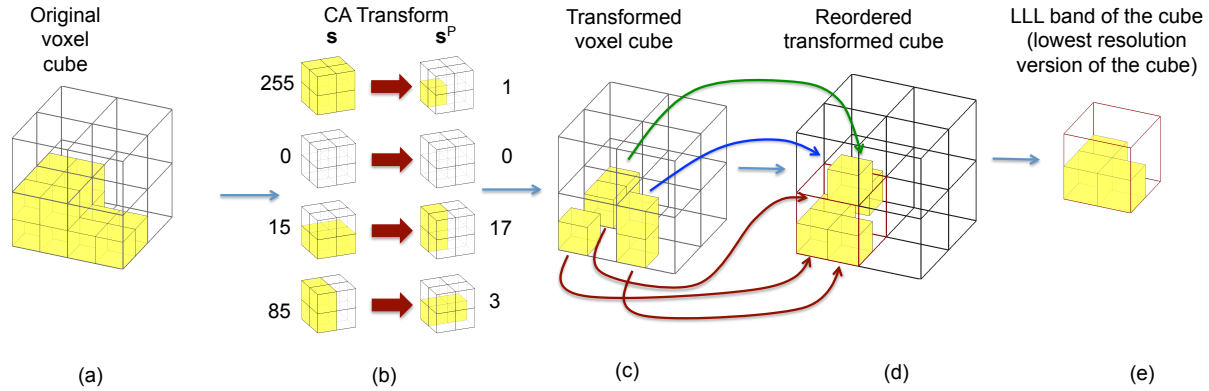

Fig. 3. Example of Cellular Automata block transformation of a $4 \times 4 \times 4$ voxel block.

e) highly-probable blocks $\mathbf{b}$ with $n_0(\mathbf{b}) > 4$ should be converted to $\mathbf{b}^P$ s.t. $n_0(\mathbf{b}^P) = n_0(\mathbf{b})$;

f) highly-probable blocks $\mathbf{b}^P$ should have $s_i^P = 1$ for small i values.

Note that requirements (a) and (b) imply that an empty block remains empty, while a full block is converted into a block with $s_0^P = 1$ and $s_i^P = 0 \; \forall i \neq 0$. This resembles the transformation operated by DWT which would concentrate the energy of a smooth surface on a single coefficient. The sub-cube created taking the s_0^P states of all the blocks lead to an halved resolution version of the original point cloud model. Note also that design principles could be expanded including some requirements related to the visual appearance of the lower resolution model defined by the s_0^P states. As a matter of fact, the design of $P(\cdot)$ could be driven by both bit rate and visual quality optimization needs.

Given these requirements, it is possible to design a transform $P(\cdot)$ optimized on the statistics of specific point cloud to be coded or on a set of training models.

In the proposed implementation, the design procedure computes the statistics of values $\mathbf{b}$ for a set of training voxelized models. At first, the mappings defined by points (a) and (b) are set. Then, vectors $\mathbf{b}$ s.t. $n_1(\mathbf{b}) > 4$ are ordered in decreasing probability order and mapped to $\mathbf{b}^P$, where $b_0^P = 1$ and $\mathbf{b}^P$ are assigned considering increasing values of $n_1(\mathbf{b}^P)$. Similarly, vectors $\mathbf{b}$ s.t. $n_1(\mathbf{b}) < 5$ are mapped to $\mathbf{b}^P$, where $b_0^P = 0$ and the most probable $\mathbf{b}^P$ are linked to vectors $\mathbf{b}^P$ with the smallest $n_1(\mathbf{b}^P)$. As for the vectors $\mathbf{b}$ s.t. $n_1(\mathbf{b}) = 4$, they are assigned to the remaining $\mathbf{b}^P$ following the previous probability-driven strategy. The voxelized 3D models used to compute the statistic distribution of $\mathbf{b}$ are `hockey`, `model`, and `city` (see Section 5).

In the following section, the whole coding engine will be described.

4. THE PROPOSED POINT CLOUD CODER

At first, the input voxel cube C^N with size $N \times N \times N = N^3$ is partitioned into non-overlapping $2 \times 2 \times 2$ blocks, whose configuration can be described by the states $\mathbf{b}(u, v, z)$ (where (u, v, z) are the block coordinates). Each value $\mathbf{b}(u, v, z)$ is then converted into $\mathbf{b}^P(u, v, z) = P(\mathbf{b}(u, v, z))$, and all the states $s_i^P(u, v, z)$ are gathered into a new voxel cube $C_i^{N/2}$ with size $N/2 \times N/2 \times N/2$. Mutuating the same terminology of DWT coding, in the following $C_0^{N/2}$ will be called DC subband and $C_i^{N/2}$, $i \geq 1$, AC subbands.

