A Transform Coding Strategy for Dynamic Point Clouds

Simone Milani *Member, IEEE*, Enrico Polo, Simone Limuti

***Abstract*—The development of real-time 3D sensing devices and algorithms (e.g., multiview capturing systems, Time-of-Flight depth cameras, LIDAR sensors), as well as the widespreading of enhanced user applications processing 3D data, have motivated the investigation of innovative and effective coding strategies for 3D point clouds. Several compression algorithms, as well as some standardization efforts, has been proposed in order to achieve high compression ratios and flexibility at a reasonable computational cost.**

This paper presents a transform-based coding strategy for dynamic point clouds that combines a non-linear transform for geometric data with a linear transform for color data; both operations are region-adaptive in order to fit the characteristics of the input 3D data. Temporal redundancy is exploited both in the adaptation of the designed transform and in predicting the attributes at the current instant from the previous ones. Experimental results showed that the proposed solution obtained a significant bit rate reduction in lossless geometry coding and an improved rate-distortion performance in the lossy coding of color components with respect to state-of-the-art strategies.

***Index Terms*—dynamic point cloud compression, cellular automata, transform coding, octree, voxel color**

I. Introduction

The recent development of real-time 3D acquisition devices and algorithms has allowed the inclusion of dynamic three-dimensional models in a wide range of applications, spanning from augmented reality to autonomous navigation. This availability has also highlighted a new challenging problem to the attention of researchers: enabling effective and versatile fruition of such contents. In fact, the amount of generated data tends to be quite large and heterogeneous, depending on the acquiring devices or algorithms, as well as on the target applications. As a matter of fact, several efforts have been recently entailed to investigate new coding solutions that enable an effective and versatile compression and transmission of 3D visual information.

Multiview-plus-depth video sequences are among the first 3D visual signals to be considered. They consist in low-level 2D video and depth signals from 3D scenes that can be efficiently compressed using the multiview extensions of traditional video coding schemes (e.g., MV-HEVC [1]). Such systems have recently been applied to the compression of light-field image and video signals, although the efficiency of these data representations tend to be limited when referring to single 3D objects. For this reason, single dynamic 3D models have been so far represented using either polygonal meshes [2] or point clouds, along with their associated color information. Meshes represent surfaces very efficiently, but they are not robust to noise and other artifacts, which are typically generated in live 3D captures. Moreover, the creation of high quality 3D meshes requires significant amounts of calculation because of polygonal fitting and refining operations.

The authors are with University of Padova, Dept. of Information Engineering, via Gradenigo 6/B, 35131 Padova, Italy, e-mail: simone.milani@unipd.it, {enrico.polo,simone.limuti}@studenti.unipd.it. This work has been partially-supported by the POR-FESR project "Rebuilding The Past" and by the SID 2018 project "SartreMR" prot. BIRD187839.

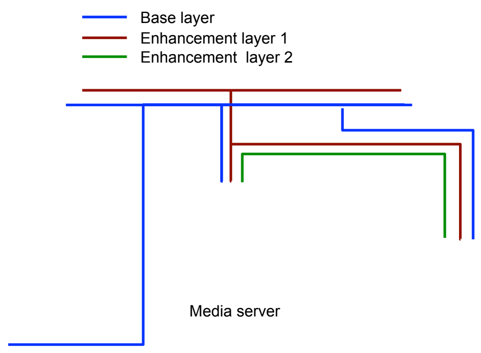

Fig. 1. Streaming of a dynamic 3D content in a mixed or augmented reality application for the sequence `longdress`.

On the other hand, dynamic point clouds (PCs) provide a less effective representation of 3D surfaces, which are sometimes approximated by sparse sets of 3D points depending on the acquisition systems. However, such models are less sensitive to noise and easier to generate in real time: most of the 3D sensing devices (e.g., Time-of-Flight sensors [3], LIDAR [4], etc.) are able to acquire a dynamic point cloud at significant acquisition rate (about 30 frame/s). From these premises, recent works have focused their attention on the compression and visualization of PCs [5], [6]; on Oct. 2017, MPEG-I committee has started an ongoing standardization activity aimed at defining a coding format (lossless and lossy) for dynamic point cloud [7].

The need for versatility and efficiency has focused many of the investigation efforts towards hierarchical and scalable coding solutions [8]. The same PC model needs to be deployed to different terminals with heterogeneous upload and download capacities (see the example of Fig. 1 reporting the streaming of a dynamic 3D content in a mixed or augmented reality application). Moreover, the original PC can present different levels of sparsity depending on the characteristics of the acquiring device or algorithm (as well as on the peculiarities of the object itself). Such requirements can be satisfied by a scalable coding solution that compresses the acquired 3D points into a layered stream enabling at the end terminal an increasingly-refinable reconstruction of the transmitted models.

This paper proposes a transform-based scalable compression solution for dynamic PCs. The input point cloud is quantized into a voxel grid, where octants of $2 \times 2 \times 2$ voxels are processed hierarchically. The main innovation proposed by the current approach relies on the coding strategy for the geometry component. Octets of voxels are reversibly transformed by a context-based binary 3D Cellular Automata (CA) that was tailored on an adaptive binary arithmetic coder and adapts itself to the processed content. The color attributes are coded using the a simplified version of the well-known region-adaptive transform in [9]. The geometry coding strategy proves to be quite efficient when compared with state-of-the-art solutions like TMC1 or PCL. Indeed, for a given bit rate, the reconstructed point clouds present a higher quality at the expense of a limited increase in the computational load.

In the following, Section II overviews the main recently published works on voxel coding, while Section III shows how voxel volumes can be modelled using CA. Section IV describes the proposed coding solution, whose performance is measured by the experimental results reported in Section VII. Section VIII draws the final conclusions.

II. Related works

Nowadays, a dynamic 3D model can be represented by a wide variety of different formats, such as multiview and depth videos, lighfield datas, dynamic meshgrids or point clouds. Among these, PCs have proved to be effective in terms of robustness to noise and effortless processability [10]. This type of 3D model can be generated by different 3D sensors and algorithms, such as Time-of-Flight (ToF) or structured-light depth sensors, laser scanners, LIDARs, etc. Unfortunately, such representation proves to be highly inefficient in terms of storage space since a single point cloud model requires large amount of bits to be stored after its acquisition. To deal with this, several compression strategies have been analyzed during the last years.

A few initial solutions adopted a 2D wavelet transform based scheme [11] or a multi-resolution decomposition of the 3D points [12]. Nevertheless, the need of adapting the coded bit stream to heterogeneous devices and transmission capacities have focused the research efforts towards scalable solutions [13]. One of the first to be presented was based on voxelizing the volume occupied by the model and applying a hierarchical octree decomposition [14]. Such approaches divide the object volume into a voxel grid, where the state of each element or voxel $g(x, y, z)$ (located at coordinates (x, y, z)) depends on whether the voxel contains points ($g(x, y, z) = 1$) or not ($g(x, y, z) = 0$). This voxelized geometry can then be compressed by octree-based strategies [14], [6] into several quality layers which allow reconstructing the original 3D model at different Level-Of-Details (LODs). In fact, the initial voxel volume $g(x, y, z)$ is divided into 8 equally-sized subvolumes. A flag bit signals in the bit stream 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. This hierarchical division permits creating a layered bitstream that enables several partial reconstructions at different resolutions of the original volume (whenever the decoding process stops at one of the upper levels of the coding tree). Note also that different parts of the 3D model can be reconstructed at different LODs allowing a computationally-effective visualization for large point cloud models [15].

This partitioning is very simple to obtain, requires a limited computational effort [16], and can be employed to code different types of data and attributes associated to locations (x, y, z) such that $g(x, y, z) = 1$. Examples of side attributes can be the color information[1] $i_c(x, y, z)$ [14], [17], normals[2] $n_d(x, y, z)$ or roll, pitch, yaw data (available in point clouds acquired by a LIDAR sensor), to mention some of them. The knowledge of the geometry information $g(x, y, z)$ permits reducing the amount of bits to be coded for each additional attribute components [17].

Later approaches have tried to improve the efficiency of octree decomposition by adopting a dynamic depth adaptation whenever no further subdivisions are required [18]. Moreover, computing the statistics of voxel states permit reordering scanning path of voxels in order to minimize the size of the coded bit rate. The solution reported in [19] scans voxel values according to different orders in order to maximize the probability of long sequences of equal occupancy flags. Probability can also be used to optimize the entropy coding algorithm as [20] suggests. Other works combine the octree solution with graph-based transforms [17], [21] in order to decorrelate the signal.

