Geometry Coding for Dynamic Voxelized Point Clouds Using Octrees and Multiple Contexts

Diogo C. Garcia, Senior Member, IEEE, Tiago A. Fonseca, Member, IEEE, Renan U. Ferreira, and Ricardo L. de Queiroz, Fellow, IEEE

Abstract—We present a method to compress geometry information of point clouds that explores redundancies across consecutive frames of a sequence. It uses octrees and works by progressively increasing the resolution of the octree. At each branch of the tree, we generate an approximation of the child nodes by a number of methods which are used as contexts to drive an arithmetic coder. The best approximation, i.e. the context that yields the least amount of encoding bits, is selected and the chosen method is indicated as side information for replication at the decoder. The core of our method is a context-based arithmetic coder in which a reference octree is used as reference to encode the current octree, thus providing 255 contexts for each output octet. The 255 × 255 frequency histogram is viewed as a discrete 3D surface and is conveyed to the decoder using another octree. We present two methods to generate the predictions (contexts) which use adjacent frames in the sequence (inter-frame) and one method that works purely intra-frame. The encoder continuously switches the best mode among the three and conveys such information to the decoder. Since an intra-frame prediction is present, our coder can also work in purely intra-frame mode, as well. Extensive results are presented to show the method’s potential against many compression alternatives for the geometry information in dynamic voxelized point clouds.

I. INTRODUCTION

Augmented and virtual reality applications (AR/VR) have gained popularity with the advance of latest technologies in capture and rendering, and have greatly impacted immersive communications [1]–[6]. Holoportation is an example of such an application, wherein telepresence is taken to higher levels through the use of solid modeling of persons and objects and through new rendering technologies [7]. The 3D objects and persons are captured in 3D, for example, in point clouds, and such information is then transmitted to a decoder for rendering. We are interested here in “voxelized” point clouds, wherein the capture space is divided into a number of small volumetric elements, or voxels, which have several implementation and real-time processing advantages over other 3D representations, such as lower rendering complexity (e.g. as opposed to triangular meshes) and removal of redundancies and inconsistencies from overlapping sensor maps [8]–[10]. For example, the capture space can be divided into 512 × 512 × 512 or 1024 × 1024 × 1024 voxels. For live action, we may capture 30 frames per second, so that we may have to convey information of about nearly 2^{35} voxels per second to the decoder. The information for a (voxelized) point cloud could be the state of each voxel, if it is occupied or not, along with its attributes, e.g. its color. In telepresence systems, the voxel space is well sparse, with less than 0.15% of the voxels being occupied. Hence, it may be easier to encode the position of the occupied voxels, rather than the state of all voxels in the space. As a result, voxels in omnidirectional point clouds are typically represented by its position (geometry information) and its color (attribute information). The whole point cloud frame is represented by an unordered list of six-tuple of information, like

\[ \{x_k, y_k, z_k, R_k, G_k, B_k\}, \]

where the \(xyz\) represent the grid position of the \(k\)-th occupied voxel and \(RGB\) represent its color.

We need compression for all this data in order to make the transmission practical. Geometry (spatial position of the voxels) and color attributes are usually separately encoded. Color can be, for example, compressed using the methods based on the region-adaptive hierarchical transform (RAHT) [9],[10], the Graph Transform [11],[12], or the Gaussian Process Transform (GPT) [13]. We are, however, concerned with the lossless compression of the geometry information. Note that we are only concerned with point clouds, not with any mesh-based representation nor in the compression of mesh-based geometry information.

Most methods that compress point cloud geometry information use octree coding [14]–[22]. Octree rate reduction has been achieved using predictive coding with local surface approximations [15],[16], inter-frame prediction [17],[18], cellular-automata block-reversible transform [19] and context modeling [20],[21],[22]. Other approaches include shape-adaptive wavelet coding [23] and multi-resolution decomposition [24]. Open-source libraries were also developed for the coding of point-clouds and meshes [25],[26].

We propose an advanced method based on octrees that arithmetically encodes the geometry information using a large number of contexts. To do so, we developed a new super-resolution (SR) technique, a previously-proposed SR technique [27] and an improvement over an intra-coding method [22] to generate a large number of possible contexts. The previously-proposed SR technique was also employed for point-cloud geometry coding, but using a different probability model [21]. We also propose a combination of these three methods that reduces the overall rate over each method individually. Furthermore, we encode the contexts’ frequency tables for the
arithmetic coder in a more efficient manner, allowing for 255 contexts to be conveyed to the decoder.

The core of our method is a context-based octree coding framework which is described in Sec. II. Sec. III describes the three context-generation methods used in this work, which are combined to form the overall coder which is described in Sec. IV. Extensive simulation results are presented in Sec. V, while the conclusions are presented in Sec. ??.

