Fractional Super-Resolution of Voxelized Point Clouds

Tomás M. Borges, Diogo C. Garcia, *Senior Member, IEEE,* and Ricardo L. de Queiroz, *Fellow, IEEE*

***Abstract*—We present a method to super-resolve voxelized point clouds downsampled by a fractional factor, using lookup-tables (LUT) constructed from self-similarities from its own downsampled neighborhoods. Given a downsampled point cloud geometry V_d, and its corresponding fractional downsampling factor s, the proposed method determines the set of positions that may have generated V_d, and estimates which of these positions were indeed occupied (super-resolution). Assuming that the geometry of a point cloud is approximately self-similar at different scales, LUTs relating downsampled neighborhood configurations with children occupancy configurations can be estimated by further downsampling the input point cloud to V_{d^2}, and by taking into account the irregular children distribution derived from fractional downsampling. For completeness, we also interpolate texture by averaging colors from adjacent neighbors. We present extensive test results over different point clouds, showing the effectiveness of the proposed method against baseline methods.**

***Index Terms*—Point clouds, super-resolution, resampling.**

I. Introduction

RECENT technology evolution has enabled the capture and rendering of 3D structures, enhancing several immersive applications like tele-presence, 3D sensing for smart cities, and autonomous driving. Amongst the several alternatives of content representation in extended reality, point cloud (PC) imaging gained popularity due to its relatively low-complexity and high-efficiency in capturing, encoding, and rendering of 3D models.

A PC is a list of points in the 3D space, each with spatial coordinates (x, y, z) and attributes like colors, normals, reflectances, etc. For a single-color (RGB) attribute, a PC is defined by its geometry V and its color C sets:

$$V = \{\mathbf{v}(k)\}, \text{ with } \mathbf{v}(k) = (x_k, y_k, z_k), \text{ and}$$

$$C = \{\mathbf{c}(k)\}, \text{ with } \mathbf{c}(k) = (R_k, G_k, B_k).$$

The Moving Picture Expert Group (MPEG) has an effort into standardizing the representation and compression of PCs [1], [2]. Two codecs have been developed to this end: the Video-based Point Cloud Compression (V-PCC) [3] and the Geometry-based Point Cloud Compression (G-PCC) [4]. The former leverages existing video encoders to compress PCs by mapping 3D structures into 2D depth maps and texture "atlases", while the latter uses the 3D geometry properties for compression. In G-PCC, geometry and attributes are treated separately. The geometry is represented using the octree structure [5], then it is entropy encoded. Lossy geometry is obtained by performing an octree pruning. Additionally, a surface reconstruction approximation can be added to the lossy geometry bitstream using a series of triangles in a mesh-like structure, called *trisoup*. Attributes are transferred to the newly reconstructed geometry (recoloring). They are subband decomposed, using one of the available wavelet-like transforms—the region-adaptive hierarchical transform (RAHT) [6], or the predicting and lifting (*predlift*) [4] transform, quantized, and finally entropy encoded [1].

In order to use the octree representation for the geometry in G-PCC, the input PC needs to go through a voxelization pre-processing step, where the coordinates are quantized. Input coordinates are transformed such that all points lie within a bounding cube $[0, 2^d)^3$, for some non-negative integer parameter d. Voxels represent the center of any of the unit cubes $[i-0.5, i+0.5)\times[j-0.5, j+0.5)\times[k-0.5, k+0.5)$, for i, j, k integers between 0 and 2^d-1 [4]. All points from the original set that lie within a voxel's boundary are mapped to that voxel. This mapping step may result in multiple points with the same position—duplicate points. Usually, duplicate points are consolidated into one, and their attributes are averaged. Modern acquisition processes of real-world PCs usually imply in discrete volumetric samples being captured on a regular grid, which can be seen in many datasets [7]–[10].

The downsampling of PCs can be approached in two ways. The first is to decimate points of the original set without changing the voxel resolution. We refer to this as *set downsampling*. The second approach is to (re-)voxelize the PC using bigger voxels, i.e., with a lower volumetric resolution, referred to as *grid downsampling*. In this paper, we propose a super-resolution (SR) method that takes a grid-downsampled low-resolution (LR) voxelized PC as input, and combines the restrictions imposed by the voxel grid, its density, and self-similarities, to output a super-resolved version of that input, *i.e.*, with smaller voxels allowing for finer representation.

PC SR is the problem of creating a high-resolution (HR) PC from a LR version. Although many approaches exist, often they cannot be directly compared, either because of different number of inputs, different LR versions, or even the use of extra information.

In optimization-based PC SR, a cost function is defined, then new points are added seeking to minimize this function. Alexa *et al.* [11] proposed constructing a Voronoi diagram

Work partially supported by CNPq under grants 88887.600000/2021-00 and 301647/2018-6.

T. M. Borges is with the Electrical Engineering Department at Universidade de Brasilia, Brasilia, Brazil, e-mail: tomas@divp.org.

D. C. Garcia, is with the Gama Engineering College, Universidade de Brasilia, Brasilia, Brazil, e-mail: diogogarcia@unb.br.

R. L. de Queiroz is with the Computer Science Department at Universidade de Brasilia, Brasilia, Brazil, e-mail: queiroz@ieee.org.

on the 3D surface, then, inserting points in the vertices to minimize a moving least squares (MLS) cost function. Other works were also developed using the MLS cost function [12], [13], but all of those tend to over-smooth the geometry. An edge-aware solution was introduced by Huang *et al.* [14] to mitigate the over-smoothness of prior methods, relying on the accuracy of normals and on a thorough parameter tuning. Hamdi-Cherif *et al.* [15] combined local descriptors by their similarities for PC SR, however this approach required several PCs, normals calculations, and assumed surface smoothness. Dinesh *et al.* [16], [17] used Delaunay triangulation in the LR PC and optimize an L_1-norm graph-total-variation (GTV) cost function for neighborhood surface normals. It promotes piecewise smoothness in reconstructed 2D surfaces, under the constraint that the LR coordinates are preserved.

Deep learning was used for PC SR with the introduction of PU-NET by Yu *et al.* [18], which learns multi-scale features by downsampling the input and expands the point set via multi-branch multilayer perceptrons (MLP). Yu *et al.* also proposed EC-NET [19] an edge-aware network for point consolidation, alas, it requires a very expensive edge-notation for training. A progressive network, 3PU, was proposed by Wang *et al.* [20], to suppress noise and to preserve details in the upsampling geometry. It is computationally expensive, though, and requires a lot of data for training. PU-GAN [21] was designed to obtain more uniformly distributed SR results, with its major contribution and performance gains coming from the discriminator part. Graph convolutional networks (GCN) were used for PC SR by Wu *et al.* [22], and by Qian *et al.* [23]. PUGeo-Net [24] proposes to perform the upsampling by learning the first and second fundamental forms to represent the local geometry, although normals are required. Nonetheless, all these networks require retraining when different upsampling scales are required, making them cumbersome for dynamic applications. Recently, an independent work by Ye *et al.* [25] proposes Meta-PU, a network that supports upsamplings for arbitrary scales without the need of retraining, using meta-learning to predict the weights of the network and dynamically change behavior for each scale factor.