In [9], a region-adaptive transform is introduced in order to effectively compress color information. Transform based approaches have also been applied to non-discrete point cloud by adopting volumetric transforms [22].

Whenever multiple PC are acquired along time, it is possible to exploit the redundancy between temporally-adjacent acquisitions. Naming $g_t(x, y, z)$ the 3D point cloud acquired at time t and $i_{c,t}(x, y, z)$ the corresponding color attributes, their values prove to be highly-correlated with those from previous acquisitions, and therefore, the efficiency of the coding process could significantly improve. Unfortunately, using such information to reduce the amount of coded bits has proved to be a tricky task. Block-based temporal prediction is quite ineffective when applied to dynamic voxelized models. The motion of different object parts can be quite complex (and therefore, hardly-modelled by block displacements), and the number of occupied voxel can significantly change at different instants [23]. For these reasons, most of the proposed approaches perform a motion compensation in the 3D point cloud domain [10], [24] before the voxelization. The corresponding 3D points at different instants are mapped via an iterative 3D registration, e.g., using Iterative Closest Point (ICP) algorithm [25]. Then, the difference is voxelized and coded. Other solutions perform a sort of soft voxel prediction by compressing the current voxelized volume via an arithmetic coding where contexts have been computed on the previous voxel volume [23].

[1]The variables $i_c(x, y, x)$, with $c = R, G, B$ or $c = Y, U, V$, denote the three RGB or YUV color components associated to (x, y, z), respectively.

[2]In this case, the variable $d = x, y, z$ is associated to the axis.

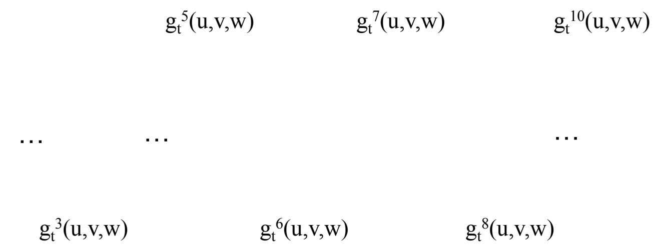

Fig. 2. Progression of LODs for the sequence `longdress`.

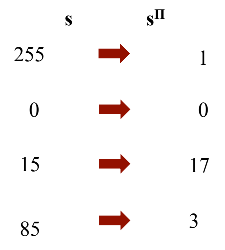

Fig. 3. Example of CA block transform.

According to these premises, more efficient coding strategies need to be investigated; preliminary results show that some effective solutions can be found among transform-based coding. Transform-based compressors [17], [21], [9] have been targeting color information since traditional transforms work well on integer or real data. When dealing with binary values, the possibilities of energy concentration are largely reduced and compression becomes harder. The work in [26] introduces a non-linear reversible transform operating on volumes of binary data, which are modelled as lattices of Cellular Automata. The proposed solution extends the previous strategies designed for binary images [27] and permits obtaining higher compression gains with respect to existing solutions. Such approach has been extended to the temporal dimension in the following paper [28]. The current paper improves the performance of these solutions by designing spatially and temporally adaptive transforms tailored on the point cloud data to be compressed.

III. Transform-based coding of binary voxels using a Cellular Automata modelling

The proposed solution implements a transform-based coding schemes using two non-linear transformations of the input voxels and attributes. The adopted scheme inherits the hierarchical architecture of octree coding, i.e., decomposing $g_t(u,v,n)$ and $i_{c,t}(u,v,w)$ into multiple volumes at different spatial resolutions.

Experimental results showed that it is possible to achieve significant compression gains by tailoring the adopted transform to the statistics of the data and the final arithmetic coder. This can be achieved by designing a context-adaptive transform that changes depending on the configuration of the neighbouring blocks. The design strategy is one of the major novelties presented by this work and the analysis reported in Appendix A proves its optimality with respect to the entropy measurements of bit statistics.

Naming $g_t^r(u,v,w)$ the voxel volume associated to resolution r (or LOD r), it is possible to use the values of $g_t^{r-1}(u,v,w)$ to code $g_t^r(u,v,w)$ (and analogously, $i_{c,t}^{r-1}$ can be used to compress $i_{c,t}^r(u,v,w)$). The modelling that parameterizes such operations will be described in the following paragraphs.

A. Modelling 3D voxel volumes as Cellular Automata lattices

Assuming that the input point cloud is uniformly-quantized into regular cells, the voxel volume $g_t^r(u,v,w)$ at instant t $(u,v,w = 0,\dots,2^r-1)$ can be associated to a block cellular automata (CA) structure. Each voxel/automaton is an atomic cell whose state can assume two possible values (0 or 1). The lattice of CA is partitioned into a set of non-overlapping blocks of size $2\times2\times2$ (Necker neighbourhood); the state of each cell evolves depending on the values of neighbouring voxels (block cellular automata). To parameterize this behaviour, we denote to the configuration of a CA block located at coordinates (u,v,w)

$$\mathbf{s}(u,v,w) = [s_0,\dots,s_7] = [g_t^r(m,n,d)]_{(m,n,d)\in\mathcal{N}}$$

(where s_i denotes the state of the i-th automaton in the neighbourhood $\mathcal{N}$) using an 8-bit integer. Therefore, the configurations $\mathbf{s}(u,v,w)$ span in the range $[0,255]$. The evolution of cell states can be defined by a reversible transform $\mathbf{s}^{\mathbf{\Pi}}(u,v,w) = \mathbf{\Pi}(\mathbf{s}(u,v,w))$, which maps $\mathbf{s}(u,v,w)$ into another state $\mathbf{s}^{\mathbf{\Pi}}(u,v,w)$ (see Fig. 3) and operates independently on each neighbourhood. Since $\mathbf{\Pi}(\cdot)$ has to be invertible (see [29]), the operated transform can be associated to a permutation of the strings in $\{0,1\}^8$, which aims at reorganizing the order of bits in order to enable a more efficient compression. This target can be pursued by concentrating the amount of energy associated to the current block, i.e., mimicking the energy-concentration behaviour of a Discrete Cosine Transform (DCT) or a Discrete Wavelet Transform (DWT).

Similarly to other block CA (like Critters [30]), after an initial transform step, the transformed strings are grouped together into new voxel cubes with halved dimensions. After this reordering, the transform can be operated again on this new smaller volume. Assuming that the values $g_t^{r-1}(u,v,w)$ were decoded before $g_t^r(u,v,w)$, they can be used to enhance the compression gain for the following LODs. This progression of LODs is reported in Figure 2.

B. Transform properties

Since the signal $g(u,v,w)$ is binary, we need to adopt a suitable energy function for each CA block. The binary state of each automata can be associated to an Ising model (spin *up* or *down*), and therefore, the energy of the CA neighbouring can be generically modelled by Potts energy equation

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

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 [31].

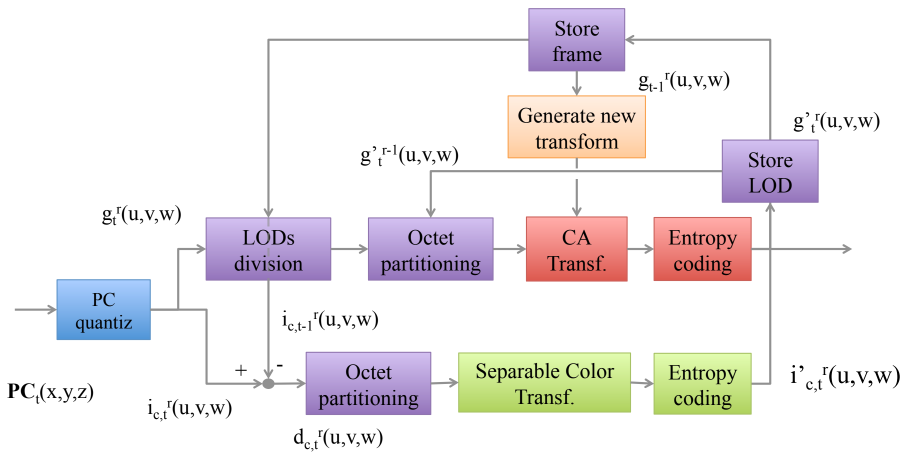

Fig. 4. Block diagrams for the proposed transform coder

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.