II. OCTREE CODING BASED ON MULTIPLE CONTEXTS

A. Octree coding

Let the voxel space comprise a cube of $M \times M \times M$ homogeneous voxels, where $M = 2^D$ and $D$ is the depth of the decomposition. Each of the $2^{3D}$ voxels may be occupied or not. The octree is an efficient method to encode the geometry. In the first level of the octree, the voxel space is partitioned into 8 smaller cubes of dimensions $M/2 \times M/2 \times M/2$, as depicted in Fig. 1. If a smaller cube has at least one occupied voxel, it is flagged occupied, otherwise it is flagged empty. We can then build a tree (octree) as in Fig. 1, wherein an occupied voxel, it is flagged occupied, otherwise it is flagged empty. We can then build a tree (octree) as in Fig. 1, wherein an occupied cube is an internal branch node while the empty one is a leaf node. In a second level, each of the occupied cubes is further partitioned in exactly the same way, each generating 8 cubes of dimensions $M/4 \times M/4 \times M/4$. If all cubes are split we obtain 64 cubes of reduced dimensions. The process can be repeated for up to $D$ levels, yielding $2^{3D}$ voxels in the last level.

The octree has nodes $\theta(n,k)$, which represents the $k$-th node at level $n$. $\theta(n,k)$ is the parent of nodes $\theta(n+1,8k)$ through $\theta(n+1,8k+7)$. A bit can be attributed to $\theta(n,k)$, so that 0 means the octant is empty and none of its descendants needs to be encoded, and 1 means that respective bits representing $\theta(n+1,8k)$ through $\theta(n+1,8k+7)$ need to be encoded, for example by grouping them in a byte $B(n+1,k)$. Figure 1 exemplifies this representation. The sequence of bytes $\{B(n,k)|n = 1, \ldots, D\}$ for all levels comprises the representation of the octree $O$. Note that $1 \leq B(n,k) \leq 255$ and not all bytes are encoded, just those whose parent node is not a leaf (i.e. $\theta(n-1, [k/8]) = 1$).

B. Multiple contexts for octree arithmetic coding

Let the current octree to be encoded be $O_c$ and assume there is a reference octree, for example a prediction of $O_c$, labeled $O_r$ which we would like to use to build a context for $O_c$. Let $O_c$ be described by $\{B(n,k)\}$ while $O_r$ is described by $\{B'(n,k)\}$, both having the same size. We use $B'(n,k)$ as a context to encode $B(n,k)$. Hence, there are 255 contexts to encode $B(n,k)$, i.e. we encode $B(n,k)$ using context $C$ ($1 \leq C \leq 255$) if $B'(n,k) = C$. So, we have 255 contexts to encode $B(n,k)$ which also assumes one out of 255 values. Therefore, we need somehow to convey to the decoder a map with $255 \times 255$ probabilities of each $q_{ij} = \text{Prob}(B(n,k) = i|B'(n,k) = j)$ in order to process all bytes and pass them through arithmetic coders, one for each context.

Our approach to do so assumes the probabilities $q_{ij}$ are derived from a map of frequencies $\phi_{ij}$, where we count the occurrence of each value for each context. Each entry $\phi_{ij}$ can assume any non-negative value to account for all bytes in $\{B(n,k)\}$, and is not limited to values between 0 and 255. Such a 2D map of all $255 \times 255$ $\phi_{ij}$ entries can also be seen as a point cloud in 3D space, hence, it can be represented using an octree $O_\phi$. Entries with $\phi_{ij} = 0$ are simply ignored and skipped. Such an octree can be represented using octets $\{B_\phi(\ell,m)\}$, which are encoded using an arithmetic coder, and all 255 frequencies of $\{B_\phi(\ell,m)\}$ are provided to the decoder.

Apart from the clear advantages of context coding, like reducing conditional entropy, the usage of multiple contexts may help parallelizing the process, which could be very important for real-time applications.

C. Encoding algorithm summary

In summary, in the proposed encoding method, we first perform the following steps:

- Given the current and reference octrees $O_c$ and $O_r$, and their respective representations $\{B(n,k)\}$ and $\{B'(n,k)\}$, one separates $O_c$ by the values in $O_r$ (i.e., $\{B(n,k) | B'(n,k) = C\}$, $1 \leq C \leq 255$), and calculates the corresponding frequencies

$$\phi_{ij} = \text{Count}(\{B(n,k) = i | B'(n,k) = j\}).$$

- Considering $\phi_{ij}$ as a point cloud in 3D space, obtain the corresponding octree $O_\phi$, represented by $\{B_\phi(\ell,m)\}$.

- Calculate the 255 frequencies of $\{B_\phi(\ell,m)\}$.