The voxelization process needed for G-PCC can severely affect some SR methods, by changing the requirements of input and output PCs. Octree-based PC SR, on the other hand, already cope with the G-PCC requirements. Garcia *et al.* developed two PC SR methods based on statistics gathered in previous frames of the sequence: SR by example and SR by neighborhood inheritance [26]. The first explores similarities between time-adjacent frames to predict voxels at higher levels of the octree [27]. The LR frame is super-resolved using the similarities from the previous frame at full-resolution. The second creates a dictionary of child nodes based on the neighborhood configuration from previous full-resolution frames. Then, the neighborhood from each occupied voxel is used to estimate its child nodes.

We propose to expand the neighborhood inheritance method to the intra-frame case and to generalize it for fractional scale factors, allowing for SR of arbitrary octree pruning. We also consider color interpolation together with the geometry upsampling for a complete PC SR approach.

857,966 points 2.99 bpov | 64,176 points 16.01 bpov | 62.130 points 2.67 bpov

Fig. 1: Downsampling approaches. Original PC, set (Poisson disk sampling), and grid downsampling.

II. Point cloud resampling

A. Downsampling

In set downsampling, points are usually decimated using some distance-based criterion, like Poisson disk sampling [28]. This kind of approach does not seek to lower the the PC resolution (bit depth) but its density. The advantage is that all the points in the LR version are also present in the original PC, which is desirable for interpolation. However, this kind of downsampling decreases the PC's spatial correlation, making its octree representation less efficient, and, consequently, rendering it less efficient for G-PCC.

In grid downsampling, points are decimated through voxelization. In this approach, the resolution is lowered but the density increases, when compared to the original PC. The remaining points in the LR PC cannot be rescaled to its original resolution without error. Yet, since the point's spatial correlation is larger, this LR PC can be more efficiently represented with an octree. Figure 1 showcases the downsampling of a PC using both approaches to arrive at approximately the same number of points. We calculated the rate, in bits per occupied voxel (bpov), necessary for the octree representation for each PC to illustrate the difference in spatial correlation among points in both cases.

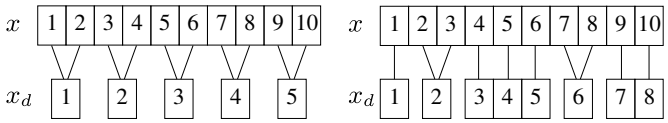
Since we are more interested in LR representations that can be efficiently encoded, for example, with G-PCC, we only focused on grid downsampling. This downsampling is achieved by dividing the geometry V by a scale factor $s > 1$, and rounding the results to maintain the integer grid. Duplicate points are merged, and their texture is averaged. Thus, we achieve a downsampled geometry V_d using:

$$V_d = \texttt{unique}\left(\texttt{round}\left(\frac{V}{s}\right)\right), \tag{1}$$

where $\texttt{unique}(X)$ is the function that returns only the unique vectors in the set X, and $\texttt{round}(\cdot)$ is the function that rounds the components of a vector to the nearest integer. The consolidation of duplicate position information is similar to the voxelization process.

V_d and V form a hierarchical tree structure, such that the former represents parent nodes, and the latter represents child nodes. This is illustrated in 1D in Fig. 2, where we can see that when s is an integer, the number of child nodes is equal for all positions in x. This downsampling process is regular as all parent nodes have the same number of children. When

s is not an integer, however, the number of child nodes varies depending on each parent node's position. For $1 < s < 2$, there are parent positions with only one child (uniparous), and others with two children (multiparous).



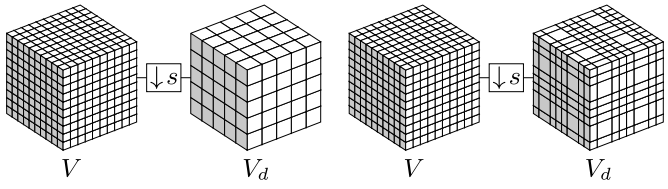
(a) $s = 2$ (b) $s = 1.25$

Fig. 2: Illustration of the downsampling process over a fully-occupied single-axis x. It is also possible to see a hierarchical tree configuration with x_d as parent nodes of x.

In 3D, regular downsampling (integer values of s) translates to every group of voxels in a $s \times s \times s$ cube in V being reduced to just one voxel in V_d, as depicted in Fig. 3(a). For this case, parent nodes in V_d have s^3 children. This represents a pruning of the original octree structure. When $s = 2^n$, $n = 1, 2, 3, \ldots$, the pruning occurs exactly[1] at level $d - n$. When $s \in \mathbb{Q}$, parent nodes in V_d have *up to* s^3 children. For example, when $1 < s \leq 2$ parent nodes may have 1, 2, 4 or 8 children, depending on each parent node's coordinate value, as depicted in Fig. 3(b). If a parent node has a multiparous coordinate value in x, and uniparous coordinates in y and z, it, thus, has $2 \cdot 1 \cdot 1 = 2$ child nodes, and we know that at least one of them must have been occupied in V. The number of possible children for a given parent can be generalized as

$$i_{\max} = (\lceil s \rceil - 1)^u \lceil s \rceil^m, \tag{2}$$

where u is the number of uniparous coordinates, m the number of multiparous coordinates (here we extend the meaning of uniparous to indicate parents with fewer children then the multiparous ones), and $\lceil\ \rceil$ means the ceiling operator. Although a non-integer value of s produces an irregular voxel grid, there is a pattern on such grid when s is rational in the form p/q, for $p > q$. We call fractional resampling the use of a non-integer value of s to perform down- or upsampling.



(a) $s = 2$ (b) $s = 1.25$

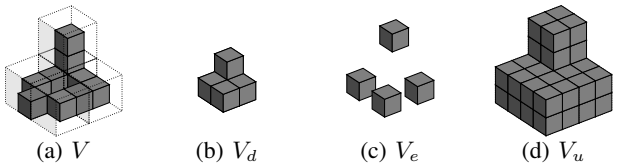
Fig. 3: Downsampling in the voxel grid. For a fractional value of s in (b) the different parenthood relationships are manifest.

B. Upsampling

We define PC upsampling as the inverse of the just-defined downsampling process. The space of the bounding cube containing the voxelized PC is again re-quantized, this time with the purpose of increasing the number of voxels inside the bounding cube. It can be done by the simple expansion of the downsampled geometry,

$$V_e = \texttt{round}\,(V_d \cdot s)\,. \tag{3}$$

However, it yields a very sparse PC. In order to maintain the density of the LR version in the upsampled version, we need to interpolate the missing points. The baseline technique for completing the missing points is the nearest-neighbor interpolation (NNI), which only sets as occupied all children from the parent nodes in V_d. The texture upsampling for the NNI usually follows the same idea used for the geometry: the colors from parent nodes are just replicated to their correspondent children. Figure 4 illustrates what happens to a geometry after being downsampled, expanded, and, finally, upsampled using NNI.