Since strings $\mathbf{s}^{\mathbf{\Pi}}(u,v,w)$ are entropy-coded progressively, high compression ratios can be obtained by minimizing the frequency of change in the bit values. Assuming that $n_1(\mathbf{s})$ and $n_0(\mathbf{s})$ respectively refer to the number of states equal to 1 and 0 for $\mathbf{s}$ (coordinates u, v, w have been omitted for the sake of clarity), let us consider the symbol 1 the most probable bit or MSB. An efficient transform would map the most probable values $\mathbf{s}$ into strings $\mathbf{s}^{\mathbf{\Pi}}$ where the number of MSB (i.e., $n_1(\mathbf{s}^{\mathbf{\Pi}})$) is maximized. This operation leads to long sequences of constant binary values which can be effectively compressed by a binary arithmetic coder. Being 0s and 1s equivalent in the Ising modelling, such operation can be associated to the maximization of the number of null coefficients operated by DCT and DWT on natural images.

Considering that the null string is signaled by the previous LOD (i.e., $\mathbf{s}(u,v,w) = \mathbf{0}$ if the corresponding voxel $g^{r-1}(u',v',w')$ is 0), it is possible to design the permutation $\mathbf{\Pi}(\cdot)$ so that some properties are verified:

a) $\mathbf{\Pi}(0) = 0$;
b) the highly-probable blocks $\mathbf{s}$ should be converted to $\mathbf{s}^{\mathbf{\Pi}}$ s.t. $n_1(\mathbf{s}^{\mathbf{\Pi}})$ is maximized;
c) the least probable blocks $\mathbf{s}$ should be converted to $\mathbf{s}^{\mathbf{\Pi}}$ s.t. $n_1(\mathbf{s}^{\mathbf{\Pi}})$ is minimized.

Note that requirements (a) implies that an empty block remains empty since it does not have to be coded, while the other two requirements aim at maximizing the number of 1s whenever the corresponding block is non null. Such requirements differ from those adopted in [26], where the adopted transform was designed to keep the Potts energy of $\mathbf{s}$ and $\mathbf{s}^{\mathbf{\Pi}}$ unaltered in order to allow multiple intermediate reconstruction levels between one LOD and the following one.

Given these requirements and bit-plane based arithmetic coder (which will be described in Section V), the probability mass function $P[\mathbf{s}]$ for the symbols $\mathbf{s}$ (which can be obtained for a specific point cloud to be coded or on a set of training models) completely defines the transform $\mathbf{\Pi}(\cdot)$. The detailed characteristics of such procedure are reported in the Appendix A.

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

IV. A General Overview of the Proposed Coder

The modelling framework described in the previous section can be used to implement a transform-based dynamic point cloud coder that employs adaptive non-linear transforms for both geometry and color attribute coding. Figure III-A reports a block diagram of the proposed scheme, whose building blocks will be described in the following sections. The input point cloud can be seen as a set of attributes (color components, normals, etc.) at different three-dimensional locations (x,y,z), i.e,, a multidimensional function such that

$$\mathbf{PC}_t(x,y,z) : \mathbb{R}^3 \mapsto \mathcal{A} \cup \texttt{empty}$$

where $\mathcal{A}$ is $[0,255]^3$ in case it refers to color components or $[0,1]^3$ in case it refers to normals. The symbol `empty` denotes that no points are available at those coordinates.

At first, coordinates (x,y,z) are uniformly-quantized into a three-dimensional voxel grid such that

$$u = Q_x(x) \quad v = Q_y(y) \quad w = Q_z(z) \tag{2}$$

where $Q_x(\cdot)$, $Q_y(\cdot)$, $Q_z(\cdot)$ denote the quantization functions associated to the three different axis. In our implementation, we considered uniform quantizers such that

$$Q_x(\cdot) = \left\lceil \frac{x}{\Delta} \right\rfloor = Q_y(\cdot) = Q_z(\cdot)$$

where $\lceil \cdot \rfloor$ denotes a rounding operation. Assuming that r bits are assigned to each quantizer, it is possible to generate the voxel volume $g_t(u,v,w)$ sized $N \times N \times N$, where $N = 2^r$.

Voxel $g_t^r(u,v,w)$ is set to 1 if it exists at least one triplet (x,y,z) that is quantized into (u,v,w) and whose corresponding value $\mathbf{PC}_t(x,y,z) \neq \texttt{empty}$; otherwise, $g_t^r(u,v,w)$ is set to 0.

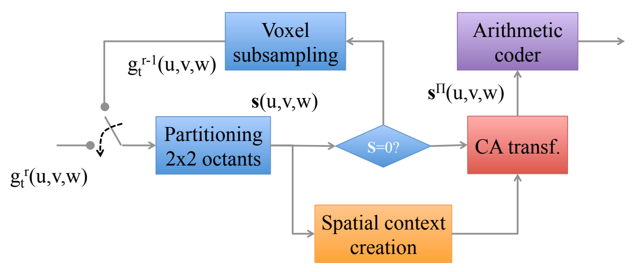

Fig. 5. Block diagrams for transform coding of geometry.

For non-empty points, the component values $\mathbf{PC}_t(x,y,z) \in \mathcal{A}$ are assigned to $i_c^r(u,v,w)$; in case multiple points (x,y,z) are mapped to the same location (u,v,w), values $\mathbf{PC}_t(x,y,z)$ are averaged.

The point cloud $\hat{\mathbf{PC}}_t$ can be reconstructed at the decoding stage by dequantizing (u,v,w) such that

$$\hat{x} = u\,\Delta \quad \hat{y} = v\,\Delta \quad \hat{z} = w\,\Delta \tag{3}$$

and assigning

$$\hat{\mathbf{PC}}_t(\hat{x},\hat{y},\hat{z}) \leftarrow \hat{\imath}_c^r(u,v,w) \tag{4}$$

where $\hat{\imath}_c^r(u,v,w)$ is the reconstruction of $i_c^r(u,v,w)$.

Signals $g^r(u,v,w)$ and $i_c^r(u,v,w)$ are compressed by the geometry coding and the color coding modules. These schemes perform a motion compensation (MC) at first, which exploits the previously-coded data to enhance the compression performance on the current instance or frame. Then, two non-linear transform coding solutions are entailed.

V. Geometry Coding

In the coding phase, the input voxel volume $g_t^r(u,v,w)$ is partitioned into $2\times 2\times 2$ octants whose values can be characterized by integers $\mathbf{s}(u,v,w)$. These are converted into the octets

$$\mathbf{s}^{\mathbf{\Pi}}(u,v,w) = \left[s_i^{\mathbf{\Pi}}\right] = \mathbf{\Pi}\left(\mathbf{s}(u,v,w)\right),$$

whose bits can be compressed by a set of entropy coders. In fact, binary values $s_i^{\mathbf{\Pi}}$ $(i=0,\dots,7)$ are stacked into 8 separate binary streams q_i. Each stream q_i is then compressed by a context-adaptive binary arithmetic coder, whose contexts models the probability of the MSB.

Note that $\mathbf{s} = \mathbf{s}^{\mathbf{\Pi}} = \mathbf{0}$ does not need to be coded since the decoding of the previous LOD $g_t^{r-1}(u,v,w)$ permits inferring whether the current octant is empty or not. As a matter of fact, streams q_i contain information regarding non-empty octants only, thus reducing the amount of coded information. The adopted coding strategy is visualized in Fig. 5.

The efficiency of the adopted scheme is significantly affected by the designed transform, which must fit accurately the statistics $P[\mathbf{s}]$ of the data to be compressed. As a matter of fact, the use of an adaptive transform proves to be crucial in the bit rate minimization. This adaptability can be achieved by introducing a set of contexts that condition the computation of the octets statistics $P[\mathbf{s}]$.

