Set Partitioning in Hierarchical Trees for Point Cloud Attribute Compression

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

22-07-2021 / 26-07-2021

CITATION

de Queiroz, Ricardo; Souto, Andre; Figueiredo, Victor; Chou, Philip (2021): Set Partitioning in Hierarchical Trees for Point Cloud Attribute Compression. TechRxiv. Preprint.
<https://doi.org/10.36227/techrxiv.15032901.v1>

DOI

[10.36227/techrxiv.15032901.v1](https://doi.org/10.36227/techrxiv.15032901.v1)

Set Partitioning in Hierarchical Trees for Point Cloud Attribute Compression

André L. Souto, Victor F. Figueiredo, *Student Member, IEEE*, Philip A. Chou, *Fellow, IEEE*, and Ricardo L. de Queiroz, *Fellow, IEEE*

Abstract—We propose an embedded attribute encoding method for point clouds based on set partitioning in hierarchical trees (SPIHT) [1]. The encoder is used with the region-adaptive hierarchical transform [2] which has been a popular transform for point cloud coding, even included in the standard geometry-based point cloud coder (G-PCC) [3],[4]. The result is an encoder that is efficient, scalable, and embedded. That is, higher compression is achieved by trimming the full bit-stream. G-PCC's RAHT coefficient prediction prevents the straightforward incorporation of SPIHT into G-PCC. However, our results over other RAHT-based coders are promising, improving over the original, non-predictive RAHT encoder, while providing the key functionality of being embedded.

Index Terms—Bit-Plane Encoding, Point Cloud, RAHT, SPIHT.

I. INTRODUCTION

PPOINT clouds, as a 3D data representation, can be used in several applications, including virtual/augmented reality, immersive telepresence, autonomous driving, and cultural heritage archival [3], [4]. A point cloud (PC) is described by its geometry, the three-dimensional coordinates (x, y, z) of each point in the set, and its attributes, like color and reflectance. A PC usually comprises a large number of points and, because of that, the Moving Pictures Expert Group (MPEG) developed standard algorithms for PC compression [3], [4]. MPEG's Geometry-based Point Cloud Compression (G-PCC) [5] standard was designed for compressing static PCs and compresses geometry and attributes separately. Geometry compression is outside the scope of this work and, in G-PCC, one of the attribute coding methods is based on the region-adaptive hierarchical transform (RAHT) [5], [2].

RAHT is an orthogonal transform that resembles a geometry-adaptive version of the Haar transform [2]. Transform coefficient encoding typically uses arithmetic coding (AC). Since RAHT was developed to be used in real-time telepresence systems, a simpler adaptive run-length Golomb-Rice encoding (RLGR) [6] was originally tested with RAHT. It led to a noticeably inferior performance, compared to the AC coder, but enabling a much simpler implementation. In

[7], it was realized that reordering RAHT coefficients could improve the performance of the RLGR-based coder to rival the AC-based one. G-PCC provides a RAHT encoder with a number of improvements in quantization, arithmetic encoding and coefficient prediction.

In this work, we explore the use of set partitioning in hierarchical trees (SPIHT) [1] in the context of PC attribute compression. SPIHT is an alternative to encoding RAHT coefficients with either AC or RLGR. SPIHT, unlike either alternative, is an embedded coder, where the bit-stream is build by progressively increasing the quality and resolution of the PC. It is inherently scalable and allows for progressive transmission. There is a single bit-stream, and the amount of compression achieved is determined by how much of the bit-stream is kept.

II. SET PARTITIONING IN HIERARCHICAL TREES

The SPIHT algorithm [1] is a bit-plane coder that uses a hierarchical set partition to code the bit-plane information. The core of the algorithm is very simple. The encoder is given a step-size Δ , a collection of real-valued coefficients $\{c_t : t \in T\}$ indexed by a set T , and a hierarchical set partition of T . The encoder quantizes each coefficient c_t into an integer $k_t = \text{round}(c_t/\Delta)$, and then transmits the bit-planes of $|k_t|$ along with $\text{sign}(k_t)$ if k_t is non-zero. The bit-planes are transmitted from the most significant bit-plane $b = b_{\max} = \lfloor \log_2(\max_{t \in T} |k_t|) \rfloor$ to the least significant bit-plane $b = 0$. A coefficient c_t is said to be significant in bit-plane b if $\lfloor \log_2(|k_t|) \rfloor \geq b$. Otherwise it is insignificant in bit-plane b .