(a) V (b) V_d (c) V_e (d) V_u

Fig. 4: Child and parent node representation for $s = 2$. (a) Child nodes (V) surrounded by its parent nodes. (b) Parent nodes (V_d). (c) Expanded geometry. (d) NNI upsampling.

To express the NNI, we have that all child nodes from a parent voxel $\mathbf{v}_d(k)$ satisfy,

$$\texttt{round}(\mathbf{v}_u(i)/s) = \mathbf{v}_d(k), \tag{4}$$

for every $i = 1, 2, \ldots, i_{\max}$. Inversely,

$$\mathbf{v}_u(i) = \texttt{round}(s \cdot \mathbf{v}_d(k)) + \epsilon(i), \tag{5}$$

where $\epsilon(i)$ is the rounding error. Let $\mathcal{E}(k) = \{\epsilon(i)\}$ be the set containing all the $i_{\max}$ error samples for the parent node $\mathbf{v}_d(k)$. Thus, the set containing all possible children from $\mathbf{v}_d(k)$ is

$$\mathcal{V}_u(k) = \texttt{round}(s \cdot \mathbf{v}_d(k)) + \mathcal{E}(k). \tag{6}$$

Therefore, the geometry from the NNI upsampling is

$$V_u = \texttt{unique}\left(\bigcup_{k=1}^{K} \mathcal{V}_u(k)\right). \tag{7}$$

The colors for $\mathcal{V}_u(k)$ are $\mathcal{C}_u(k) = \{\mathbf{c}_u(i)\},\ i = 1, 2, \ldots, i_{\max}$, such that, $\mathbf{c}_u(i) = \mathbf{c}_d(k)$, and

$$C_u = \bigcup_{k=1}^{K} \mathcal{C}_u(k). \tag{8}$$

In order to improve quality of the upsampled geometry V_u, further processing can be applied to smooth both its geometry and its texture. Laplacian smoothing [29], for example, is used to smooth meshes and can be adapted to work on PCs, in order to improve NNI results by reducing the aliasing caused by this coarse interpolation. In the Laplacian smoothing, the position of each occupied voxel is updated by averaging the positions of occupied neighbors in a $3 \times 3 \times 3$ neighborhood.

[1]The term *exactly* is somewhat relaxed here, because of the use of `round`. To get the strictly exact pruning equivalence, Eq. (1) should be defined using the `floor` function instead.

III. Intra-frame super-resolution of voxelized point clouds

A. Proposed method

We took the idea of using a dictionary of child nodes from Garcia *et al.* [26], which was devised for the inter-frame case for SR of LR frames downsampled by a scale factor of $s = 2^n$. We, then, extrapolated it to the intra-frame case for LR PCs downsampled by fractional values of s. The dictionary creation process is similar to the one proposed by Garcia *et al.* [26]. However, it is done using the very same PC we want to super-resolve, and an additional step for parenthood stratification was introduced to cope with irregularities of the fractional downsampling, which makes for the creation of one dictionary for each parenthood condition. This ensures that the parent-child relation is preserved in the upsampling scheme.

Let $\varphi_M(\mathbf{v}(k))$ be a (M^3-1)-binary number indicating the occupancy of neighbor voxels inside an $M \times M \times M$ cube, defined as the neighborhood of the voxel $\mathbf{v}(k)$. The smallest neighborhood, when $M = 3$, leads to $3^3 - 1 = 26$ neighbors (adjacent voxels). Similarly, let the child occupancy configuration of parent voxel $\mathbf{v}_d(k)$ be defined as $\sigma(\mathbf{v}_d(k))$, a $\lceil s \rceil^3$-binary number indicating which of the possible children of $\mathbf{v}_d(k)$, i.e., $\mathcal{V}_u(k)$ in our notation, are indeed occupied. We take the input geometry V_d and perform yet another downsampling using the same scale factor s to generate its parent geometry V_{d^2}. In this way, we can find the child occupancy configuration for each parent voxel $\sigma(\mathbf{v}_{d^2}(k))$ and couple this information with its neighborhood configuration $\varphi_M(\mathbf{v}_{d^2}(k))$. In order to create the m-th entry of the dictionary, or look-up-table (LUT), we estimate the most likely child occupancy for each $\varphi_M(m)$,

$$\bar{\sigma}(m) = E\{\sigma(\mathbf{v}_{d^2}(k)) \mid \varphi_M(m)\}, \tag{9}$$

i.e., the expected value (bitwise mean) of all child occupancies sharing the same neighborhood configuration. Neighborhood configurations not present in the input data are associated with fully occupied child states. Table I illustrates such a LUT.

Using the scale factor s, we can stratify each parent voxel $\mathbf{v}_{d^2}(k)$ depending on the position and number of its possible children. Figure 5 illustrates the eight possible conditions for the parenthood stratification for a scale factor $1 < s \leq 2$. Note that in such case, there is no need to create a dictionary when all parent coordinates are uniparous, since the children under this condition can be upsampled without error. One LUT is, thus, created for each condition, depending on the parent coordinates. The steps required for the construction of the LUTs are shown on the left-hand side of Fig. 6.

From V_d and V_{d^2} we can "learn on-the-fly" how to super-resolve the geometry using the neighborhood configuration of each voxel. Then, we use this knowledge to super-resolve from V_d back to its original resolution V_{sr}. As depicted on the right-hand side of Fig. 6, it suffices to get the neighborhood for each voxel in V_d, stratify those voxels according to s, and finally, using the previously acquired LUTs, estimate child occupancy. Thus, the set of super-resolved children from $\mathbf{v}_d(k)$ is

$$\mathcal{V}_{sr}(k) = \mathcal{V}_u(k \mid \sigma(\mathbf{v}_{d,j}(k))), \tag{10}$$

where $\sigma(\mathbf{v}_{d,j}(k)) = \text{LUT}_j\left(\varphi_M(\mathbf{v}_{d,j}(k))\right)$ indicates that a given neighborhood configuration determines which of the possible children of $\mathbf{v}_d(k)$ must be set as occupied, and $0 \leq j \leq 7$ indicates the parenthood stratification (Fig. 5). The super-resolved geometry is the union of all $\mathcal{V}_{sr}(k)$,

$$V_{sr} = \bigcup_{k=1}^{K} \mathcal{V}_{sr}(k). \tag{11}$$

TABLE I: Illustration of the created LUT.

m	φ_M	$\bar{\sigma}$
0	0000000...0000	1111 1111
1	0000000...0001	0000 1100
2	0000000...0010	0110 1001
⋮	⋮	⋮
$2^{M^3-1}-1$	1111111...1111	1111 1111

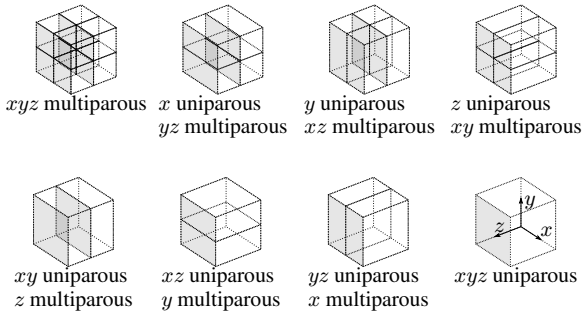


Fig. 5: The 8 types of parenthood stratification for $1 < s \leq 2$.

