A TRANSFORM CODING STRATEGY FOR VOXELIZED DYNAMIC POINT CLOUDS

Simone Limuti, Enrico Polo, Simone Milani

University of Padova, Dept. of Information Engineering, e-mail: {simone.limuti, enrico.polo}@studenti.unipd.it, simone.milani@unipd.it

ABSTRACT
With the advent of virtual and augmented reality applications, 3D and free-viewpoint representations have evolved towards solid scene models using meshes and point clouds. Recent works have been addressing point clouds compression via octree-based hierarchical strategies in order to enable a multi-resolution coding and visualization at a reasonable computational cost.

This paper presents a voxelized dynamic point cloud coding scheme that combines a Cellular Automata block reversible transform for geometric data with a region adaptive transform for color data. Temporal redundancy is removed using a low-complexity prediction scheme to minimize the computational complexity and reduce the coded bit rate. 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

1. INTRODUCTION AND RELATED WORKS
The wide availability of different methods and devices for real-time 3D capture have allowed the inclusion of three-dimensional dynamic models of real persons and objects in augmented and virtual reality applications[1]. Consequently, the acquired 3D data can be available in a wide variety of formats which present different levels of versatility and adaptability in terms of compression and transmission performances. Multiview-plus-depth video sequences 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 [2]). In computer graphics, polygonal meshes are widely-used and prove to be extremely effective in representing single 3D objects [3]; unfortunately, the generation of a high quality 3D mesh requires a significant amount of calculation because of polygonal fitting and refining operations. As a matter of fact, dynamic point clouds have been recently considered as versatile solution to model moving 3D objects without requiring excessive computing power or processing delays [4, 5]. Point clouds are usually generating sampling 3D object surfaces and characterizing each 3D point with color and orientation attributes [6]; the capturing of real objects often generates sparse set of points which requires efficient algorithms for their compression and organization [7, 8]. A few initial solutions adopted a 2D wavelet transform based scheme [9] or a multi-resolution decomposition of the 3D points [10].

Recent works have instead adopted hierarchically-iterative data coding schemes [11]. Such approach divide the object volume into a voxel grid, where the state of each element or voxel depends on whether the voxel contains points or not. This voxelized geometry can then compressed by octree-based strategies [6] into several quality layers which allow reconstructing the original 3D model at different Level-Of-Details (LODs). It is possible to associate color information to each voxel; its compression can take advantage of the knowledge of geometry to reduce the redundancy of the different components.

Redundancy can be reduced along time dimension as well. Block-based temporal prediction proves to be 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 [12]. As a matter of fact, most of the proposed approaches perform a motion compensation in the 3D point cloud domain [13, 14] 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 [15]. 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 [12].

The current paper presents a new compression scheme for voxelized dynamic point clouds that adopts a hierarchical non-linear transform coding scheme for voxel geometry and a region adaptive transform for color information. Along the temporal dimension, voxel color components are predicted from the previous frames. The proposed coding scheme ex-
tends the work in [16] to colored and dynamic 3D models obtaining a better compression performance that other octree-based coding solutions.

In the following, Section 2 reports the general structure of the coder, with subsection 2.1 presenting the transform coding strategy for the geometry, and subsection 2.2 describing the transform color coding scheme. Subsection 2.3 describes the temporal prediction strategy. Experimental results are in Section 3 and final conclusions in Section 4 end the paper.

2. THE PROPOSED CODER

At each instant, every point cloud is voxelized into a regular grid of \(N \times N \times N\) elements. The voxel at coordinates \((x, y, z)\), \(x, y, z = 0, \ldots, N - 1\), can be characterized by two types of data: geometric information \(g(x, y, z)\) and color information \(i_c(x, y, z)\):

\[ s(x, y, z) = g(x, y, z) \ldots g(x + 1, y + 1, z + 1) \]

which can be represented by an integer number in the range \([0, 255]\). We will denote this resolution level as LOD \(< q >\).

In the Cellular Automata (CA) modelling framework [17], each voxel is associated to an automata whose state (the voxel value 0 or 1) evolves depending on the state of its neighbours. From this assumption, it is possible to change the voxel/automata states in \(s(x, y, z)\) into a new string \(s'(x, y, z)\) according to a permutation function \(s'(x, y, z) = P(s(x, y, z))\), which was designed to maximize the coding performance.