Then, the encoded information consists of:

1) The 255 frequencies of $\{B_\phi(\ell,m)\}$;
2) The $\{B_\phi(\ell,m)\}$, arithmetically encoded;
3) The first level of $O_c$, $B(1,1)$;
4) All the $\{B(n,k) | B'(n,k)\}$, arithmetically encoded separately by each context $\{B'(n,k)\}$ and with the probabilities $q_{ij}$ derived from $\phi_{ij}$. 
D. Decoding algorithm summary

The decoding process comprises the following steps:
1) Arithmetically decode \( \{ B_\phi(t, m) \} \), and calculate the corresponding frequency maps \( \phi_{ij} \).
2) Arithmetically decode \( \{ B(n, k) | B'(n, k) \} \) based on \( \phi_{ij} \);
3) Given the first level of \( O_c \), \( B(1, 1) \), reconstruct the second level of \( O_r \), \( B'(2, k) \), and then reconstruct the second level of \( O_c \), \( B(2, k) \), based on \( \{ B(n, k) | B'(n, k) \} \).
4) Repeat the previous step to reconstruct all further levels of \( B'(n, k) \) and \( B(n, k) \), stopping when reaching last level (\( \{ B(n, k) | B'(n, k) \} \) are no longer available).
5) Obtain the current frame of the point cloud from \( \{ B(n, k) \} \).

Of course, steps 3 and 4 assume \( O_c \) can be perfectly reconstructed at the decoder side. Since the octree progressively divides the \( M \times M \times M \) cube, the proposed method allows for progressive reconstruction at the decoder side, yielding different resolutions to different clients (computers, smartphones, tablets etc.) as they see fit.

III. CONTEXT GENERATION

From the previous section, it is clear we need to build the reference octree \( O_r \), described by octets \( \{ B'(n, k) \} \) from which to derive the contexts. We now describe the three methods we developed and are concurrently used in our coder.

A. Parent-node inheritance

Context generation by parent-node inheritance consists of creating a reference octree \( O_r \) by propagating an octet to all its immediate descendants [22]. An octet \( B(n, k) \) at level \( n \) represents 8 nodes whose 64 descendants are represented by 8 octets \( B(n + 1, 8k + m) \), \( 0 \leq m < 8 \). We then set \( B'(n + 1, 8k + m) = B(n, k) \) (recall that all nodes whose parent node is 0 are omitted). The propagation is carried from level to level, i.e. octets \( B(n, \cdot) \) from level \( n \) are propagated into next level as \( B'(n + 1, \cdot) \). Once the \( B(n + 1, \cdot) \) are encoded and known to the decoder, they can be used to propagate the references into level \( n + 2 \) as \( B' \) \( n + 2, \cdot \). The process is repeated until completing the tree.

The process is illustrated in Fig. 2, which depicts an example of a simple 3-level octree and its corresponding reference. In Fig. 2 the root of current octree \( O_c \) branches into \( B(1, 0) = 01100010 \) which indicates 5 leaves and 3 nodes which are parents of \( B(2, 1) = 11000000, B(2, 2) = 00000010, \) and \( B(2, 6) = 00000010 \). These octets branch out to octets \( B(3, 8) = 01110101, B(3, 9) = 00111000, B(3, 22) = 01100111, \) and \( B(3, 54) = 00001100 \). In order to build the reference octree, \( B'(1, 0) = 00000000 \) cannot be propagated from a previous level. Once \( B(1, 0) \) is encoded, we can make \( B'(2, 1) = B'(2, 2) = B'(2, 6) = B(1, 0) \). Once the 3 bytes of the second level are encoded and known at the decoder side, we can make \( B'(3, 8) = B'(3, 9) = B(2, 1) \), \( B'(3, 22) = B(2, 2) \) and \( B'(3, 54) = B(2, 6) \), thus completing \( O_r \).

This method of reference generation allows for intra-frame coding using contexts and is suited to still frames or to restart frames, wherein time-domain references are not available or not desirable.

B. Super-resolution by example

Whenever a sequence of point-cloud frames is available, similarities between time-adjacent frames can be explored to predict voxels at higher levels of the octree [27]. Each level of the octree of a given frame represents that frame at a different resolution. After decoding each level, the decoder should have at its disposal an incomplete low-resolution version of the current frame, as well as a previous frame at full-resolution. Using both frames as input, the encoder and the decoder can independently generate a super-resolved version of the current frame whose octree representation may serve as the reference octree \( O_r \) for the current octree \( O_c \). Furthermore, this process does not increase the overall rate with side information, as opposed to other typical coding methods such as motion estimation and compensation, which require motion vectors to be sent. From \( O_r \), we derive contexts to improve the encoding performance.

To ease notation, a frame \( V \) is a list of voxels \( \{ v(k) \} \) where each \( v \) is a vector with the \( x,y,z \) positions of the occupied voxels. Each \( v(k) \) is associated with its neighborhood, of size \( N \times N \times N \) centered at the position of \( v(k) \). Hence, \( \varphi(k) \) is the set of occupied voxels within such neighborhood of \( v(k) \). Our technique explores similarities between \( \varphi(k) \) in the current frame and reference frames, considering a \( W \times W \times W \) spatial window around \( v(k) \). Figure 3 presents an example in 2D of a current frame \( V_c \), its downsampled version \( V_c^d \) by a factor of 2, and \( 3 \times 3 \) neighborhoods \( \varphi(k) \) around each occupied pixel in \( V_c^d \).