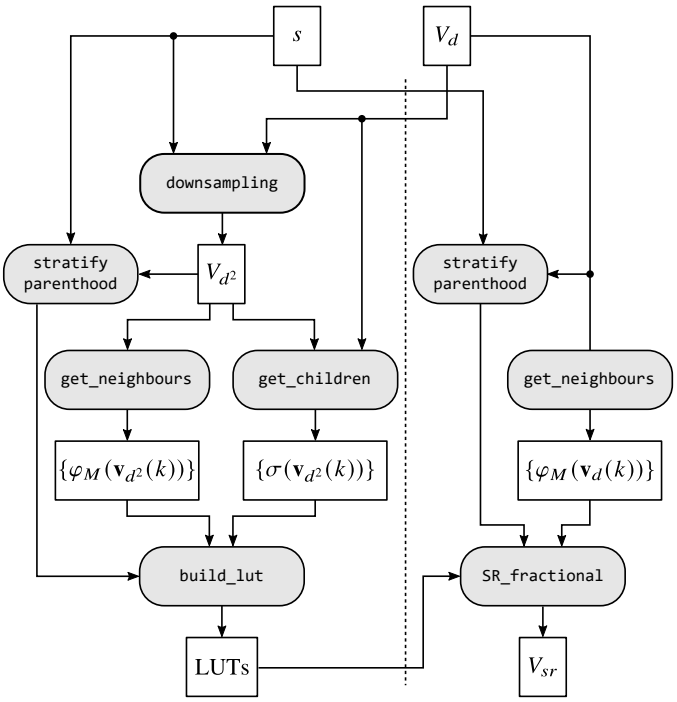


Fig. 6: The architecture of the proposed method. The steps required for the LUTs' construction are shown on the left side. On the right, we show how those LUTs are used to super-resolve the input LR geometry V_d.

B. Implementation Issues

In the current implemented version, some constraints had to be defined and additional steps were introduced to improve the super-resolved PC.

For a symmetric neighborhood around a voxel, the neighborhood size, M, must be an odd number. As M increases by a single step, from 3 to 5, the possible neighborhood configurations go from 2^{26} to 2^{124}. This makes impractical to use large values of M, not only because it takes more computational effort to find a bigger neighborhood, but also because building an effective dictionary with so many entries from a single PC is not possible. The entries become overly specific and the output geometry would be approximately equal to the NNI upsampling. For this reason, we decided to fix $M = 3$, such that whenever $\varphi(k)$ is mentioned it is implicit that a neighborhood size 3, $\varphi_3(k)$, is considered.

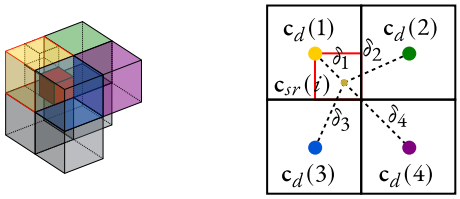
Memory size limitations constrain the dictionary size and the scale factor s. Moreover, as s increases, the number of meaningful entries in the dictionary decreases, since there is not much information in the lower levels of the geometry. In other words, the preservation of self-similarities is diminished with the increase of s. Thus, we decided to constrain the values of s, $\{s \in \mathbb{Q} \mid 1 < s \leq 2\}$. Inside this interval, we can profit from partial downsampling and super-resolve a full octree level. If $s > 2$ is required, the proposed SR method can still be used, by performing $t = \lceil \log_2(s) \rceil$ nested SRs with a new scale factor $s' \approx \sqrt[t]{s}$, where the approximation sign is needed since s' must be a fractional number.

As a means of data augmentation, we applied incremental translations to the input frame to increase the population of the LUT. Since $1 < s \leq 2$, coordinate shifts of ± 1 to each are sufficient to change the result of Eq. (1), and the parenthood stratification. Other transformations, such as rotation or scaling, were left to future work.

C. Color interpolation

In order to find the texture for the upsampled geometry, the usual approach is to first put both LR and SR geometries in the same scale, then interpolate the colors for the SR voxels using a distance-based weighted average of the LR voxels colors. In the case where no direct correspondence is found between the SR and the LR voxels, the average, weighted by the inverse of the distances δ^{-1}, is taken over a $3 \times 3 \times 3$ neighborhood.

A slight improvement to the previous interpolation method can be achieved borrowing the color prediction used in G-PCC in a new function within the present SR context. In the transform domain prediction of RAHT [30], the estimated color for each occupied child node is the average of the parent node's color, with the colors of the uncle nodes that share an edge with that child node, as illustrated in Fig. 7. The average is weighted by the inverse distance between each parent node and the current child node being estimated. We refer to this method as the weighted average of adjacent neighbors (WAAN). A variable weight, ζ dependent of s, was introduced in the WAAN to take into account that the parent color should be more important as the scale factor decreases.



(a) (b)

Fig. 7: Illustration of the neighbors used in the WAAN calculation. (a) For the highlighted child node, only uncle nodes sharing a face with it are considered. (b) The distances δ_ℓ, from the child node to its uncles.

Thus, $\mathcal{C}_{sr}(k) = \{\mathbf{c}_{sr}(i)\}$ is the set containing the respective colors of $\mathcal{V}_{sr}(k)$ of a given parent $\mathbf{v}_d(k)$, such that:

$$\mathbf{c}_{sr}(i) = \frac{\mathbf{c}_d(k) + \zeta \sum_\ell \delta_\ell^{-1} \mathbf{c}_d(\ell)}{1 + \zeta \sum_\ell \delta_\ell^{-1}}, \tag{12}$$

where ℓ is the index of the occupied neighbors sharing a face with $\mathbf{v}_{sr}(i)$ (Fig. 7(b)). ζ was empirically found as $\zeta = \delta_1 s/8$. The super-resolved colors are, then,