At this point, subbands $C_i^{N/2}$, $i = 1, \ldots, 7$, are coded in the bitstream using the following procedure. At the beginning the volume $E_{\overline{0}}$ is generated so that $E_{\overline{0}}(u, v, z) = 1$ if $n_1(\mathbf{b}^P(u, v, z)) - s_0^P(u, v, z) > 0$. Then, 6 additional volumes E_k are generated such that $E_k(u, v, z) = 1$ if $n_1(\mathbf{b}^P(u, v, z)) - s_0^P(u, v, z) = k$. The parameter k defines the number of states equal to 1 in $\mathbf{b}^P(u, v, z)$ excluding $s_0^P(u, v, z)$ and varies in the range $[0, 6]$. In this way, $E_k(u, v, z)$ signals which locations (u, v, z) have k non-zero subband components, i.e., $\sum_{i=1}^{7} C_i^{N/2}(u, v, z) = k$.

At first, $E_{\overline{0}}$ is coded using the standard octree strategy. Then, an iterative coding process signals the value E_k (k going from 1 to 6) for those locations where $E_{\overline{0}}(u, v, z) > 0$. Finally, for all the coordinates (u, v, z) s.t. $E_k(u, v, z) = 1$ the coder writes the integer corresponding to the string $[C_1^{N/2}(u, v, z) \ldots C_7^{N/2}(u, v, z)]$. Before coding matrices E_k and $C_i^{N/2}$, a binary flag signals whether there is significant information in the matrix or not. Using this escape variable, it is possible to compact the final bit streams.

The DC subband $C_i^{N/2}$ can be further decomposed into 8 subbands $C_i^{N/4}$ iterating the same procedure. When the coder decides to end the decomposition procedure, C_0^M is coded using a standard octree algorithm.

Fig. 4 reports a coding example for the AC subbands of Fig. 3 (d). It is possible to notice that octree strategy requires 32 bits to code the whole block, while the proposed solution requires only 24.

The whole coding scheme is summarized by the following pseudo-code.

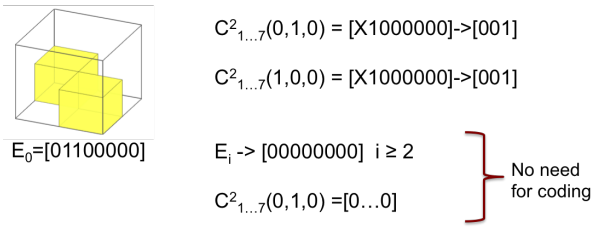

Fig. 4. Coding of of a $4 \times 4 \times 4$ voxel block.

Starting voxel cube C^N
repeat
  transform C^N and generates $C_i^{N/2}$, $i = 0, \ldots, 7$.
  compute $E_{\overline{0}}$; octree coding of $E_{\overline{0}}$.
  for k going from 1 to 7 **do**
    for (u, v, z) **do**
      if $E_{\overline{0}}(u, v, z) == 1$ **then**
        write $E_k(u, v, z)$
      end if
    end for
    for (u, v, z) **do**
      if $E_k(u, v, z) == 1$ **then**
        write $[C_1^{N/2}(u, v, z) \ldots C_7^{N/2}(u, v, z)]$
      end if
    end for
  end for
until no more decomposition of C_0^M.
octree coding of C_0^M.

The generated bitstream is then entropy coded using an adaptive arithmetic coder which permits achieving extra compression performance. Three different context models were used in this case: one for the octree coding, a second for the compression of E_k, and the third related to coding $[C_1^{N/2}(u, v, z) \ldots C_7^{N/2}(u, v, z)]$.

5. EXPERIMENTAL RESULTS

In order to evaluate the performance of the proposed strategy, experimental tests considered an extensive set of point clouds generated using different algorithms and devices. Some of the voxel cubes were obtained from computer-generated 3D models, others were acquired using a 3D laser scan or a Time-of-Flight cameras. Each model was voxelized into a volume of $512 \times 512 \times 512 = 512^3$ voxels. Volumes hockey, inukshuk, and model were obtained from [21]; lucy is from [22]; supernova, mammuth, crab, sculpture are from [23]; city, house, minecraft, dragon, castle are from [24].