A. Context structure

The design of the CA transform relies on the fact that the probability distribution $P[\mathbf{s}]$ for the current block $\mathbf{s}(u,v,w)$ can be conditioned by the value of the neighbouring voxels. Previous works have shown how the probability of the occupancy value of a given voxel (or octant) can be conditioned by the occupancy values of the neighboring voxels (octants) [19]. In our implementation, upper and left voxels are considered as they have already been coded and reconstructed when processing $g_t^r(u,v,w)$. More precisely, we considered the context

$$\texttt{ctx}(u,v,w) = \begin{bmatrix} g_t^r(u,v,w-1) \\ g_t^r(u,v+1,w-1) \\ g_t^r(u+1,v,w-1) \\ g_t^r(u+1,v+1,w-1) \\ g_t^r(u-1,v,w+1) \\ g_t^r(u-1,v+1,w+1) \\ g_t^r(u,v-1,w+1) \\ g_t^r(u+1,v-1,w+1) \end{bmatrix}. \tag{5}$$

The selected voxel values were identified from a set of experimental tests on different voxel models to maximize the coding gain. Given a set of training data or a training model, the probabilities $P\left[\mathbf{s}|\texttt{ctx}\right]$ were computed for each context value (coordinates u,v,w were omitted for the sake of clarity). The values $P\left[\mathbf{s}|\texttt{ctx}\right]$ are used to generate the optimal $\mathbf{\Pi}_{\texttt{ctx}}$ using the procedure reported in the Appendix A.

Although transform design requires some additional computational effort with respect to standard coding procedure, the temporal correlation among adjacent frames can be exploited to avoid recomputing the transform at every frame.

B. Extension to time dimension

As it was anticipated in the Introduction, the prediction of geometry voxels is quite difficult since an accurate matching is not possible. Nevertheless, temporal correlation was exploited in the computation of the best transforms $\mathbf{\Pi}_{\texttt{ctx}}(\cdot)$. In fact, the compression performance is maximized whenever the adopted transforms are tailored to the input data. This implies estimating $\mathbf{\Pi}_{\texttt{ctx}}(\cdot)$ on the current voxel volume and code $\mathbf{\Pi}_{\texttt{ctx}}^{-1}(\cdot)$ in the bitstream in order to allow the decoder to inverse the coding process and reconstruct the volume. In order to reduce the amount of coded bits, no information about $\mathbf{\Pi}_{\texttt{ctx}}(\cdot)$ is included in the bitstream; instead, after coding each frame, the voxel values probability distribution is computed for each context value. The computed statistics are used to generate a new set of context-related transforms $P_{\texttt{ctx}}(\cdot)$ which are going to be used for the following frame.

VI. Color Coding

After coding voxels $g_t^r(u,v,w)$, the proposed coding scheme can re-use this information to drive the compression of color data. The proposed color coding strategy is very close to the Region Adaptive Hierarchical Transform (RAHT) coding strategy [9], where an adaptive separable transform on blocks of 2 voxels is progressively applied along each axis. **The difference with respect to the solution in [9] relies on**

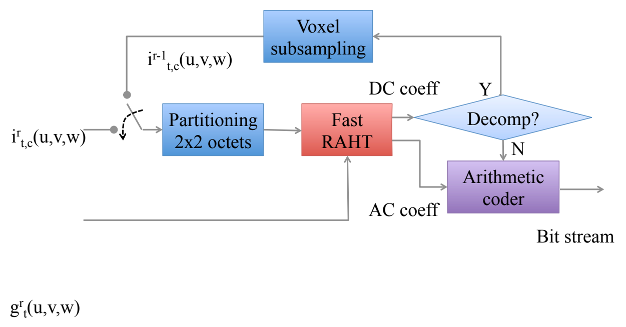

Fig. 6. Block diagrams for transform coding of color.

the transform coefficients, which in our implementation have been recomputed so that all the multiplications can be implemented with multiplication-free operations. After transforming the couple of voxels, the resulting AC coefficients are then compressed by an adaptive arithmetic coder, while DC coefficients are employed to generate the color information for $i^r_{c,t}(u,v,w)$. These passages are described in the following subsections

A. Transform coding using RAHT

Assuming that $u = 2m$, $v = 2n$, $w = 2o$ at LOD r, the adopted transform can be described by the equation

$$\begin{bmatrix} I^r_{c,t}(u,v,w) \\ I^r_{c,t}(u+1,v,w) \end{bmatrix} = \frac{1}{K} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} i^r_{c,t}(u,v,w) \\ i^r_{c,t}(u+1,v,w) \end{bmatrix} \quad (6)$$

where we assume that it is operated along the x axis and $g^r_t(u,v,w) = g^r_t(u+1,v,w) = 1$. **Note that all the multiplications were removed from the transform matrix, which can be implemented with simple sums. Moreover, the constant K, which is a rescaling factor and is represented with a float value in the RAHT coder, is now approximated by an even integer number in order to implement the whole transformation with integer register shifts and sums only. This permits reducing the computational complexity of the whole approach.**

The coefficient $I^r_{c,t}(u,v,w)$ is referred as DC coefficient, while $I^r_{c,t}(u+1,v,w)$ is the AC coefficient. If only $g^r_t(u,v,w) = 1$, then the DC coefficient $I^r_{c,t}(u,v,w)$ is $i^r_{c,t}(u,v,w)$; on the contrary, if only $g^r_t(u+1,v,w) = 1$ then $I^r_{c,t}(u,v,w) = i^r_{c,t}(u+1,v,w)$. In these latter cases, no AC coefficient is generated. Then, $I^r_{c,t}(u,v,w)$ are processed by the same transform along the y axis and z axis, separating the resulting DC and AC coefficients at each application. The final resulting $I^r_{c,t}(u,v,w)$ is then sent to LOD r, i.e., $i^{r-1}_c(m,n,o) = I^r_c(u,v,w)$. The transform is then iterated on $i^{r-1}_{c,t}(m,n,o)$.

The resulting DC and AC coefficients are then quantized with quantization step Δ_c and coded using an 8-bits arithmetic coder.

Note that the proposed coding strategy is coupled to the geometric coding strategy s.t. it is possible to decode a given LOD for both geometry and color.

Static	Dynamic	Format of coord. values	Attributes
longdress	longdress	uint10	RGB
soldier	soldier	uint10	RGB
Ford	Ford	float32	normals
loot		uint10	RGB
redandblack*		uint10	RGB
queen		uint10	RGB
ArcoValentino		float32	RGB
PalazzoCarignano		float32	RGB
HouseWithoutRoof*		float32	RGB
Facade15		float32	RGB

TABLE I
STATIC AND DYNAMIC MODELS USED IN EXPERIMENTS.

Model	TMC1	Proposed
longdress	1.67	1.13 (−32.34 %)
queen	1.51	1.37 (−11.26 %)
soldier	2.17	1.43 (−34.10 %)
loot	1.53	1.02 (−33.33 %)
Average	1.78	1.24 (−27.73 %)

TABLE II
BIT RATES (IN BPP) FOR TMC1 AND THE PROPOSED GEOMETRY CODER OBTAINED FROM LOSSLESS COMPRESSION OF DIFFERENT STATIC MODELS

Model	TMC1	Proposed
longdress	44.95	29.65 (−34.03 %)
soldier	63.32	41.39 (−34.62 %)
Average	54.13	35.52 (−34.32 %)

TABLE III
BIT RATES (IN MBIT/S) FOR PCL, TMC1 AND THE PROPOSED GEOMETRY CODER OBTAINED FROM LOSSLESS COMPRESSION OF DIFFERENT DYNAMIC SEQUENCES.

B. Temporal prediction

Differently from the case of geometry, color prediction can lead to some bit rate reduction. As anticipated in the Introduction, point cloud prediction can be operated on $\mathbf{PC}_t(x,y,z)$ via ICP alignment of points. This operation proves to be effective for small motion and requires a significant amount of resources. A simpler solution consists in operating on the voxel domain by predicting the current attributes $i^r_{c,t}(u,v,w)$ from the previous ones $i^r_{c,t-1}(u,v,w)$. More precisely, the color coding engine computes the difference

$$d^r_{c,t}(u,v,w) = i^r_{c,t}(u,v,w) - i^r_{c,t-1}(u,v,w) \quad (7)$$

which is then processed by the RAHT transform and the resulting coefficients are coded in a binary stream as described before. When $g^r_t(u,v,w) = 1$ and $g^r_{t-1}(u,v,w) = 0$, there is not a reference color for $i^r_{c,t}(u,v,w)$ in the previous voxel frame; the reference is then generated averaging the values of the non-empty neighbouring voxels within the $3 \times 3 \times 3$ block centered on (u,v,w).