$$C_{sr} = \bigcup_{k=1}^{K} \mathcal{C}_{sr}(k). \tag{13}$$

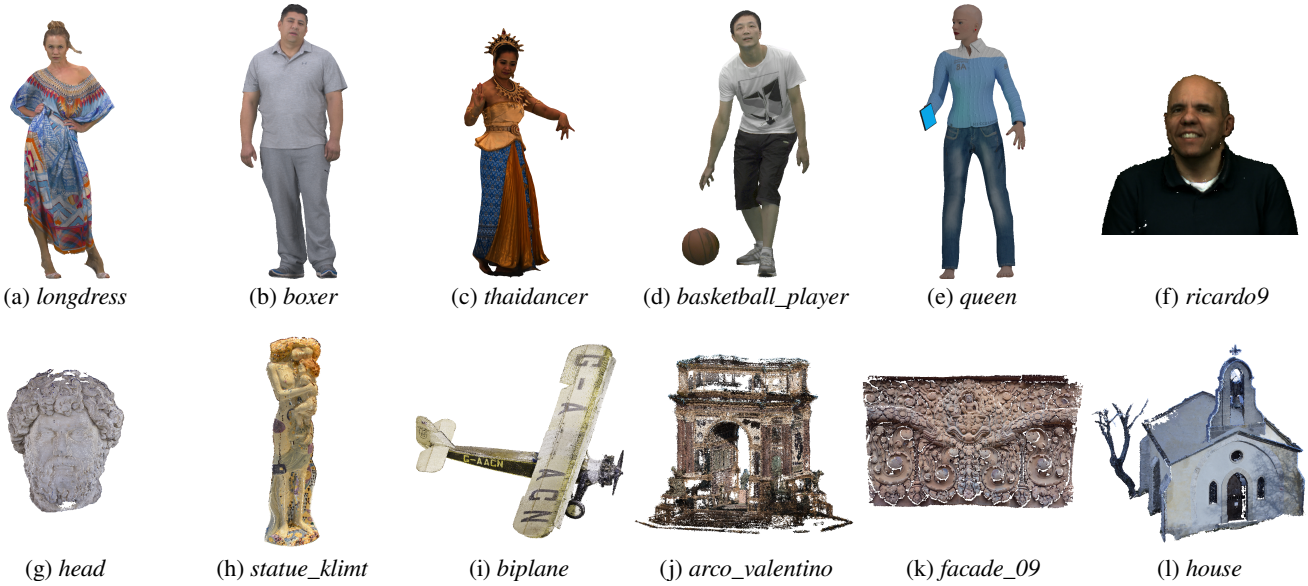
IV. Performance Assessment and Analysis

A. Datasets and test conditions

We focused on PCs of static objects and scenes from the MPEG's G-PCC common test conditions (CTC) [10] and set a point cap of 4 million voxels, due to the current implementation's memory restrictions. Those PCs are originally voxelized, avoiding biases introduced in the voxelization process. In order to reduce the number of comparisons and to obtain a more representative result, we clustered PCs sharing the same source, voxel depth, and density into groups from (a) to (l), whenever possible. Table II summarizes information about the chosen contents, where "Vox." indicates if the PC required a pre-processing voxelization step, and ρ_φ is a density measure taken by the average neighborhood occupancy-rate of adjacent voxels to an occupied voxel. Figure 8 depicts representatives viewpoints from the PCs.

Some PCs representing objects were voxelized from meshes. PCs with more than 4 million occupied voxels were downsampled to meet the point cap. Other PCs were extremely sparse, and downsampling them with $s = 2$ would decimate less than 1% of the original points, making SR unjustified. In these cases, we reduced the bit depth to increase density.

The density measure ρ_φ is used as a predictor of the proposed method's performance, since we rely on similarities at different scales. Those are somewhat maintained for dense PCs, but not so much for sparse ones. We empirically found out that when ρ_φ is beyond 0.3 or so, the PC have watertight projections, i.e., there is a one-to-one relationship between rendered pixels and voxels without holes. We consider PCs with watertight projections to be somewhat dense. If, however,



(a) *longdress* (b) *boxer* (c) *thaidancer* (d) *basketball_player* (e) *queen* (f) *ricardo9*

(g) *head* (h) *statue_klimt* (i) *biplane* (j) *arco_valentino* (k) *facade_09* (l) *house*

Fig. 8: Representative viewpoints of some of the human figures from (a) to (f), and of the objects from (g) to (l).

TABLE II: Summary of tested PCs.

Point clouds	Vox.	Depth	# voxels	ρ_φ
(a) Group: *8i_vox10* [7]				
longdress_vox10_1300	✗	10-bit	857,966	0.429
loot_vox10 _1200	✗	10-bit	805,285	0.428
redandblack_vox10_1550	✗	10-bit	757,691	0.433
soldier_vox10_0690	✗	10-bit	1,089,091	0.432
(b) Group: *8i_vox12* [8]				
boxer_viewdep_vox12	✗	12-bit	3,493,085	0.031
longdress_viewdep_vox12	✗	12-bit	3,096,122	0.027
loot_viewdep_vox12	✗	12-bit	3,017,285	0.029
redandblack_viewdep_vox12	✗	12-bit	2,770,567	0.025
(c) *Thaidancer_viewdep_vox12* [8]	✗	12-bit	3,130,215	0.332
(d) Group: *owlii* [31]				
basketball_player_vox11_00000200	✗	11-bit	2,925,514	0.452
dancer_vox11_00000001	✗	11-bit	2,592,758	0.445
(e) *queen_frame_0200*†	✗	10-bit	1,000,993	0.524
(f) Group: *MVUB* [9]				
andrew9_0000	✗	9-bit	279,664	0.547
david9_0000	✗	9-bit	330,797	0.542
phil9_0000	✗	9-bit	370,798	0.543
ricardo9_0000	✗	9-bit	214,656	0.550
sarah9_0000	✗	9-bit	302,437	0.538
(g) *Head_00039_vox12*†	✓	9-bit	938,112	0.532
(h) *1x1_Biplane_Combined_000*‡	✓	10-bit	1,181,016	0.567
(i) *Statue_Klimt_vox12*†	✓	10-bit	483,068	0.209
(j) *Arco_Valentino_Dense_vox12*†	✗	12-bit	1,481,746	0.025
(k) *Facade_00009_vox20*†	✓	11-bit	1,560,786	0.165
(l) *House_without_roof_00057_vox12*†	✓	11-bit	3,638,139	0.247

† https://mpegfs.int-evry.fr/mpegcontent/
‡ https://jpeg.org/plenodb/

ρ_φ is below 0.3, we considered the PC to be sparser, as there will likely be holes in its projections. If $\rho_\varphi = 0$ for a given neighborhood size M, but $\rho_\varphi > 0$ for a neighborhood size $M' > M$, then it is possible to reduce an initially assumed sparse PC into a dense one at a lower resolution.

B. Self-similarities at different scales

The proposed SR method assumes that the geometry of a PC is approximately self-similar at different scales. The under-complete dictionaries are constructed using a coarser geometry and applied to recreate a finer geometry. One way to verify the preservation of similarities at different scales is to create a dictionary using the original HR PC and compare it with the one created from the LR version. Using *redandblack_vox10_1550* and $s = 1.5$, we found out that on average 48% of the neighborhood configurations present at both dictionaries are identical, and 86% differ at most by 1 position. When comparing the outputs of identical neighborhood entries, there is a 63% of identical children occupancy, and 82% of the children also differ at most by 1 position. From these numbers, we can see that the assumed self-similarities are approximately preserved at different scales.

C. Evaluation framework

Ten LR versions of each input PC were created by varying the scale factor s in the interval $1.1 \leq s \leq 2$. To assess the quality of the proposed method (LUT), we applied the NNI upsampling to each LR version to serve as the baseline. Additionally, in order to mitigate the aliasing effects from the NNI approach, we also considered smoothing it using the Laplacian smoothing technique (labeled as NNI+LS). Comparisons with other methods were not carried out because they were developed with set downsampling in mind and would require adaptations to work with grid downsampling LR PCs. Also, most of the deep learning methods would require a lot of retraining to cope with the different scale factors and with the real-world voxelized PCs of our test set. The other methods from Garcia *et al.* [26] also cannot be used because they do not allow for fractional scale factors.