Each level in the octree \( O_r \) of the current frame \( V_c \) represents one of its downsampled versions \( V_c^d \). We would like to obtain a super-resolved version \( \hat{V}_c \) of \( V_c^d \) based on similarities with a reference (previous) frame \( V_r \). \( \hat{V}_c \) can be obtained by matching neighborhoods between \( V_c \) and \( V_c^d \). Since \( V_c^d \) and \( V_r \) have different resolutions, we downsample the reference frame, yielding \( V_r^d \), such that best-match search
Fig. 3. 2D example of a current frame \( V_c \), its downsampled version \( V_r^d \) by a factor of 2, and the \( 3 \times 3 \) neighborhoods \( \varphi_r^d(k) \) around each occupied pixel in \( V_r^d \). Each occupied pixel in \( V_r^d \) is marked with a different symbol, and the corresponding neighborhoods are presented at the bottom of the image. For example, the pixel marked with a circle is surrounded by two pixels in its south and southeast positions.

can be performed for all voxels in two lower-resolution point-clouds \( V_r^d \) and \( V_c^d \). The corresponding child nodes \( \sigma(V_r^d) \) from \( V_r^d \) to \( V_c^d \) are then applied to \( V_c^d \), rendering \( \hat{V}_c \).

Because neighborhood matching is performed at a reduced resolution, the resolution of the associated displacement vectors is reduced as well. Better matches can be achieved by generating several downsampled versions of the reference frame considering incremental displacements in all directions, i.e. displacement that is below the resolution of the displacement vectors. For example, if the decimation factor is \( d_f = 2 \) in all directions, we cannot search on displacements of length 1. Hence, in this example, 8 downsampled versions \( V_{r,i}^{d,i} \), \( 0 \leq i \leq 7 \), can be generated by considering \( xyz \) translations by \( \{0,0,0\}, \{0,0,1\}, \{0,1,0\}, \{0,1,1\}, \{1,0,0\}, \{1,0,1\}, \{1,1,0\} \) and \( \{1,1,1\} \). Figure 4 illustrates this concept in a 2D scenario, where only \( xy \) translations are considered.

Using the neighborhoods \( \varphi_r^d(k) \) as support, matches are searched for each voxel \( \varphi_r^d(k) \) in \( V_c^d \), yielding individual corresponding child nodes \( \sigma(V_r^d) \). Specifically, we try to match the neighborhood \( \varphi_r^d(k) \) of \( V_c^d \) to any neighborhoods \( \{\varphi_r^d(k+h)\} \) in any of the incrementally-displaced frames \( V_{r,i}^{d,i} \), where we search both in \( i \) (index of displaced frame) and \( h \) (candidate displacement vector index), \( h \) is such that we search all valid neighborhoods around \( \varphi_r^d(k) \) within a spatial window of \( W \times W \times W \) voxels. Clearly, the search window size \( W \) creates a tradeoff between speed and super-resolution accuracy. The cost function \( \Psi(k,m,i) \) of the neighborhood matching in between \( \varphi_r^d(k) \), the \( k \)-th voxel in \( V_c^d \), and \( \varphi_r^d(m) \), the \( m \)-th voxel in \( V_{r,i}^{d,i} \), is defined as:

\[
\Psi(k,m,i) = \delta_H(k,m,i) + \omega(i)\delta_E(k,m,i) \frac{\omega(i)+1}{\omega(i)+1},
\]

where \( \delta_H(k,m,i) \) is the Hamming distance between the neighborhoods \( \varphi_r^d(k) \) of \( \varphi_r^d(m) \) and of the displaced \( \varphi_r^{d,i}(m) \), \( \delta_E(k,m,i) \) is the Euclidean distance between \( \varphi_r^d(k) \) and \( \varphi_r^{d,i}(m) \), and \( \omega(i) \) is the inverse of the Euclidean distance between the centers of mass of \( V_c^d \) and \( V_{r,i}^{d,i} \). A small Euclidean distance between these centers of mass indicates small motion between the current and reference frames, increasing the value of \( \omega(i) \), i.e. \( \Psi(k,m,i) \) favors smaller displacement vectors. For each \( k \), the voxel \( \varphi_r^{d,i}(m) \) which minimizes \( \Psi(k,m,i) \) is taken as the candidate to fill in the missing positions in order to infer \( V_c \). The octree at the current level for \( V_c \) can then be used as a context to enhance the frame encoder performance.

Figure 5 is a 2D illustration of the whole method of super-resolving one pixel in the downsampled version \( V_c^d \) of the current frame. Given the pixel’s neighborhood \( \varphi_r^d(k) \) and 4 downsampled versions \( V_{r,i}^{d,i} \) of the reference frame \( V_r \), the best match between \( \varphi_r^d(k) \) and the neighborhoods \( \varphi_r^{d,i}(k) \) in \( V_{r,i}^{d,i} \) is found, based on Eq. 1. The child node \( \sigma(V_r^d) \) (the upsampled version of the center pixel in \( \varphi_r^{d,i}(k) \)) is then used as the super-resolution for the pixel in \( V_r^d \). This process is repeated for all pixels in \( V_c^d \) in order to create \( \hat{V}_c \).