A hierarchical set partition of T is a partition of T into sets, and a further partition of those sets into sets until there is only a single set of sets T_0 remaining. The hierarchical set partition forms a tree structure. At the leaves of the tree are singleton sets, that is, sets $\{t\}$ containing only single elements $t \in T$. At the internal nodes of the tree are compound sets, that is, sets containing one or more other sets. A set in the hierarchical set partition is said to be significant (in bit-plane b) if any of its member sets are significant, or insignificant if all of its members sets are insignificant. A singleton set $\{t\}$ is significant if and only if coefficient c_t is significant.

SPIHT can use a single bit to declare that an entire set T' in the hierarchical set partition is insignificant in bit-plane b , needing no further attention in that bit-plane. However, if T' turns out to be significant, then SPIHT must open up the set to consider which of its elements are actually significant.

This study was financed in part by Coordenação de Aperfeiçoamento de Pessoal Superior - Brasil (CAPES) - Finance Code 001 and by CNPq under grant 301647/2018-6.

V. F. Figueiredo and A. L. Souto are with the Electrical Engineering Department, University of Brasília, Brasília, Brazil (e-mails: fabre@ieee.org and andre@image.unb.br).

P. A. Chou is with Google, Seattle, WA, USA, (e-mail: pachou@ieee.org).

R. L. de Queiroz is with the Computer Science Department, University of Brasília, Brasília, Brazil (e-mail: queiroz@ieee.org).

If T' is a singleton set $\{t\}$, then c_t is newly significant. Otherwise, if T' is a compound set $\{T'_1, \dots, T'_n\}$, then at least one of the sets $T'_1, \dots, T'_n$ must be significant, and SPIHT must recursively use a single bit to signal whether or not each member set is significant. A version of the SPIHT *encoding* algorithm, analogous to the original algorithm in [1], is shown in Algorithm 1. The *decoding* algorithm is similar, replacing encoding each bit by decoding each bit. In the algorithm, the following quantities are defined:

- $\mathcal{O}(t)$: the nodes that are direct descendants of node t ,
- $\mathcal{D}(t)$: the set of nodes that are descendants of node t ,
- $\mathcal{L}(t)$: set defined by $\mathcal{L}(t) = \mathcal{D}(t) - \mathcal{O}(t)$,
- LIS : List of Insignificant Sets, and
- LSN : List of Significant Nodes.

With this, two different types of compound sets are defined:

- *Type A*: $T' = \mathcal{D}(t) = \{\{t_c\} : t_c \in \mathcal{O}(t)\} \cup \{\mathcal{L}(t)\}$, and
- *Type B*: $T' = \mathcal{L}(t) = \{\mathcal{D}(t_c) : t_c \in \mathcal{O}(t)\}$.

The algorithm operates as follows. The encoder initializes a List of Significant Nodes (LSN) as an empty set, a List of Insignificant Sets (LIS) as the list of sets $\{t_0\}$ and $\mathcal{D}(t_0)$ in the top level $T_0 = \{\{t_0\}, \mathcal{D}(t_0)\}$ of the hierarchical set partition, and b as the most significant bit-plane $b = b_{\max}$. Then, for each bit-plane $b = b_{\max}, \dots, 0$, the algorithm goes through a refinement pass and a sorting pass.

In the refinement pass, the indices t of any coefficients that became significant in a previous bit-plane are found in the LSN. SPIHT refines such coefficients by encoding the b -th bit of $|k_t|$ for all elements t in the LSN.

In the sorting pass, the sets T' found in the LIS are processed sequentially. When a set T' in the LIS is processed, a bit $S_b(T')$ is encoded to signal whether T' is significant or not in bit-plane b . If T' is insignificant, then no action is taken and it is revisited in subsequent bit-planes. However, if T' is significant, then it is opened and its members are examined. If T' is a singleton $\{t\}$, then the sign of k_t is encoded, T' is removed from LIS, and t is added to LSN. Otherwise, T' is a set of sets, either $T' = \mathcal{D}(t) = \{\{t_c\} : t_c \in \mathcal{O}(t)\} \cup \{\mathcal{L}(t)\}$ (type A) or $T' = \mathcal{L}(t) = \{\mathcal{D}(t_c) : t_c \in \mathcal{O}(t)\}$ (type B). If T' is type A, then every singleton set $\{t_c\}$ in $\mathcal{D}(t)$ is processed immediately as a singleton set, and $\mathcal{L}(t)$ (if it is not empty) is placed at the end of LIS for later processing in the same level and same bit-plane. If T' is type B, then every set $\mathcal{D}(t_c)$ in $\mathcal{L}(t)$ is placed at the end of LIS for later processing in the next level and same bit-plane. Then T' is deleted from the LIS. The process is repeated until the end of LIS is reached.