Three point-based and three projection-based metrics were chosen for the assessment. For the point-based metrics we used: point-to-point (D1), point-to-plane (D2) and luma end-to-end. Point-based metrics are computed symmetrically, first using the original PC as reference, then using the distorted version as reference. The final value is the maximum error

observed between the two measurements. The D1 metric [32] is calculated as the average squared distance between each point in the first PC and its nearest neighbor in the second one. The D2 metric [33] is similar to the D1 metric, except that the distances are projected to the normal direction before being averaged, imposing larger penalties on errors that move further away from the local plane surface. D1 and D2 MSE are converted to PSNR using the length of the diagonal of the bounding cube containing the PC as the normalization factor. For the texture assessment, the colors of each point in the first PC are compared to the ones from their nearest neighbors in the second PC. RGB colors are converted to YUV_{709} for the Y-PSNR calculation. These metrics were selected as they are widely used in MPEG's core experiment evaluations.

For the projection-based metrics [34], [35] we used PSNR, SSIM [36] and VIFp [37]. They are referred with a preceding "P", as in projected PSNR (PPSNR). To get the projections, voxels were rendered as cubes and point size was set equal to 1. Although this choice of rendering may generate holes in sparse PCs, different choices could add rendering distortions to SR intrinsic artifacts. Six projection views were used (the six faces of the cube containing the PC). In this way, each visible voxel side is projected into a single pixel, so that in a depth-10 PC there are six 1024×1024 pixel projections. In order to decrease the effect of the background in the metrics, its color was set to a mid-gray value, and we only considered the rectangular region formed by the union of the foregrounds of the reference and the distorted projections, as suggested by Alexiou *et al.* [38].

D. Results

Table III shows the average gain of the NNI+LS and the LUT approaches when compared to the NNI upsampling. As we can see from it, the proposed method is superior than the baseline or its smoothed counterpart, for every content in almost every metric. Note that there are odd cases for resampling such as totally occluded voxels with a different colour than its neighbours (occurs in *queen*), cropped edges (in *MUVB*), and noisy geometry scans in almost every scanned PC. Thus, we expect better results from "well-behaved" PCs.

Geometry distortion comparisons are presented in Figs. 9 and 10, for some of the PCs, to illustrate the behavior over different values of s. From these plots, we can see the consistent gain of the proposed method. Also it is possible to observe that the proposed method is able to generate better results when the input PCs are somewhat dense (*Thaidancer* and *Biplane*). This is because, in denser PCs, the geometry changes are smaller than in sparser ones with the increasing of s, which is a desired property for the creation of the dictionaries. Also, in general, we observe better results for the proposed method for lower values of s. This is expected, as s increases more guesses must be made (less uniparous parents), less information is available to create the LUTs (coarser geometry).

We can see that the proposed method outperforms the others for all PCs but *Biplane*, in the Y-PSNR plots of Fig. 11. As it is possible to see in Fig. 8(i), *Biplane* has a significant amount of

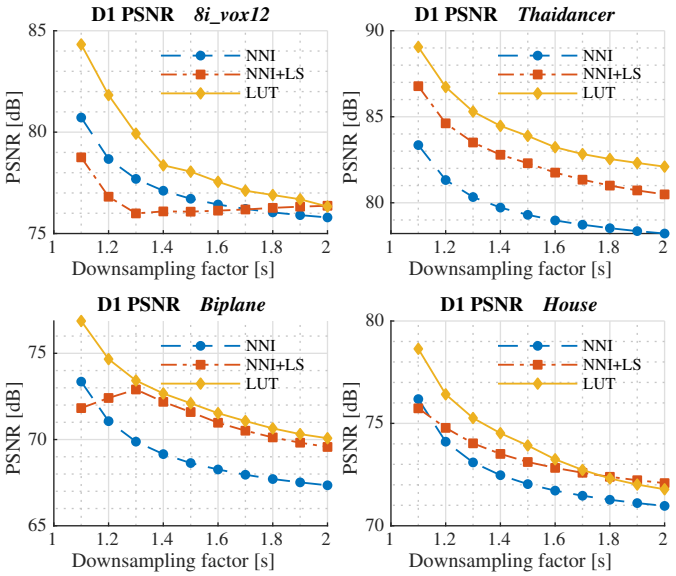


Fig. 9: D1 metric for PCs (b), (c), (h) and (l). NNI is the baseline upsampling, NNI+LS is the latter followed by Laplacian smoothing, and LUT is the proposed method.

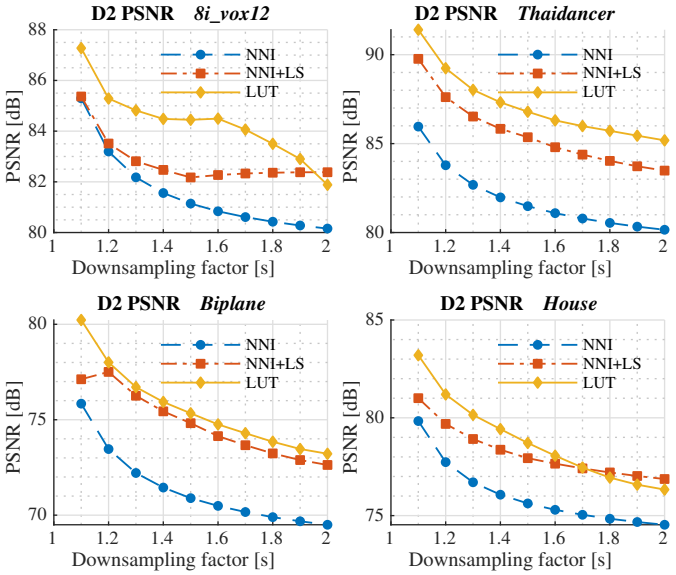


Fig. 10: D2 metric for PCs (b), (c), (h) and (l).

noise in its texture, which is hard to replicate considering that both texture interpolation solutions are based on smoothing neighboring colors.

Plots with PPSNR distortion metric are shown in Fig. 12. Holes caused by missing occupied children affect more this metric than the others, which occurs more frequently for the sparser clouds. The low values of PPSNR observed, particularly for sparse PCs, occur because of the way we chose to render the projections. The majority of errors made by NNI upsampling-related methods are usually of excess nature, i.e., they add more points than necessary in their attempts to recreate the original PC. In sparse PCs, those errors are more apparent, reducing the absolute level of the

TABLE III: Average gain over the NNI.