C. Super-resolution by neighborhood inheritance

Another way of super-resolving the current downsampled frame \( V_c^d \) is by creating from \( V_r \) a dictionary of children nodes based on the \( N \times N \times N \) neighborhood \( \varphi_r^d(k) \) around each voxel in \( V_r^{d,i} \) (i.e., considering incremental motion in all directions), as illustrated in Fig. 6. In this case, the dictionary is built from the whole reference frame \( V_r \), and the child node is inherited purely based on the current neighborhood. For voxels at the border of the 3D space, the non-existing neighbors are considered unoccupied.

In order to build our dictionary, we look at all data available at both the encoder and decoder and we estimate the most likely descendants for a given neighborhood around a voxel, i.e.

\[
E(\sigma(v(k))|v(k))
\]

The data available to estimate the conditional expected values are the \( V_r^{d,i} \). We scan each occupied voxel \( \varphi_r^{d,i}(k) \) of \( V_r^{d,i} \) and build a list of child nodes \( \sigma(\varphi_r^{d,i}(k)) \) for each neighborhood \( \varphi_r^{d,i}(k) \). Note that distinct voxels with the same neighborhood may lead to different children. The expected
Fig. 5. Full example of the super-resolution of one pixel in 2D. Given a pixel in the downsampled version $V_d$ of the current frame, its neighborhood $\varphi_d^c(k)$ and 4 downsampled versions $V_{d,i}^r$ of the reference frame $V_r$, the best match between $\varphi_d^c(k)$ and the neighborhoods $\varphi_{d,i}^r(k)$ in $V_{d,i}^r$ is found. From there, the upsampled version of the center pixel in $\varphi_{d,i}^r(k)$ is used as the super-resolution for the pixel in $V_d^c$.

Fig. 6. Building a dictionary of children nodes based on the current voxel’s neighbors.

The value of $\sigma$ is obtained by averaging all entries for each descendant. Because of the large number of table entries, many neighborhood values may not exist in the point cloud (no voxel has that specific set of neighbors). For these cases, $\sigma(V_{d,i}^r(k)) = \{1, 1, \ldots, 1\}$, meaning that all children octants should be occupied when these neighborhoods are referred to at the super-resolution step. Based on this dictionary of pairs $\varphi_{r,i}^d(k)$ and $\sigma(V_{r,i}^d(k))$, $V_c^d$ is super-resolved, and the corresponding octree at the current level can be used as a context.

IV. FRAME CODER

The context-driven encoder described in Sec. II would work well as long as accurate references are given to build meaningful contexts. Each of the three context generation methods, here described, would provide excellent results. However, we have found that a better coding method arises if we test all the three methods for each octree braching, i.e. for each $B(n,k)$ we compute three references $B_0(n,k)$ (intra-frame), $B_1(n,k)$ (SR using neighborhood) and $B_2(n,k)$ (SR by example), corresponding to each of the three methods described in Sec. III. We pick the method that yields the lowest Hamming distance, i.e.

$$\arg\min_x \delta_H(B(n,k), B_x(n,k))$$

where $\delta_H$ denotes the Hamming distance. The reasoning to use Hamming distances is to privilege fewer combinations of $(B(n,k) = i, B_x(n,k) = j)$, thus biasing the frequencies $\phi_{ij}$ towards fewer points.

Fig. 7. Proposed coder diagram.

The selection is adaptive and that information has to be transmitted to the decoder. We opted for encoding the information using an arithmetic coder and sending the probabilities of each selection beforehand. Obviously, an intra-coded frame may only require a global signaling rather than any local selection information. If we adapt the selection for each octet $B(n,k)$ the overhead would be too large. In order to reduce the overhead, we group the octets. We found that grouping the octets by their parent octet, which would combine up to 8 octets, is enough to reduce the overhead so that it accounts for a small percentage of the overall rate. In other words, we combine the octets in larger vectors as:
\[ B(n, k) = [B(n, 8k), \ldots, B(n, 8k + 7)] \]
\[ B'_n(n, k) = [B'_n(n, k), \ldots, B'_n(n, 8k + 7)] \]

where one may note that not all octets exist. Hence, if \( B(n, k) \) exists, it may contain 1 through 8 octets, since \( B(n, i) \) does not exist if \( B(n - 1, i) = 0 \). We, then, select the same context generation method for the whole group, picking the one that minimizes the Hamming distance in between octet groups as:

\[ \arg\min \, \delta_H(B(n, k), B'_n(n, k)) \]

In this way, adaptation overhead is reduced while still generating valuable contexts.

Taking all of this into consideration, Fig. 7 illustrates the proposed coder.