After this decomposition, a new voxel volume \(g'\) with halved dimensions \((N/2 \times N/2 \times N/2)\) is generated setting \(g'(m, n, t) = 1\) if at least one state in \(s(x, y, z)\) is non-null or setting \(g'(m, n, t) = 0\) otherwise. The resolution level of this new volume will be referenced as LOD \(< q - 1 >\) and the coding procedure is iterated. The proposed transform operates similarly to a Discrete Cosine Transform on binary values, concentrating most of the energy on a few spatial frequencies (i.e., maximizing the number of zero coefficients), and requiring a minimum computational complexity.

Considering that \(s'(x, y, z) = 0\) is not coded (because it can be inferred from the previous LOD), we designed \(P\) in order to maximize the number of 1s in \(s'(x, y, z)\), i.e., the transform is computed on the probability distribution of \(s(x, y, z)\) from a set of training data. The idea is to assign \(s'(x, y, z)\) with the highest number of 1s to the most probable \(s(x, y, z)\), made exception for \(s'(x, y, z) = s(x, y, z) = 0\) (like in [16]).

To this purpose, we adopted a spatially-varying transform \(P\), where the mapping policy depends on the values of the voxels adjacent to the current octets. Figure 2 reports 3 different configurations (named C1, C2, and C3) which select different neighbouring voxels, whose values have already been coded and can be used to define a local context that can determine the employed transform or the probability distribution employed by the arithmetic coder.

---

Footnote: Index \(c\) denotes the color component \((R, G, B\) or \(U, V)\).
For a given configuration $C_h$, $h = 1, 2, 3$, the proposed algorithm compute the context value $v_{h,k}$ (which is determined by the neighbouring pixels) for the $k$-th $2 \times 2 \times 2$ octant. After computing all the $v_{h,k}$ values for the current volume, each context value is associated to a probability distribution for $s$. This leads to 256 different probability distributions. Using the algorithm reported in [16] to estimate $P(\cdot)$ from a given probability mass function, the coder generates a set of $P(\cdot)$ that are going to be used adaptively depending on the values of the neighbouring voxels. The efficiency of configurations $C_1$, $C_2$, and $C_3$ were tested in terms of bit rate reduction, and experimental results are reported in Section 3.

The generated transformed octets $s^P(x,y,z)$ are then compressed by an adaptive binary arithmetic coders according to a raster scan order, and the coding process is then restarted for the following LOD.

Then, geometric information can be used to drive the compression of color components, as described in the following subsection.

### 2.2. Color coding

After coding voxels $g(x,y,z)$, the proposed coding scheme can re-use this information to drive the compression of color data. Starting from the RAHT coding strategy [18], an adaptive separable $2 \times 2$ transform is progressively applied along each axis.

Assuming that $x = 2m$, $y = 2n$, $z = 2t$ operating at LOD $< q >$, the adopted transform is similar to that in [18], made exception for the fact that an integer rescaling factor is introduced, i.e.,

$$
\begin{pmatrix}
I_c(x,y,z) \\
I_c(x+1,y,z)
\end{pmatrix} = \frac{1}{K} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} i_c(x,y,z) \\
i_c(x+1,y,z)\end{pmatrix} \tag{2}
$$

where we assume that it is operated along the x axis and $g(x,y,z) = g(x+1,y,z) = 1$. The coefficient $I_c(x,y,z)$ is referred as DC coefficient, while $I_c(x+1,y,z)$ is the AC coefficient. If only $g(x,y,z) = 1$, then the DC coefficient $I_c(x,y,z)$ is $i_c(x,y,x)$; on the contrary, if only $g(x+1,y,z) = 1$ then $I_c(x,y,z) = i_c(x+1,y,x)$. In these latter cases, no AC coefficient is generated. Then, $I_c(x,y,z)$ 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 $I_c(x,y,z)$ is then sent to LOD $< q-1 >$, i.e., $i_c(m,n,t) = I_c(x,y,x)$. The transform is then iterated on LOD $< q-1 >$.

The resulting DC and AC coefficients are then quantized 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.

### 2.3. Exploiting temporal correlation

The coded bit rate can be utterly reduced by exploiting the temporal correlation for both $g(x,y,z)$ and $i_c(x,y,z)$. After the first voxel frame, the previously-coded information can be used to improve the compression performance for the current voxel frame. Due to the different nature of $g(x,y,z)$ and $i_c(x,y,z)$, we used two prediction strategies.

#### 2.3.1. Temporal correlation for geometry

As it was anticipated in the Introduction, the prediction of geometry voxels proves to be quite difficult since an accurate matching is not possible. Nevertheless, temporal correlation was exploited in the computation of the best transforms $P_{\cdot}$. In fact, the compression performance is maximized whenever the adopted transforms are tailored to the input data. This implies estimating $P_{\cdot}$ on the current voxel volume and coding $P_{\cdot}^{-1}(\cdot)$ 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 $P_{\cdot}$ 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 $P_{\cdot}$ which are going to be used for the following frame.