Point clouds		D1 PSNR [dB]		D2 PSNR [dB]		Y-PSNR [dB]		PPSNR [dB]		PSSIM		PVIFP	
		NNI+LS	LUT	NNI+LS	LUT	NNI+LS	LUT	NNI+LS	LUT	NNI+LS	LUT	NNI+LS	LUT
(a)	*8i_vox10*	3.27	**5.47**	4.29	**5.91**	0.17	**1.95**	2.06	**3.80**	0.02	**0.03**	**0.06**	0.04
(b)	*8i_vox12*	-0.63	**1.58**	1.24	**2.75**	-2.79	**2.87**	-1.60	**0.27**	-0.13	**-0.02**	-0.16	**0.02**
(c)	*Thaidancer*	2.85	**4.57**	3.67	**5.26**	0.14	**2.92**	2.42	**4.28**	0.01	**0.02**	**0.06**	0.05
(d)	*owlii*	3.39	**6.47**	4.41	**6.77**	0.04	**1.61**	2.37	**3.97**	0.01	**0.02**	0.05	0.05
(e)	*queen*	3.18	**6.24**	4.66	**7.12**	-0.99	**0.76**	1.27	**3.84**	0.01	**0.02**	0.05	**0.08**
(f)	*MVUB*	2.72	**3.98**	4.10	**4.80**	-0.37	**1.31**	0.12	**1.72**	0.01	**0.02**	0.02	**0.03**
(g)	*Head*	2.86	**4.32**	4.23	**5.42**	-0.69	**-0.12**	0.01	**1.43**	0.01	**0.04**	-0.03	-0.03
(h)	*Biplane*	2.10	**3.25**	3.41	**4.22**	-0.82	**-0.28**	-0.54	**0.86**	-0.01	**0.01**	-0.05	-0.05
(i)	*Statue_Klimt*	**0.88**	0.75	**1.75**	1.16	-0.94	**1.29**	-0.55	**0.30**	-0.02	**0.01**	-0.06	**-0.03**
(j)	*Arco_Valentino*	0.00	**0.93**	0.03	**1.02**	-3.59	**0.01**	-0.09	**1.02**	-0.01	**-0.03**	**-0.08**	-0.30
(k)	*Facade_00009*	0.61	**1.43**	1.88	**2.29**	-1.61	**1.58**	-0.93	**0.04**	-0.05	**-0.01**	-0.12	**-0.06**
(l)	*House*	0.89	**1.64**	2.18	**2.77**	-1.16	**0.74**	-0.79	**-0.04**	-0.02	**-0.01**	-0.09	**-0.04**

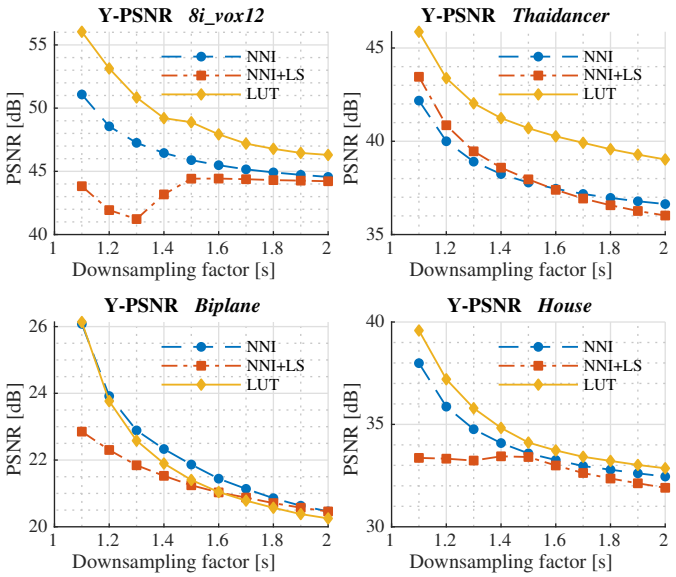


Fig. 11: Y-PSNR metric for PCs (b), (c), (h) and (l).

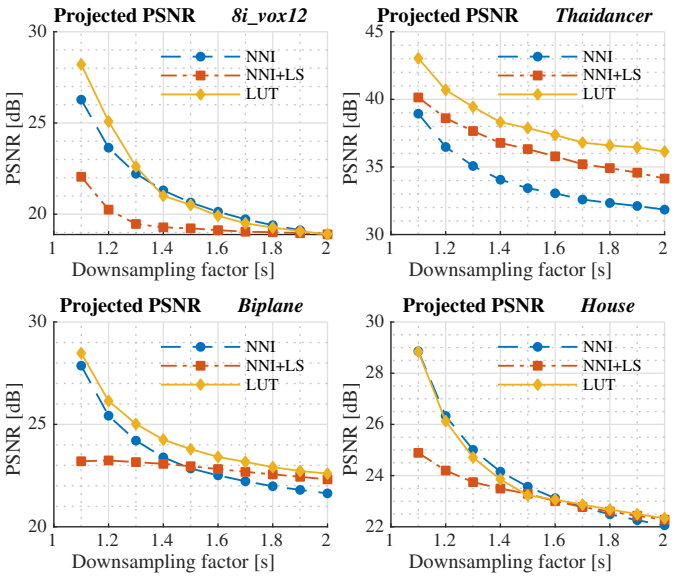


Fig. 12: Projected PSNR metric for PCs (b), (c), (h) and (l).

PPSNR. If a different rendering approach was used, probably different values were to be observed. Still, the proposed method achieves better results in almost all cases.

Viewpoint projections for *redandblack_viewdep_vox12* and *Biplane* are shown in Fig. 13 for visual comparison.

V. Conclusions

In this work, we presented a method for super-resolving voxelized PCs at fractional scales, in which self-similarities at lower scales are used to define which of the possible children should be occupied. Extensive results show that the proposed method yields lower distortion results when compared to upsampling by NNI, and to the NNI followed by smoothing. Although only static point-clouds were used, the proposed method can be transferred to dynamic ones, with probably even better results [26], as more frames are available to create the LUT. Future works may focus on robustness against outlier regions in the PC. We also plan to improve the super-resolution of sparse PCs, and of texture attributes.

References

[1] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai, "An overview of ongoing point cloud compression standardization activities: video-based (V-PCC) and geometry-based (G-PCC)," *APSIPA Transactions on Signal and Information Processing*, vol. 9, 2020.

[2] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, 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, 2019.

[3] 3DG, "V-PCC Codec Description," ISO/IEC JTC 1/SC 29/WG 11, Geneva, CH, Approved WG 11 doc. N18892, Oct. 2019.

[4] ——, "G-PCC codec description v5," ISO/IEC MPEG JTC 1/SC 29/WG 11, Geneva, CH, Approved WG 11 doc. N18891, Oct. 2019.

[5] D. Meagher, "Geometric modeling using octree encoding," *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129–147, Jun 1982.

[6] 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.

[7] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i Voxelized Full Bodies, version 2 – A Voxelized Point Cloud Dataset," ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG), Geneva, input document m40059/M74006, January 2017.