The voxel cubes were compressed using an octree strategy combined with an arithmetic coder in order to ensure a fair comparison between the two strategies. Then, the proposed solution was tested on the same data using different decomposition levels. The adopted transform was computed considering block statistics from models hockey, crab and city; applying the rules described in Section 3.2 a permutation was obtained and employed for all the other models. Table 1 reports the bit rates obtained with different numbers of decomposition levels: it is possible to notice that the gain in coding efficiency does not relevantly increase after $3 - 4$ decomposition stages. Moreover, it is also possible to notice that the proposed solution permits reducing the coded bit rate of 34 % on average with respect to the standard octree strategy.

The flexibility of the proposed coding scheme was tested considering the quality of the model hockey decoded at different bit rates. More precisely, the distortion in the reconstructed model via the percentage of erroneously-reconstructed pixels was measured as

$$e = \frac{XOR(C_R^N, C^N)}{N^3} \quad (2)$$

where C_R^N is the reconstructed volume at the original resolution.

Figure 5 reports the distortion e as a function of the decoded bit rate. It is possible to appreciate a higher compression efficiency in the trade-off between accuracy and coded bit rate. Moreover, the number of possible reconstruction points for the proposed approach is higher. Further visualizations are available at the web page [25].

Table 1. Coded bit rate for Octree and the proposed coder

Model	Octree	Proposed			
		1	2	3	4
hockey	470	348 (26 %)	332 (29 %)	321 (32 %)	320 (32 %)
supernova	1222	948 (22 %)	949 (22 %)	916 (25 %)	913 (25 %)
lucy	637	458 (28 %)	437 (31 %)	423 (33 %)	421 (34 %)
mammuth	504	342 (32 %)	329 (35 %)	318 (37 %)	316 (37 %)
house	628	347 (45 %)	332 (47 %)	319 (49 %)	318 (49 %)
inushuk	2.6	0.67 (74 %)	0.61 (77 %)	0.60 (77 %)	0.60 (77 %)
minecraft	398	261 (34 %)	233 (41 %)	224 (44 %)	223 (44 %)
crab	655	450 (31 %)	437 (33 %)	422 (36 %)	421 (36 %)
dragon	296	191 (37 %)	182 (39 %)	174 (41 %)	173 (41 %)
model	270	207 (23 %)	198 (27 %)	193 (28 %)	192 (29 %)
sculpture	1363	931 (32 %)	926 (32 %)	903 (34 %)	898 (34 %)
city	1688	769 (51 %)	769 (54 %)	736 (56 %)	734 (56 %)
castle	5260	2153 (59 %)	1955 (63 %)	1955 (63 %)	1955 (63 %)

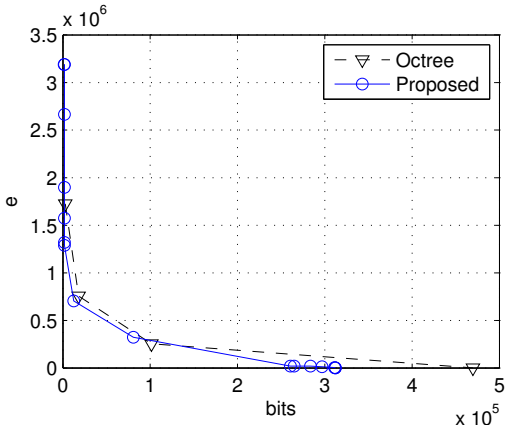

Fig. 5. Error e vs. bit rate for the model hockey.

In the end, the computational complexity of the proposed approach was compared with its Octree counterpart. In its current implementation, the amount of calculation increases up to 90 % with respect to Octree strategy although it still runs in real time. Further optimization can be applied such as generating the streams related to E_k and $C_i^{N/2}$ using a Finite State Machine (implemented via a look-up table).

6. CONCLUSIONS

The paper has presented a novel point cloud coding strategy which employs voxelization and a recursive block Cellular Automata transform. The proposed solution permits generating a multilayer stream that presents more reconstruction points and a higher coding efficiency with respect to that generated by state-of-the-art octree coding solutions. Moreover, the accuracy of partially-reconstructed models improves as well. Future work will be devoted in investigating alternative types of transform building rules and including temporal prediction in the coding schemes in order to compress dynamic point clouds. Moreover, computational complexity will be improved by implementing most of the coding routines via look-up tables.

7. REFERENCES

[1] Jingliang Peng, Chang-Su Kim, and C.-C. Jay Kuo, "Technologies for 3d mesh compression: A survey," *Journal of Visual Communication and Image Representation*, vol. 16, no. 6, pp. 688 – 733, 2005.