When the refinement and sorting passes are done for bit-plane b , the bitplane is decreased and the process is repeated. The result is a bit-stream that can be truncated at any point to recover the bit-planes of coefficients processed up to that point.

III. SPIHT FOR RAHT

It is clear that SPIHT can use any hierarchical set partition of T . Certain hierarchical set partitions result in lower numbers of bits encoded than others. The best hierarchical set partitions are those for which the member sets of a set have highly correlated significance. Specifically, if one member of a set

Algorithm 1 - SPIHT encoder

Input: $\Delta, \{c_t : t \in T\}, T_0$

```

1: /* Initialization */
2: for  $t \in T$  do  $k_t = \text{round}(c_t/\Delta)$ 
3:  $LSN \leftarrow$  empty list ▷ Significant Nodes
4:  $LIS \leftarrow$  list of member sets in  $T_0$  ▷ Insignificant Sets
5:  $b \leftarrow b_{\max} = \lfloor \log_2(\max_{t \in T} |k_t|) \rfloor$  ▷ index of highest bitplane
6: /* Refinement Pass */
7: for each element  $t \in LSN$  do
8:   Encode bit  $b$  of  $k_t$ 
9: end for
10: /* Sorting Pass */
11: for each set  $T'$  in  $LIS$  do
12:   Encode  $S_b(T') = \mathbb{1}\{\lfloor \log_2(\max_{t \in T'} |k_t|) \rfloor \geq b\}$ 
13:   if  $S_b(T') = 1$  then
14:     if  $T'$  is a singleton  $\{t\}$  then
15:       Encode the sign of  $k_t$ 
16:       Remove  $T'$  from  $LIS$ ; add  $t$  to  $LSN$ 
17:     else //  $T'$  is a set of sets
18:       if  $T' = \mathcal{D}(t) = \{\{t_c\} : t_c \in \mathcal{O}(t)\} \cup \{\mathcal{L}(t)\}$  (type A) then
19:         for each singleton set  $\{t_c\}$  in  $\mathcal{D}(t)$  do
20:           Encode  $S_b(\{t_c\}) = \mathbb{1}\{\lfloor \log_2(|k_{t_c}|) \rfloor \geq b\}$ 
21:           if  $S_b(\{t_c\}) = 1$  then
22:             Encode the sign of  $k_{t_c}$ 
23:             Add  $t_c$  to  $LSN$ 
24:           end if
25:         end for
26:       if  $\mathcal{L}(t)$  is not empty then
27:         Move  $\mathcal{L}(t)$  to the end of the  $LIS$  as a type B
28:       end if
29:       else //  $T' = \mathcal{L}(t) = \{\mathcal{D}(t_c) : t_c \in \mathcal{O}(t)\}$  (type B)
30:         Add each  $\mathcal{D}(t_c)$  in  $\mathcal{L}(t)$  to end of  $LIS$  as type A
31:       end if
32:       Delete  $T'$  from  $LIS$ 
33:     end if
34:   end if
35: end for
36: /* Iterate */
37:  $b \leftarrow b - 1$ 
38: if  $b > 0$  then go back to Refinement Pass
39: end if

```

Output: bitstream

is insignificant, then the other members of the set are also highly likely to be insignificant. This allows the whole set to be declared insignificant with a single bit.

For RAHT coefficients, a natural hierarchical set partition is given by the binary tree that defines the RAHT coefficients. This is a natural grouping because if a coefficient c_t at a node t is insignificant, then it likely means that the signal in the block associated with node t is flat. If the signal is flat, then all the RAHT coefficients below node t are also likely to be insignificant.

The binary tree that defines the RAHT coefficients is a sparse binary tree of depth $3L$ given by the Morton [8] codes of a set of occupied voxels (where the voxels have L bits of precision).¹ The root is at binary level $\ell = 0$ and the leaves are at binary level $\ell = 3L$. Each leaf corresponds to an occupied voxel. Each interior node at binary level ℓ corresponds to an occupied block of $2^{3L-\ell}$ voxels (at least one of which is occupied). The number of occupied voxels

¹Example: if a voxel geometry has 3 bits of precision as $x_0x_1x_2, y_0y_1y_2$ and $z_0z_1z_2$, then its Morton code is a 9-bit word $x_0y_0z_0x_1y_1z_1x_2y_2z_2$.

in a block is the weight of the block. The occupied block corresponding to an interior node at level ℓ is split along an axis (x , y , or z) into two sub-blocks, at least one of which is occupied. The occupied sub-blocks of a block correspond to child nodes of the block's node. Thus, each internal node has either 1 or 2 children. A child is designated either a left or right child depending on whether its Morton prefix (the path from the root to the node) is even or odd.