V. Results

Tests for the proposed methods were carried using six point-cloud sequences in \( 512 \times 512 \times 512 \)-voxel resolution (Andrew9, David9, Man, Phil9, Ricardo9 and Sarah9), and four sequences in \( 1024 \times 1024 \times 1024 \)-voxel resolution (Loot10, Longdress10, Redandblack10 and Soldier10) [9],[28],[29]. Figure 8 presents the first frame of each sequence, and Fig. 9 presents frames 101 to 105 of sequence Man, where rotation and translation of the subject can be seen. The average rate for the first 200 frames of each sequence was calculated for several coders, as indicated in Table I, in accordance with the latest and best results in the field. For arithmetic coding, an implementation for byte-sized symbols was used. We used \( W = 20 \) when generating contexts using super-resolution by example.

Table II presents the average rate in bits per occupied voxel (bpov) for each coder, as well as the average percentage-point (p.p.) rate gain over the octree representation. From the table we can infer that:

1) Previous methods [18],[26] are not competitive with the proposed method in any of its fixed forms nor in its adaptive form;
2) The use of super-resolution methods greatly improve rate performance, with 15.8 p.p. (SR-NI) and 11.3 p.p. (SR-Ex) gains over the proposed intra mode, respectively;
3) The proposed frame-coder offers overall improvement over non-adaptive context-generation methods, yielding percentage-point gains of 20.6, 4.8 and 9.3 over the fixed PNI, SR-NI and SR-Ex context generation methods, respectively.

Figures 10(a) and 10(b) present the rate on a frame basis, in bpov, for some of the coders in Table I, considering sequences Loot10 and Soldier10. These results are highly representative of the performance of the proposed methods, which offer consistent rate gains throughout the sequences. Furthermore, it can be seen that super-resolution by example may occasionally outperform the super-resolution by neighborhood inheritance method, depending on the motion quantity and on the search window. Quieter sequences tend to favor the former. Of course, an increase in the search window size \( W \) leads to an increase in computational complexity.

Figure 11 illustrates the frequency map \( \phi_{ij} \) of methods \( P(\text{Full}) \) and \( P(\text{PNI}) \), computed for the second frame of sequence Ricardo9, which is very representative. Even though \( \phi_{ij} \) assumes large values, requiring up to 11 bits for its representation, the map is highly sparse. Only 6.5% entries are non-null in the \( P(\text{Full}) \) case, and 8.3% in the \( P(\text{PNI}) \) case. Hence, fewer contexts are much more frequent, which greatly improves the arithmetic coder performance. Its sparsity also helps to reduce the rate to encode \( O_\phi \) and explains why the method in Section II is so efficient. Other sequences present very similar characteristics, as shown in Table III (statistics for the second frame of each sequence).

Figure 11 shows that the diagonals of the frequency maps \( \phi_{ij} \) are much more occupied than other regions. The diagonals account for the cases where \( B'(n, k) = B(n, k) \), which minimizes their Hamming distance, as defined in Section IV. The diagonal for \( P(\text{Full}) \) is then more occupied than the diagonal for \( P(\text{PNI}) \), as Hamming-distance minimization is performed in the former case. For the \( P(\text{PNI}) \) case, however, \( \phi_{ij} \) still has most non-null values in its diagonal, accounting for a good intra-based performance.

Figure 12 depicts the relative context selection rates, computed for several sequences. Both SR methods are chosen more often than the parent-node inheritance method, which is inherently an intra-based approach. However, the latter is chosen near 10% of the time! This shows that the proposed inter-frame coding method can eventually benefit from intra-frame coding.

The prototype for this work was mainly developed using Matlab/Octave frameworks, with some CPU-intensive modules natively implemented, so that it does not work in real-time, and implementation improvements can surely be applied. Table IV presents time profiling for the proposed method, for the second frame of some representative sequences. The majority of the time is spent in finding the best predictions in terms of rate. SR-based methods correspond to more than 80% of execution time for the test set and this trend seems to be independent of resolution. The tests were made on a personal computer with an Intel Core i7-4500U CPU @ 1.80GHz, with a cache size of 4096 KB and 16 GB of RAM.
Fig. 8. First frame of each of the tested sequences. In clockwise order: Loot10, Longdress10, Man, Redandblack10, Soldier10, Sarah9, David9, Andrew9, Ricardo9 and Phil9.