[2] Ruwen Schnabel and Reinhard Klein, "Octree-based point-cloud compression," in *Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics*, Aire-la-Ville, Switzerland, Switzerland, 2006, SPBG'06, pp. 111–121, Eurographics Association.

[3] Y. Huang, J. Peng, C. C. J. Kuo, and M. Gopi, "A generic scheme for progressive point cloud coding," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 440–453, March 2008.

[4] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 778–785.

[5] Jan Elseberg, Dorit Borrmann, and Andreas Nchter, "One billion points in the cloud ? an octree for efficient processing of 3d laser scans," {*ISPRS*} *Journal of Photogrammetry and Remote Sensing*, vol. 76, pp. 76 – 88, 2013, Terrestrial 3D modelling.

[6] A. Kuhn and H. Mayer, "Incremental division of very large point clouds for scalable 3d surface reconstruction," in *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, Dec 2015, pp. 157–165.

[7] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1–4.

[8] D. Thanou, P. A. Chou, and P. Frossard, "Graph-based compression of dynamic 3d point cloud sequences," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1765–1778, April 2016.

[9] Tilo Ochotta and Dietmar Saupe, "Compression of point-based 3d models by shape-adaptive wavelet coding of multi-height fields," in *Proceedings of the First Eurographics Conference on Point-Based Graphics*, Aire-la-Ville, Switzerland, Switzerland, 2004, SPBG'04, pp. 103–112, Eurographics Association.

[10] O. Devillers and P. M. Gandoin, "Geometric compression for interactive transmission," in *Proceedings Visualization 2000. VIS 2000 (Cat. No.00CH37145)*, Oct 2000, pp. 319–326.

[11] C. Zhang, D. Florencio, and C. Loop, "Point cloud attribute compression with graph transform," in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct 2014, pp. 2066–2070.

[12] O. Martinez-Rubi, S. Verhoeven, M. Van Meersbergen, M. Schtz, P. Van Oosterom, R. Gonalves, and T. Tijssen, "Taming the beast: Free and open-source massive point cloud web visualization," in *Proceedings of Capturing Reality Forum 2015, 23-25 November 2015, Salzburg, Austria*, Nov. 2015.

[13] K. Wenzel, M. Rothermel, D. Fritsch, and N. Haala, "An out-of-core octree for massive point cloud processing," in *Proc. of IQmulus 1st Workshop on Processing Large Geospatial Data*, 2014, pp. 53–60.

[14] P. A. Chou and R. L. de Queiroz, "Gaussian process transforms," in *2016 IEEE International Conference on Image Processing (ICIP)*, Sept 2016, pp. 1524–1528.

[15] R. L. de Queiroz and P. A. Chou, "Compression of 3d point clouds using a region-adaptive hierarchical transform," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3947–3956, Aug 2016.

[16] Rufael Mekuria, Kees Blom, and P Cesar, "Design, implementation and evaluation of a point cloud codec for tele-immersive video," *IEEE Transactions on Circuits and Systems for Video Technology*, 2016.

[17] Eduardo Pavez, Philip A. Chou, Ricardo L. de Queiroz, and Antonio Ortega, "Dynamic polygon cloud compression," *CoRR*, vol. abs/1610.00402, 2016.

[18] L. Cappellari, S. Milani, C. Cruz-Reyes, and G. Calvagno, "Resolution scalable image coding with reversible cellular automata," *Image Processing, IEEE Transactions on*, vol. 20, no. 5, pp. 1461–1468, May 2011.

[19] T. Toffoli and N. Margolus, *Cellular Automata Machines: A New Environment for Modeling*, MIT Press, 1987.

[20] Jarkko Kari, "Reversibility of 2d cellular automata is undecidable," *Physica D: Nonlinear Phenomena*, vol. 45, no. 1, pp. 379 – 385, 1990.

[21] LMI Technologies, "KScan3D Gallery," http://www.kscan3d.com/gallery/, 2016.

[22] Stanford Computer Graphics Laboratory, "The Stanford 3D Scanning Repository," https://graphics.stanford.edu/data/3Dscanrep/, 2016.

[23] The Smithsonian Institution, "Smithsonian X 3D repository," https://3d.si.edu/, 2016.

[24] TF3DM, "Gallery," http://tf3dm.com/, 2016.

[25] Simone Milani, "CA for PCL coding Page," http://www.dei.unipd.it/ sim1mil/capcl/, 2017.