[8] M. Krivokuća, P. A. Chou, and P. Savill, "8i Voxelized Surface

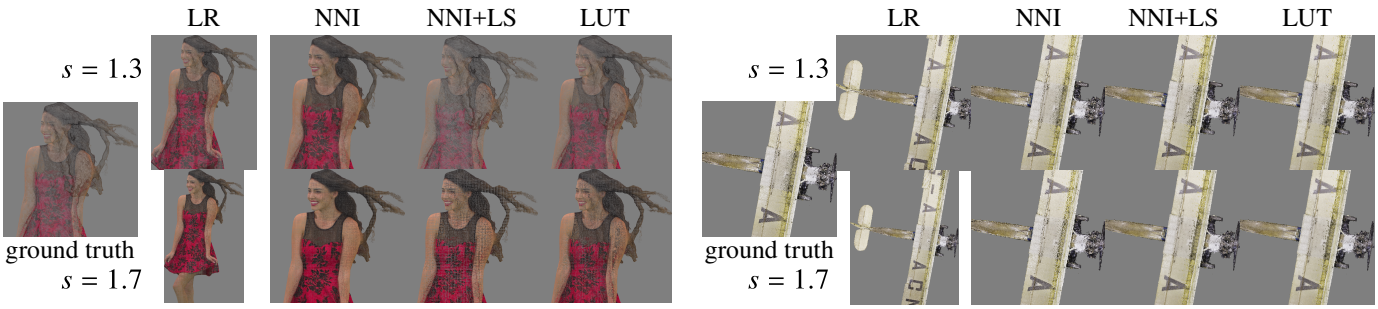


Fig. 13: Projections for *redandblack_viewdep_vox12* and *Biplane* for different values of s.

Light Field (8iVSLF) Dataset," ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG), Ljubljana, input doc. m42914, July 2018.

[9] C. Loop, Q. Cai, S. Escolano, and P. Chou, "Microsoft voxelized upper bodies - a voxelized point cloud dataset," ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG), input doc. m38673/M72012, May 2016.

[10] 3DG, "Common test conditions for point cloud compression," ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG), Gothenburg, SE, Approved WG 11 doc. N18883, July 2019.

[11] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 1, pp. 3–15, 2003.

[12] G. Guennebaud and M. Gross, "Algebraic point set surfaces," *ACM Transactions on Graphics*, vol. 26, no. 3, p. 23, jul 2007.

[13] A. C. Öztireli, G. Guennebaud, and M. Gross, "Feature preserving point set surfaces based on non-linear kernel regression," *Computer Graphics Forum*, vol. 28, no. 2, pp. 493–501, apr 2009.

[14] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. R. Zhang, "Edge-aware point set resampling," *ACM Trans. Graph.*, vol. 32, no. 1, Feb. 2013.

[15] A. Hamdi-Cherif, J. Digne, and R. Chaine, "Super-resolution of point set surfaces using local similarities," *Computer Graphics Forum*, vol. 37, no. 1, pp. 60–70, jun 2017.

[16] C. Dinesh, G. Cheung, and I. V. Bajić, "3D point cloud super-resolution via graph total variation on surface normals," in *IEEE International Conference on Image Processing (ICIP)*. IEEE, sep 2019.

[17] C. Dinesh, G. Cheung, and I. V. Bajić, "Super-resolution of 3D color point clouds via fast graph total variation," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 1983–1987.

[18] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-net: Point cloud upsampling network," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2018.

[19] ——, "EC-net: An edge-aware point set consolidation network," in *Computer Vision – ECCV 2018*. Springer International Publishing, 2018, pp. 398–414.

[20] W. Yifan, S. Wu, H. Huang, D. Cohen-Or, and O. Sorkine-Hornung, "Patch-based progressive 3d point set upsampling," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2019.

[21] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-GAN: a point cloud upsampling adversarial network," in *IEEE International Conference on Computer Vision (ICCV)*, Oct. 2019.

[22] H. Wu, J. Zhang, and K. Huang, "Point Cloud Super Resolution with Adversarial Residual Graph Networks," *BMVC 2020*, 2020.

[23] G. Qian, A. Abualshour, G. Li, A. Thabet, and B. Ghanem, "PU-GCN: Point cloud upsampling using graph convolutional networks," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[24] Y. Qian, J. Hou, S. Kwong, and Y. He, "PUGeo-net: A geometry-centric network for 3d point cloud upsampling," in *Computer Vision – ECCV 2020*. Springer International Publishing, 2020, pp. 752–769.

[25] S. Ye, D. Chen, S. Han, Z. Wan, and J. Liao, "Meta-PU: An arbitrary-scale upsampling network for point cloud," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2021.

[26] D. C. Garcia, T. A. Fonseca, R. U. Ferreira, and R. L. de Queiroz, "Geometry coding for dynamic voxelized point clouds using octrees and multiple contexts," *IEEE Transactions on Image Processing*, vol. 29, pp. 313–322, 2019.

[27] D. C. Garcia, T. A. Fonseca, and R. L. de Queiroz, "Example-based super-resolution for point-cloud video," in *IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 2959–2963.

[28] M. Corsini, P. Cignoni, and R. Scopigno, "Efficient and flexible sampling with blue noise properties of triangular meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 6, pp. 914–924, 2012.

[29] A. B. Yutaka and E. B. Y. Ohtake, "A comparison of mesh smoothing methods," in *In Proceedings of the Israel-Korea BiNational Conference on Geometric Modeling and Computer Graphics*, 2003, pp. 83–87.

[30] S. Lasserre and D. Flynn, "[G-PCC][new proposal] On an improvement of RAHT to exploit attribute correlation," ISO/IEC MPEG JTC1/SC29/WG11, Geneva, CH, Tech. Rep. m47378, Mar. 2019.

[31] Y. Xu, Y. Lu, and Z. Wen, "Owlii Dynamic Human Textured Mesh Sequence Dataset," ISO/IEC MPEG JTC1/SC29/WG11, Macau, China, Tech. Rep. m41658, Oct. 2017.

[32] D. Girardeau-Montaut, M. Roux, R. Marc, and G. Thibault, "Change detection on points cloud data acquired with a ground laser scanner," in *Proceedings of the ISPRS Workshop Laser scanning 2005*, G. Vosselman and C. Brenner, Eds., vol. XXXVI-3/W19. Enschede, the Netherlands: International Society for Photogrammetry and Remote Sensing, Sep. 2005, pp. 30–35.

[33] 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.

[34] E. M. Torlig, E. Alexiou, T. A. Fonseca, R. L. de Queiroz, and T. Ebrahimi, "A novel methodology for quality assessment of voxelized point clouds," in *Applications of Digital Image Processing XLI*, A. G. Tescher, Ed., vol. 10752, International Society for Optics and Photonics. SPIE, 2018.

[35] R. L. de Queiroz and P. A. Chou, "Motion-compensated compression of dynamic voxelized point clouds," *IEEE Transactions on Image Processing*, vol. 26, no. 8, pp. 3886–3895, Aug 2017.

[36] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. S. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, apr 2004.

[37] H. R. Sheikh and A. C. Bovik, "Image information and visual quality," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 430–444, Feb 2006.

[38] E. Alexiou, I. Viola, T. M. Borges, T. A. Fonseca, R. L. de Queiroz, and T. Ebrahimi, "A comprehensive study of the rate-distortion performance in MPEG point cloud compression," *APSIPA Transactions on Signal and Information Processing*, vol. 8, p. e27, 2019.