#### 2.3.2. Color prediction

Differently from the case of geometry, color prediction can lead to a significant bit rate reduction. Therefore, after coding the first color voxel frame, the difference

$$
d_c(x,y,z) = i_c(x,y,z) - i_{c,r}(x,y,z) \tag{3}
$$

(with $i_{c,r}(x,y,z)$ being the corresponding voxel colors in the previous reconstructed frame) is computed and processed by the color transform coder described in subsection 2.2 (Fig. 1b). When $g_c(x,y,z) = 0$ and $g(x,y,z) = 1$, there is not a reference color for $i_c(x,y,z)$ in the previous voxel frame; the reference is then generated averaging the values of the non-empty neighbouring voxels.

### 3. PERFORMANCE EVALUATION

The proposed coder was tested on four voxelized dynamic point clouds sequences [19]: David, Phil, Ricardo and Sarah. In each case, frames were voxelized into $512 \times 512 \times 512$ voxels. First, the performance of geometry compression was evaluated coding the considered sequences with the following five methods: the entropy-encoded Octree
(EO), where the generated octets are coded using an arithmetic coder; the solution by Kammerl et al. in [20] (KA), the coder in [12] (GA), and the three different context layouts (C1, C2, C3) for the proposed method.

The first frame was Intra coded, while for the following frames temporal correlation was exploited. Although the probability model and the CA transform were adapted after the encoding/decoding each frame, it is possible to reduce the update frequency to once every $M = 8$ frames with a negligible performance loss.

In Table 1, the obtained average rates are shown in bits per occupied voxel (bits/ov). The proposed encoding strategy yields to a rate reduction of 40% with respect to EO approach, while the rate savings are 43% and 36% with respect to KA and GA approaches. It can be noticed that the choice of the context layout slightly affects the outcomes.

Figure 3 reports a graph representing the bits/ov measures achieved by our coding scheme, by the plain Octree (no arithmetic coding) and by the encoded Octree, for the first 200 frames of the sequence Ricardo. The curves corresponding to our coding scheme start at the same value obtained by the encoded Octree strategy; the CA transform built on only the first frame is sufficient to provide almost a 40% rate reduction with respect to the EO. The rate gets further reduced during the encoding of the following dozen of frames, as the CA transform gets updated exploiting a growing set of frames.

The computational complexity of the proposed approach for geometry coding was compared with EO. On average, the encoding procedure requires 60 ms to be performed, while 67 ms are spent by the decoding procedure; these values represent a complexity increase of 25% and 5% for the encoding and decoding procedure respectively, although further optimizations are possible. In order to evaluate color compression performances, the PSNR-vs-rate plots obtained for two tested sequence are reported in Fig. 4.

![Fig. 3. Rates in bits per occupied voxel obtained for each frame of the sequence Ricardo by the five tested methods.](image)

![Fig. 4. RD plots for David (left) and Ricardo (right); all Intra set-up (blue), inter set-up (red).](image)

![Fig. 5. RD plots comparison of the sequence Ricardo. The blue curve refers to the approach in [18], while the red one is referred to the proposed scheme.](image)

<table>
<thead>
<tr>
<th></th>
<th>EO</th>
<th>KA</th>
<th>GA</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>david</td>
<td>2.04</td>
<td>2.09</td>
<td>1.83</td>
<td>1.19</td>
<td>1.23</td>
<td>1.21</td>
</tr>
<tr>
<td>phil</td>
<td>2.09</td>
<td>2.13</td>
<td>1.83</td>
<td>1.29</td>
<td>1.32</td>
<td>1.30</td>
</tr>
<tr>
<td>ricardo</td>
<td>2.01</td>
<td>2.39</td>
<td>2.22</td>
<td>1.17</td>
<td>1.20</td>
<td>1.20</td>
</tr>
<tr>
<td>sarah</td>
<td>2.02</td>
<td>1.92</td>
<td>1.70</td>
<td>1.17</td>
<td>1.21</td>
<td>1.19</td>
</tr>
<tr>
<td>average</td>
<td>2.04</td>
<td>2.13</td>
<td>1.89</td>
<td>1.21</td>
<td>1.24</td>
<td>1.22</td>
</tr>
</tbody>
</table>

Table 1. Average rates in bits/ov for geometry coding.

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.

4. CONCLUSIONS
5. REFERENCES