VII. PERFORMANCE EVALUATION

In order to evaluate the performance of the proposed strategy, experimental tests were performed on an extensive set of point clouds generated using different algorithms and devices (see some examples in Fig. 7). In these tests, we adopted the configurations and the test sequences specified in [32] in order to be compliant with the Call-for-Proposal [7]. The obtained results were compared with the solution in [6], which

(a) (b) (c) (d) (e)

Fig. 7. Example of models used in the tests. Models were rotate to allow the reader to better understand their structures. (a) `longdress`; (b) `soldier`; (c) `ArcoValentino`; (d) `Ford`; (e) `PalazzoCarignano`.

is the anchor codec specified in [7] and will be labelled as `PCL`, and with the reference codec `TMC1` (labelled with same name). The reference code for `PCL` can be obtained from [33]. **The adopted TMC1 implementation is the reference one provided by MPEG.** Such choice was motivated by the fact that these count among recent state-of-the-art strategies for fast point cloud compression (such solutions prove to have a similar computational load on the hardware platform).

Performances were evaluated on both static and dynamic models, which are reported in Table I. Models denoted with $*$ are used to compute voxel statistics $P[\mathbf{s}|\texttt{ctx}]$ to design the adopted transform. The compression performance was measured using the quality metrics reported in [7] and comparing the input point cloud $\mathbf{PC}_t(x, y, z)$ with its reconstructed version $\hat{\mathbf{PC}}_t(x, y, z)$. More precisely, the color reconstruction accuracy was measured using the standard PSNR metric. The similarity between the reconstructed geometry and the original values are measured by the PSNR D1 metric (which parameterizes the point-to-point accuracy) and the RMS D2 metric (which parameterizes the point-to-plane accuracy). This second measure is based on the projection of the point coordinates with respect to normals to the surface (see [34] for more details). PSNR is usually reported in dB, while RMS is adimensional. Bit rates for static models are reported in terms of bits-per-point (bpp), i.e., the total bit rate divided by the number of 3D points in the original model. As for dynamic sequences, bit rates are reported in Mbit/s.

Geometry information can be coded in lossless and lossy mode with different number of bits. For each geometry coding set-up, color information was compressed at different quality changing the quantization step, made exception for the `Ford` model and sequence which was acquired using a Lidar sensor. In this case, information about normals is compressed in place of color components: values are converted into 8-bit integers and coded using the same tools for R,G,B values.

A. *Lossless coding of geometry*

At first, we evaluated the coding performance of geometry data in lossless mode. To this purpose, we considered the static models `longdress`, `loot`, `soldier`, and `queen`, whose coordinates (x, y, z) are coded using 10 bits. The adaptive transforms $\mathbf{\Pi}_{\texttt{ctx}}(\cdot)$ were computed on the statistics of `redandblack` model. **Table II reports the coded bit rates for the proposed solution and the `TMC1` scheme. It is possible to notice that the proposed strategy permits reducing the coded bit rate of about 28 % with respect to `TMC1`. This coding gain proves to be extremely useful in reducing the overall bit rate of a complete point cloud which includes color information as well. Note also that the compression performance is improved with respect to our previous coding solutions in [26], [28], where the compression gain was about 20 % with respect to `TMC1`.** This fact was possible thanks to a better energy concentration capacity of the adopted transform (it has been tailored to the characteristics of the signals), as well as to its optimization with respect to the structure of the adopted arithmetic coder. **Table III reports the coded bit rates (in Mbits/s) generated by the proposed and the `TMC1` coders for `longdress` and `soldier` sequences. The coders used GOP of 2 frames. It is possible to notice that the same coding gain was obtained on dynamic sequences as well.**

	8 bits		**9 bits**		**10 bits**	
Model	$\mathbf{\Delta P}$	$\mathbf{\Delta R}$	$\mathbf{\Delta P}$	$\mathbf{\Delta R}$	$\mathbf{\Delta P}$	$\mathbf{\Delta R}$
longdress	1.37	−34.89	3.87	−39.03	6.33	−20.10
queen	3.49	−49.55	6.51	−41.41	4.34	−18.84
soldier	1.66	−51.71	3.09	−39.80	4.68	−24.24
loot	3.06	−56.68	4.65	−43.92	4.87	−25.36
ArcoValentino	0.13	−30.47	0.13	−22.62	0.25	−11.90
PalazzoCarignano	1.09	−54.79	1.07	−47.97	1.19	−36.04
Facade15	1.63	−62.49	1.10	−61.84	0.91	−57.61
Ford100	3.18	−4.75	3.66	−1.93	3.90	1.92
Average	1.95	−43.17	3.01	−37.31	3.31	−24.02

TABLE IV
BJONTEGAARD ΔPSNR AND ΔRATE ON DIFFERENT STATIC MODELS FOR LOSSY CODING OF COLOR AND GEOMETRY.

B. *Lossy coding of geometry and color*

When geometry information is coded in lossy mode, it is possible to change the amount n_b of bits assigned to the quantizers $Q_x(\cdot)$, $Q_y(\cdot)$, $Q_z(\cdot)$, similarly to the configuration of the `PCL` coder. This implies that, despite using the same quantization step Δ_c, the reconstruction quality of the associated color information varies, and it is possible to draw a PSNR-vs.-rate curve corresponding to a n_b value for both the proposed and the `PCL` coders. Such curves can be compared in compact form by computing the Bjontegaard ΔPSNR (here referenced as ΔP) and ΔRate (here referenced

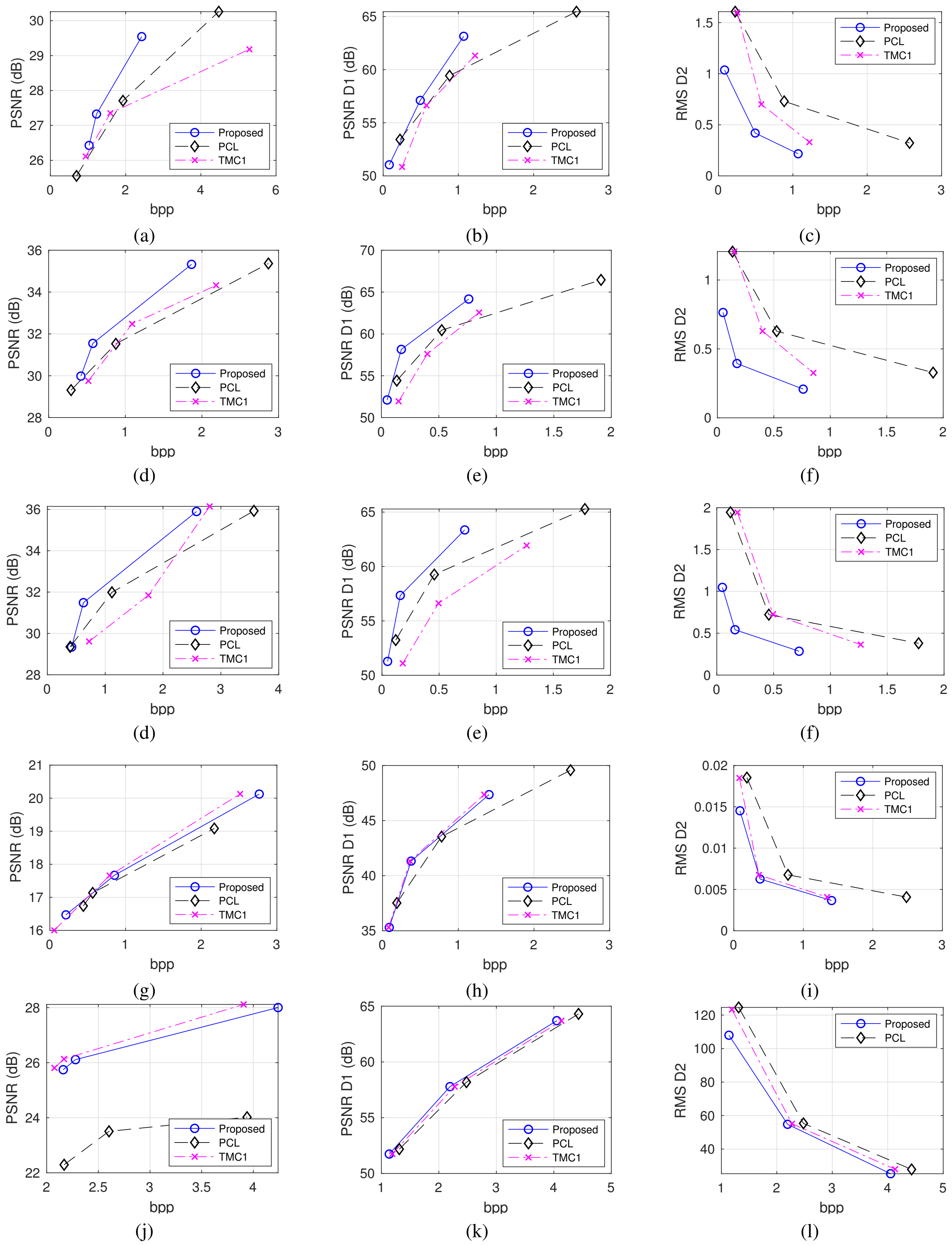

Fig. 8. Coding performance for different static models. Displayed metrics are PSNR on reconstructed color components (first column), PSNR D1 on geometry (second column), RMS D2 on geometry (third column). Static models are `longdress` (first row), `queen` (second row), `soldier` (third row), `Palazzo_Carignano` (fourth row), and `Ford_01_vox1mm-0100` (fifth row),