TABLE II
AVERAGE RATE IN BITS PER OCCUPIED VOXEL FOR THE PROPOSED SCENARIOS (TABLE I), AND THE RESPECTIVE GAINS OVER THE OCTREE REPRESENTATION.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>OR</th>
<th>AC(OR)</th>
<th>CWI</th>
<th>Draco</th>
<th>Prev-Intra</th>
<th>Prev-Inter</th>
<th>P(PNI)</th>
<th>P(SR-NI)</th>
<th>P(SR-Ex)</th>
<th>P(Full)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew\textsuperscript{9}</td>
<td>2.58</td>
<td>2.04</td>
<td>2.49</td>
<td>4.77</td>
<td>1.80</td>
<td>1.69</td>
<td>1.83</td>
<td>1.51</td>
<td>1.56</td>
<td>1.36</td>
</tr>
<tr>
<td>David\textsuperscript{9}</td>
<td>2.62</td>
<td>2.05</td>
<td>2.50</td>
<td>4.35</td>
<td>1.73</td>
<td>1.83</td>
<td>1.77</td>
<td>1.41</td>
<td>1.58</td>
<td>1.33</td>
</tr>
<tr>
<td>Longdress\textsuperscript{10}</td>
<td>3.00</td>
<td>2.35</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Loot\textsuperscript{10}</td>
<td>2.98</td>
<td>2.32</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.69</td>
<td>1.16</td>
<td>1.33</td>
<td>1.03</td>
</tr>
<tr>
<td>Man</td>
<td>3.16</td>
<td>2.65</td>
<td>3.13</td>
<td>5.04</td>
<td>2.32</td>
<td>2.29</td>
<td>2.36</td>
<td>1.83</td>
<td>1.83</td>
<td>1.58</td>
</tr>
<tr>
<td>Phil\textsuperscript{9}</td>
<td>2.64</td>
<td>2.10</td>
<td>2.55</td>
<td>4.54</td>
<td>1.84</td>
<td>1.88</td>
<td>1.88</td>
<td>1.53</td>
<td>1.66</td>
<td>1.43</td>
</tr>
<tr>
<td>Redandblack\textsuperscript{10}</td>
<td>3.00</td>
<td>2.40</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.84</td>
<td>1.32</td>
<td>1.56</td>
<td>1.23</td>
</tr>
<tr>
<td>Ricardo\textsuperscript{9}</td>
<td>2.59</td>
<td>2.03</td>
<td>2.50</td>
<td>4.38</td>
<td>2.27</td>
<td>2.22</td>
<td>1.79</td>
<td>1.43</td>
<td>1.58</td>
<td>1.34</td>
</tr>
<tr>
<td>Sarah\textsuperscript{9}</td>
<td>2.61</td>
<td>2.04</td>
<td>2.49</td>
<td>4.86</td>
<td>1.75</td>
<td>1.70</td>
<td>1.79</td>
<td>1.43</td>
<td>1.52</td>
<td>1.30</td>
</tr>
<tr>
<td>Soldier\textsuperscript{10}</td>
<td>3.00</td>
<td>2.42</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.76</td>
<td>1.18</td>
<td>1.17</td>
<td>0.93</td>
</tr>
<tr>
<td>Average</td>
<td>2.82</td>
<td>2.24</td>
<td>2.61</td>
<td>4.69</td>
<td>1.95</td>
<td>1.94</td>
<td>1.83</td>
<td>1.40</td>
<td>1.53</td>
<td>1.26</td>
</tr>
<tr>
<td>Gain over OR</td>
<td>0.0%</td>
<td>20.3%</td>
<td>7.4%</td>
<td>-66.4%</td>
<td>30.7%</td>
<td>31.3%</td>
<td>34.5%</td>
<td>50.3%</td>
<td>45.8%</td>
<td>55.1%</td>
</tr>
</tbody>
</table>

TABLE III
FREQUENCY MAP $\phi_{ij}$ PROFILING FOR DIFFERENT SEQUENCES.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>P(Full)</th>
<th>Bit depth</th>
<th>Occupancy</th>
<th>P(PNI)</th>
<th>Bit depth</th>
<th>Occupancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew\textsuperscript{9}</td>
<td>12 bits</td>
<td>9.1%</td>
<td>11 bits</td>
<td>11.2%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>David\textsuperscript{9}</td>
<td>12 bits</td>
<td>8.7%</td>
<td>11 bits</td>
<td>11.3%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Man</td>
<td>12 bits</td>
<td>4.8%</td>
<td>10 bits</td>
<td>10.1%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phil\textsuperscript{9}</td>
<td>12 bits</td>
<td>11.2%</td>
<td>11 bits</td>
<td>14.1%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ricardo\textsuperscript{9}</td>
<td>11 bits</td>
<td>6.5%</td>
<td>10 bits</td>
<td>8.3%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sarah\textsuperscript{9}</td>
<td>12 bits</td>
<td>8.6%</td>
<td>11 bits</td>
<td>11.0%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VI. CONCLUSIONS
We have presented new methods for the compression of geometry information in voxelized point clouds. The method is applicable to both static or dynamic point clouds. Four new methods were developed for this paper. First, a novel context-adaptive arithmetic-coded octree representation method was devised wherein a large number of contexts and frequencies could be generated with reduced overhead. This was possible by representing the frequency maps as octrees themselves. Second, an intra-frame coding method was developed for generating contexts based on a progressive representation of the octree. Two novel super-resolution methods were developed in order to create contexts for encoding the current octree based on statistics gathered in previous frames of the sequence. Each of these methods is a contribution on its own. Without adaptation, the proposed intra-frame octree coder is very
competitive against other lossless intra-frame geometry coder. Also, without adaptation, the proposed inter-frame context
generation techniques would lead to a state-of-the art coders
that would only be outperformed by the adaptive method. In
it, we combined all methods into one adaptive scheme that
outperforms each of the methods without adaptation.