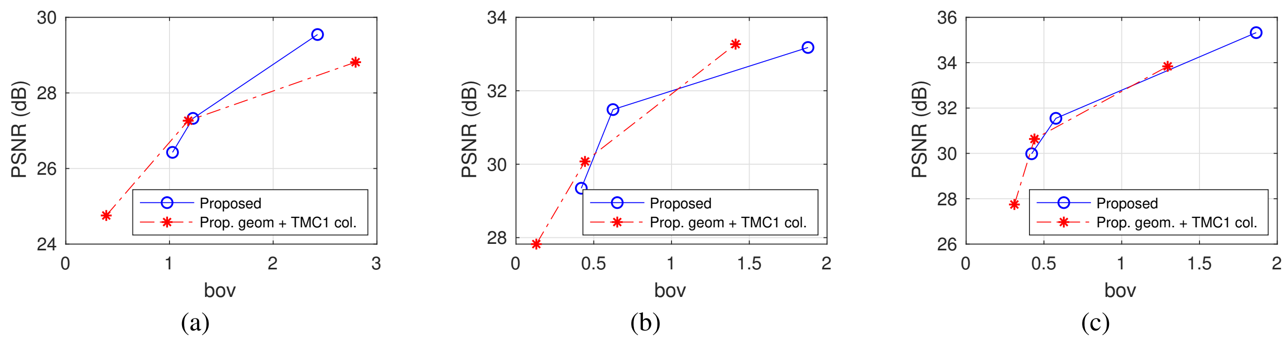

Fig. 9. Rate-distortion curves of color compression for `TMC1` and the proposed solution. Geometry was coded using the CA-based strategy. Graphs display the results for (a) `longdress`, (b) `soldier`, and (c) `queen` models.

as ΔR) parameters as suggested in [35]. **Table IV reports the ΔR and ΔP values (whose increment are measured in % and dB, respectively) for different static models. The reference coder is PCL.** It is possible to notice that the quality improvement increases as the number $n_b$of bits increases. Conversely, the average bit rate reduction decreases from 43 % for $n_b = 8$ to 24 % for $n_b = 10$. This is due to the fact that the designed transform is extremely effective for dense and convex models $g(u, v, w)$. Such condition makes the probability distribution $P(\mathbf{s})$ highly biased, and therefore, it is possible to obtain streams q_i with long trails of 1s. This is mostly verified whenever the number of bits per component is low. It is also possible to notice that performance gain is lower for sparse models or models with a lot of noise (see Fig. 7 c-e). As an example, it is possible to check the results obtained for `Arco_Valentino`, `Palazzo_Carignano`, `Facade` and `Ford` models. In these cases, the probability distribution $P(\mathbf{s})$ is less biased, and as a result, entropy coding is less effective on the final bit stream. Considering the compression of dense models of people, the least improvement has been obtained on `longdress` and `soldier`: this is mainly due to the complexity of texture information which is highly non-stationary, and therefore, it can not be effectively compacted by the RAHT transform. **As an experimental evidence for this, we report the rate-distortion curves of color compression on static models `longdress`, `soldier`, and `queen` for the `TMC1` and the proposed solution (see Fig. 9) with geometry information coded using the proposed strategy. In this case, the reconstructed point coordinates are the same, but color attributes are coded differently. It is possible to notice that the coding performance of the proposed scheme slightly improves at low bit rates.**

For a given quality level of the geometry component, it is possible to select the best coding configuration Δ_c for color compression. In our tests, we chose Δ_c such that the obtained (PSNR,rate) point is the closest to the upper left corner of the PSNR-vs.-rate plot. This generates a rate-distortion point for each n_b value; such conditions are reported in Figure 8, where the color PSNR, PSNR D1 and RMS D2 values are displayed as a function of the bit rate. The displayed graphs confirm that the coding gain is lower for sparse or noisy models (i.e., `Palazzo_Carignano` and `Ford`). The difference between PSNRD1-vs.-rate curves (second column) is lower in these cases due to the highly-complex distribution of 3D points. This fact is confirmed by the RMS D2 metric (third column) and affects the compression performance on color components as well; as an evidence for this, it is possible to compare Fig. 8(g) with Fig. 8 (a) and (d).

Fig. 10. Detail for the models `longdress` reconstructed at 2 bpp (first column), soldier ar 1.2 bpp (second column), and queen at 1.4 bpp (third colum). Original (first row); proposed (second row); `PCL` (third row); `TMC1` (fourth row).

Figure 10 reports a detail of different reconstructed PC models using the proposed (second row), `TMC1` (third row) and the `PCL` (fourth row) coders. It is possible to notice that the proposed solution is able to reconstruct a denser and more accurate point clouds with respect to the anchor.

C. *Lossy coding of geometry and color for dynamic sequences*

Final tests concerned the compression of dynamic sequences, where the correlation interlying between temporally-

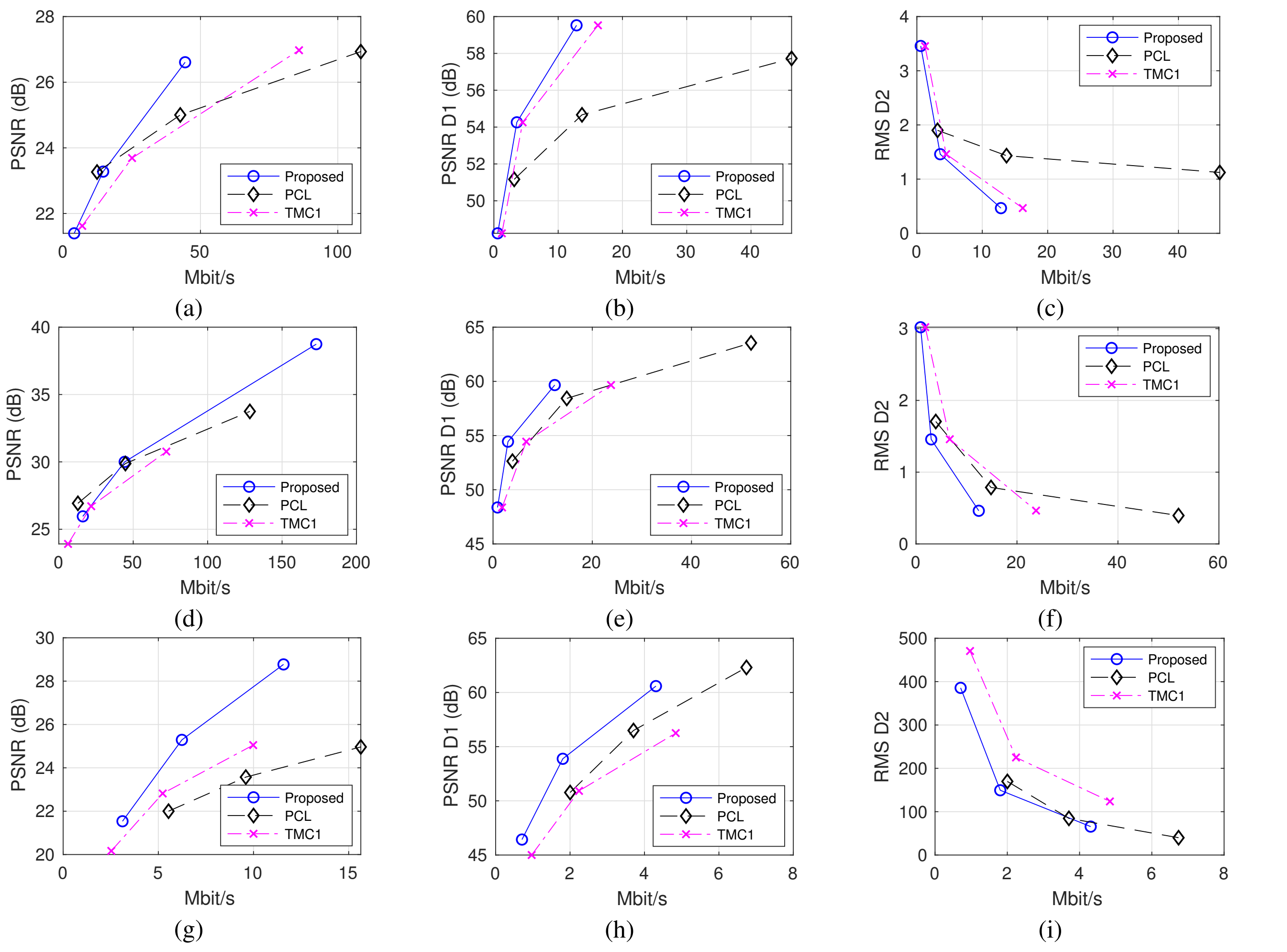

Fig. 11. Coding performance for different dynamic models. Displayed metrics are PSNR on reconstructed color components (first column), PSNR D1 on geometry (second column), RMS D2 on geometry (third column) as a function of total bit rate. Dynamic sequences are `longdress` (first row), `soldier` (second row), and `Ford_01` (third row).

adjacent point clouds can be exploited to reduce the final bit rates. In these tests, we considered GOP of 2 frames (as in [6]). Figure 11 reports PSNR, PSNR D1 and RMS D2 plots versus the coded bit rate. The quality of geometry coding is defined by choosing the number n_b, while the associated quality for color components is selected as described in the previous paragraph. Experimental results show that at high bit rates the proposed solution performs better for all the considered metrics, while the coding gain decreases at low bit rates. This fact changes for the `Ford` sequence where no significant differences can be seen in the RMS D2 metric, while the Proposed solution is able to reconstruct information about normals more accurately than the `PCL` approach.

In order to provide a further clarification about the obtained results, we report the coded bit rate and the average PSNR and PSNR D1 metrics for different configurations of the proposed and `TMC1` codec. The results obtained for the sequence `longdress` are reported in Table V. The reported results show that the geometry coding engine reduces the coded bit rate of approximately 45 % with respect to `TMC1` for both all Intra coding and IP coding. The total coding gain for both color and geometry components is lower (around 15 %) since the color compression strategy has a lower efficiency with respect to the `TMC1` solution. This fact is due to the adopted transform (see eq. (6)) and the associated bit coding strategy; the total performance could be significantly improved by adopting `TMC1` color coding strategy with the proposed geometry coding solution.

Final tests compares the proposed solution with the `TMC2` codec [36]. It is possible to notice that the rate-distortion performance of `TMC2` codec is better since many pre-processing steps that are applied to the input point cloud optimize it for compression increasing the efficiency of the codec (Fig. 12). This improvement is payed in terms of computational complexity since the coding time for `TMC2` is approximately 110 times bigger than the one required by the proposed solution (see Table VI). It is possible to notice that, whenever the texture presents a regular patterns (like in the `soldier` sequence), the gap between the presented strategy and `TMC2` is much higher since texture video is coded using well-established and effective video coding schemes. In case the texture is more irregular (like in the `longdress` case), the difference reduces. Moreover, it is possible to observe that `TMC2` performs poorly on sparse point clouds: the proposed solution proved to be the best for the `Ford_01`

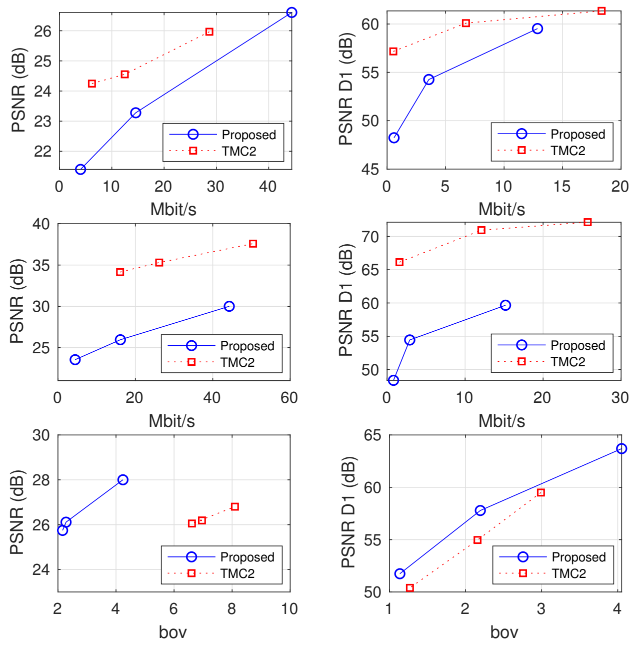

Fig. 12. Rate-distortion curves for the proposed codec and `TMC2`. Graphs display the PSNR values for the luma component (first column) and the PSNR D1 for the geometry (second column). Data are to be referred to `longdress` (first row), `soldier` (second row), `Ford_01` (third row).

sequence (where PSNR is to be referred to the information concerning normals). In fact, `TMC2` texture coding was designed for dense models and the sparsity of a LIDAR acquisition can not be compressed effectively. As a matter of fact, the difference in compression gain is utterly evident for texture compression, while the performance of geometry coding is not dramatically lower.

D. *Computational complexity analysis*

The required computational load was evaluated measuring the encoding time for the lossy compression of dynamic point cloud sequences `longdress` and `soldier`. The encoding time was obtained averaging the time elapsed for compression at different bit rates. Tests were performed on a i7 Quad Core Machine and results are reported in Table VI. It is possible to notice that the require computational load is comparable or even lower than that required by the solution in [6]. It is also worth noting that the encoding time for the proposed solution is nearly constant; this makes the computational effort independent from the input signal and easily predictable.

The low complexity is mainly due to the fact that the adopted transform shows a better energy concentration with respect to other solutions; this imply that the amount of processed occupied voxel is lower. Moreover, all the operations are implemented on binary symbols, and therefore, it is possible to implement them with fast arithmetic coding routines.

The reported data show also the encoding time for the `TMC2` coder. It is possible to notice that the computational effort is much higher because of the many pre-processing operations performed by `TMC2`. This makes the coder not suitable for real-time compression. On the other hand, the proposed solution requires a much lower complexity allowing the creation of a scalable bitstream.

VIII. Conclusions

The paper presented a transform-based coding approach for voxelized dynamic point clouds using a hierarchical Cellular Automata block transform for geometry information and a region-adaptive transform for color information. Temporal prediction allows a further reduction of the final bit stream. Future research works will be devoted to improve the temporal prediction strategy and introduce intermediate reconstruction levels between adjacent LODs.

Appendix

A compression-efficient permutation $\mathbf{\Pi}$ must be tailored with respect to a specific probability mass function $P[\mathbf{s}]$. Let us assume that a probability mass function has been computed on one or more training voxel volumes. In the transform design, we need to identify the function $\mathbf{\Pi}(\cdot)$ such that the sum of the bit streams generated by the arithmetic coders from q_i, $i = 1, \ldots, 8$, is minimal. To this purpose, we can model the streams of binary symbols q_i as independent Bernoulli sources.

Given the probabilities $P[\mathbf{s}]$, it is possible to order the strings $\mathbf{s}_k \in \{0,1\}^8 \setminus \mathbf{0}$ such that $P[\mathbf{s}_k] \geq P[\mathbf{s}_h]$ if $k < h$. Similarly, strings $\mathbf{s}^{\mathbf{\Pi}}$ can be sorted in decreasing order, i.e., $\mathbf{s}^{\mathbf{\Pi}}_k \geq \mathbf{s}^{\mathbf{\Pi}}_h$ if $k < h$. It is straightforward to verify that mapping $\mathbf{s}_k$ into $\mathbf{s}^{\mathbf{\Pi}}_k$ permits satisfying properties (b) and (c) reported in Section III-B. As an example, symbol 255 will be mapped to the most probable $\mathbf{s}$, 254 to the second most probable one, and so on. Such strategy proves to be both simple and effective since it maximizes the probability of having long trails of equal symbols in the stream without requiring an excessive computational power.