Results show that our method provides nearly twice as much
compression compared to entropy-coded octree which until
recently was the preferred way for lossless compression of
point clouds. We plan to further extend our research in order to
make the compression lossy, thus allowing for further reducing
the bit-rate per voxel.

**REFERENCES**

izing the vision of immersive communication,” *IEEE Signal Processing
[3] P. A. Chou, “Advances in immersive communication: (1) telephone, (2)
Appl.*, vol. 9, no. 1s, 2013.
and S. Wee, “The road to immersive communication,” *Proc. IEEE*
a fully distributed immersive teleconferencing system with infrared dot
Trans. on Graphics*, Vol. 34, No. 4, Article 69, Aug. 2015.
real-time,” *Proc. 29th ACM User Interface Software and Technology
voxelization with application to image-based modeling,” in *High-
Fig. 11. Frequency map $\phi_{ij}$ for the second frame of sequence Ricardo9: (a) proposed method - $P(Full)$; (b) proposed method using only parent-node inheritance - $P(PNI)$. In these matrices, the columns account for context values, the rows account for the octree’s value, and the color indicates how much the octree value occurs for a given context. $\phi_{ij}$ is usually very sparse: only 6.5% of its positions are non-null in the $P(Full)$ case, and 8.3% in the $P(PNI)$ case. Results are best seen on a screen.

Fig. 12. Relative selection rates of contexts according to Table I for several sequences, considering every other 5 frames (first frame, sixth, eleventh etc.).

Diogo C. Garcia (SM’18), received the Electrical Engineer, M.Sc. and Ph.D. degrees in electrical engineering from the University of Brasilia, Brasilia, Brazil, in 2006, 2008 and 2012, respectively. He is currently a Faculty Member at the Gama College of Engineering of the University of Brasilia. His current research interests include image and video coding, super resolution, and multiview and 3-D processing.

Tiago A. Fonseca (M’13), received the Eng., M.Sc. and D.Sc. degrees from the Dept. of Electrical Engineering at Universidade de Brasilia, Brazil, in 2006, 2008 and 2012, respectively. In 2009-2015, he was technologist at CEPESC (GSI/PR). He is currently a Faculty Member at the Gama Engineering College of the Universidade de Brasilia. His main research interests are in image processing, video processing and scalable video coding.

Renan U. Ferreira, received the Electrical Engineer, M.Sc. and Ph.D. degrees in electrical engineering from the University of Brasilia, Brasilia, Brazil, in 2006, 2009 and 2015, respectively. He is currently a faculty member at the Gama College of Engineering of the University of Brasilia. His current research interests include image and video coding, biomedical image and point cloud processing.

Ricardo L. de Queiroz, (F’17) received the Engineer degree from Universidade de Brasilia, Brazil, in 1987, the M.Sc. degree from Universidade Estadual de Campinas, Brazil, in 1990, and the Ph.D. degree from The University of Texas at Arlington, in 1994, all in Electrical Engineering.

In 1990-1991, he was with the DSP research group at Universidade de Brasilia, as a research associate. He joined Xerox Corp. in 1994, where he was a member of the research staff until 2002. In 2000-2001 he was also an Adjunct Faculty at the Rochester Institute of Technology. He joined the Electrical Engineering Department at Universidade de Brasilia in 2003. In 2010, he became a Full (Titular) Professor at the Computer Science Department at Universidade de Brasilia. During 2015 he has been a Visiting Professor at the University of Washington, in Seattle.

Dr. de Queiroz has published extensively in Journals and conferences and contributed chapters to books as well. He also holds 46 issued patents. He is a past elected member of the IEEE Signal Processing Society’s Multimedia Signal Processing (MMSP) and the Image, Video and Multidimensional Signal Processing (IVMSP) Technical Committees. He is a an editor for IEEE Transactions on Image Processing and a past editor for the EURASIP Journal on Image and Video Processing, IEEE Signal Processing Letters, and IEEE Transactions on Circuits and Systems for Video Technology. He has been appointed an IEEE Signal Processing Society Distinguished Lecturer for the 2011-2012 term.

Dr. de Queiroz has been actively involved with IEEE Signal Processing Society chapters in Brazil and in the US. He was the General Chair of ISCAS’2011, MMSP’2009, and SBrT’2012. He was also part of the organizing committee of many SPS flagship conferences. His research interests include image and video compression, point cloud compression, multirate signal processing, and color imaging. Dr. de Queiroz is a Fellow of IEEE and a member of the Brazilian Telecommunications Society.