It is possible to formally verify that this choice is optimal in terms of entropy. Let us consider the expected entropy $H^k(q_i)$ for the stream q_i at iteration k such that

$$H_k(q_i) = -p^i_{k,0} \log p^i_{k,0} - p^i_{k,1} \log p^i_{k,1} \tag{8}$$

where

$$\begin{aligned} p^i_{k,0} &= \frac{\sigma^i_{k,0}}{\sigma^i_k} = \frac{\sum_{h=0}^{k-1} P(\mathbf{s}_h)\mathcal{I}(s^{\mathbf{\Pi}}_{h,i} == 0)}{\sum_{h=0}^{k-1} P(\mathbf{s}_h)} \\ p^i_{k,1} &= \frac{\sigma^i_{k,1}}{\sigma^i_k} = \frac{\sum_{h=0}^{k-1} P(\mathbf{s}_h)\mathcal{I}(s^{\mathbf{\Pi}}_{h,i} == 1)}{\sum_{h=0}^{k-1} P(\mathbf{s}_h)}. \end{aligned} \tag{9}$$

Assuming that streams q_i are coded separately, the overall entropy can be modelled by $\overline{H}_k = \sum_{i=0}^{7} H_k(q_i)$, whose partial derivative can be written as

$$\frac{\partial \overline{H}}{\partial p^i_{k,1}} = \log \frac{\sigma^i_{k,0}}{\sigma^i_{k,1}} \qquad \frac{\partial \overline{H}}{\partial p^i_{k,0}} = \log \frac{\sigma^i_{k,1}}{\sigma^i_{k,0}} \tag{10}$$

Assigning the 0 symbols (i.e., increasing $\partial p^i_{k,0}$ and decreasing $\partial p^i_{k,1}$) to streams where $\partial \overline{H} / \partial p^i_{k,1}$ is maximum (or conversely, $\partial \overline{H} / \partial p^i_{k,0}$ is minimum) makes possible to minimize the increment of $\overline{H}_k$.

This implies that the transform design routine should map 0s to the same streams as much as possible (the transform must be invertible). As a result, the most probable $\mathbf{s}$ are mapped to $\mathbf{s}^{\mathbf{\Pi}}$ in decreasing order.

Parameters		TMC1			Proposed		
Configuration	N_b, Δ_c	**Bit rate**	**PSNR**	**PSNR D1**	**Bit rate**	**PSNR**	**PSNR D1**
longdress I & P geom.	7, –	1.05	–	48.23	0.57 (−48%)	–	48.24
	8, –	3.71	–	54.26	2.07 (−44%)	–	54.26
	9, –	13.23	–	59.52	7.54 (−43%)	–	59.52
longdress All I geom.	7, –	1.25	–	48.23	0.68 (−45%)	–	48.24
	8, –	4.09	–	54.26	2.28 (−44%)	–	54.26
	9, –	13.79	–	59.52	7.86 (−43%)	–	59.52
longdress I & P color.	7, 20	4.38	21.60	–	4.02 (−8%)	21.40	–
	8, 20	15.09	23.57	–	11.82 (−22%)	23.12	–
	9, 20	49.21	26.82	–	40.58 (−17%)	26.64	–
longdress All I color.	7, 20	5.31	21.57	–	4.87 (−8%)	21.53	–
	8, 20	17.87	23.63	–	14.10 (−19%)	23.52	–
	9, 20	55.73	26.83	–	45.96 (−17%)	26.81	–

TABLE V
BIT RATE (IN MBIT/S) AND AVERAGE PSNR AND PSNR D1 METRICS (IN DB) FOR DIFFERENT CONFIGURATIONS.

Sequence	**Proposed**	**PCL**	**TMC2**
longdress	604 ms	619 ms	98.25 s
soldier	640 ms	827 ms	72.56 s
Ford_01	560 ms	560 ms	31.30 s

TABLE VI
AVERAGE FRAME ENCODING TIME FOR DIFFERENT SEQUENCES

REFERENCES

[1] Miska M. Hannuksela, Ye Yan, Xuehui Huang, and Houqiang Li, "Overview of the multiview high efficiency video coding (MV-HEVC) standard," *Proc. of ICIP 2015*, pp. 2154–2158, 2015.

[2] 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.

[3] S.B. Gokturk, H. Yalcin, and C. Bamji, "A Time-Of-Flight Depth Sensor - System Description, Issues and Solutions," in *Proc. of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW 2004)*, June 27 – July 2, 2004, vol. 3, p. 35.

[4] J. D. Spinhirne, "Micro pulse Lidar," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 31, no. 1, pp. 48–55, Jan 1993.

[5] 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.

[6] R. Mekuria, K. 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*, vol. 27, no. 4, pp. 828–842, April 2017.

[7] MPEG 3DG and Requirements, "Call for proposals for point cloud compression v2 - doc. n16763," in *ISO/IEC JTC1/SC29/WG11 Coding of Moving Pictures and Audio Meeting Proceedings*, Apr. 2017, files: w16763_PCC_CfP.docx.

[8] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. Chou, R. Cohen, M. Krivokuca, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. M. Tourapis, and V. Zakharchenko, "Emerging MPEG Standards for Point Cloud Compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, mar 2019.

[9] 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.

[10] 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.

[11] T. Ochotta and D. 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.

[12] 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.

[13] R. Schnabel and R. 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] Y. Huang, J. Peng, C. C. Jay 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, Mar. 2008.

[20] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189?206, Apr 2013.

[21] 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.

[22] M. Krivokuca, M. Koroteev, and P. A. Chou, "A volumetric approach to point cloud compression," Sept. 2018.

[23] Diogo C. Garcia and Ricardo L. de Queiroz, "Context-Based Octree Coding For Point-Cloud Video," in *Proc. of ICIP 2017*, Sep 2017, pp. 1412–1416.

[24] E. Pavez, P. A. Chou, R. L. de Queiroz, and A. Ortega, "Dynamic Polygon Cloud Compression," *CoRR*, vol. abs/1610.00402, 2016.

[25] F. Pomerleau, F. Colas, and R. Siegwart, "A Review of Point Cloud Registration Algorithms for Mobile Robotics," *Found. Trends Robot*, vol. 4, no. 1, pp. 1–104, May 2015.

[26] S. Milani, "Fast Point Cloud Compression Via Reversible Cellular Automata Block Transform," in *Proc. of ICIP 2017*, Sept. 2017, pp. 2050–2054.

[27] 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.

[28] S. Limuti, E. Polo, and S. Milani, "A Transform Coding Strategy for Voxelized Dynamic Point Clouds," in *Proc. of IEEE ICIP 2018*, Oct. 2018, pp. 2954–2958.

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

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

[31] Franois Graner and James Glazier, "Simulation of biological cell sorting using a two-dimensional extended Potts model," *Physical review letters*, vol. 69, pp. 2013–2016, 10 1992.

[32] MPEG 3DG and Requirements, "Common test conditions for point cloud compression - doc. n17229," in *ISO/IEC JTC1/SC29/WG11 Coding of Moving Pictures and Audio Meeting Proceedings*, Oct. 2017, files: N17229_PCC-CTC.docx.

[33] 3DG-PCC team, "Point Cloud Compression Experimental Software SVN repository," web site, 2018, http://wg11.sc29.org/svn/repos/MPEG-04/Part16-Animation_Framework_eXtension_AFX/trunk/3Dgraphics/3DG-PCC/tags/hobart.

[34] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *2017 IEEE International Conference on Image Processing (ICIP)*, Sep. 2017, pp. 3460–3464.

[35] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves (VCEG-M33)," in *presented at the* 13^{th} *ITU VCEG Meeting*, Austin, TX, USA, Apr. 2 – 4, 2001, VCEG-M33.

[36] K. Mammou, "PCC Test Model Category 2 v0, in ISO/IEC JTC1/SC29/WG11 Doc. N17248, Macau, China," 2017.