Each internal node t of the tree T corresponds to a RAHT coefficient if it has two children. Each node t that corresponds to a RAHT coefficient further corresponds to a set T_t of sets in a hierarchical set partition. The set T_t of sets always contains the singleton set $\{t\}$ as a member. However, T_t may also contain one or two additional sets as members. To be specific, let left-descendants (resp. right-descendants) of t denote descendants of t in the branch of T rooted at the left (resp. right) child of t . Let t_L (resp. t_R) denote the left-descendant (resp. right-descendant) corresponding to the RAHT coefficient closest to t . Then T_t contains as member sets T_{t_L} and/or T_{t_R} , if they exist. T_{t_L} and T_{t_R} are similarly defined, recursively. That is,

$$T_t = \begin{cases} \{t\} & \text{if } T_t \text{ is a singleton set} \\ \{\{t\}, T_{t_L}, T_{t_R}\} & \text{if } T_t \text{ is a compound set} \end{cases} \quad (1)$$

With this definition of the hierarchical set partition, the SPIHT algorithm can be applied to RAHT coefficients. To illustrate, Fig. 1 shows the state of the SPIHT for RAHT algorithm at the beginning of bit-plane b . In the Figure, a gray node t is one that has only one child and therefore does not have an associated RAHT coefficient. A black node is a node whose associated coefficient has already become significant in a previous bit-plane. An isolated white node is singleton set whose associated coefficient has not become significant yet. A white node rooting a branch is a set none of whose descendants have become significant yet.

Fig. 1: Illustration of the SPIHT algorithm for RAHT coefficients.

For each plane, the nodes of the tree are visited from level $\ell = 0$ (the root) to $\ell = 3L$ (the leaves). At each level, the nodes are processed, while nodes with no coefficient (1-child nodes) are skipped. Nodes that have already become significant in a previous bit-plane are processed in the refinement pass. Nodes that were not yet significant in a previous bit-plane are processed in the sorting pass.

IV. EXPERIMENTAL RESULTS

Simulations were carried out to investigate the performance of SPIHT for RAHT coefficients. In our tests, the first frame

TABLE I: Average BD PSNR-Y gains and average bitrate savings of SPIHT relative to RLGR.

Point Cloud	PSNR-Y [dB]	Bitrate [%]
Andrew	0.23	-4.93
David	0.56	-14.69
Ricardo	1.01	-21.72
Sarah	0.70	-14.70
Phil	0.12	-2.63
Soldier	0.21	-5.61
Loot	0.34	-10.47
Redandblack	0.41	-11.66
Longdress	0.30	-6.93
Average	0.43	-10.37

of five frontal upper-body sequences from Microsoft Research (“Andrew”, “David”, “Phil”, “Ricardo” and “Sarah”) [9] and four full body sequences from 8i Labs (“Longdress”, “Loot”, “Redandblack” and “Soldier”) [10] were used.

In Table I, Bjontegaard Delta (BD) [11] PSNR-Y gains and bitrate savings of SPIHT relative to RLGR for each tested sequence are presented. From Table I, we can see that the use of SPIHT with RAHT can outperform the use of RLGR with RAHT for all tested sequences. The best performances in terms of bitrate savings and in terms of PSNR-Y are 21.72% and 1.01 dB for “Ricardo”. The worst performances in terms of bitrate savings and in terms of PSNR-Y are 2.63% and 0.12 dB for “Phil”. The average bitrate savings is 10.37% and the average PSNR-Y gains is 0.43 dB.

Fig. 3 plots the rate-distortion performance of SPIHT relative to RLGR for the first frame of point cloud “Ricardo”. The figure also includes the rate-distortion performance of G-PCC with RAHT to give the reader an idea of how much gain G-PCC with RAHT offers over its baseline, RLGR with RAHT, as well as over SPIHT with RAHT. However, one should bear in mind that unlike SPIHT, the G-PCC bit-stream is not embedded. The embedded functionality of SPIHT is very desirable, as illustrated in Fig. 2, where projections of sequence “Ricardo” are shown for PCs decoded from a single bit-stream truncated at different bit-planes.

We analyzed the reasons that G-PCC with RAHT has performance gains over RLGR with RAHT. The gains come primarily from two mechanisms. The more important mechanism is prediction of the RAHT coefficients at higher levels from the decoded RAHT coefficients at lower levels. This is called predictive RAHT or upside-down RAHT [12]. Unfortunately for G-PCC, however, such predictive coding is problematic for the construction of embedded bitstreams. The second important mechanism is arithmetic coding with context modeling, including special symbols for jointly coding YUV coefficients that are all zero.

To address both of these mechanisms, we experimented with using arithmetic coding to compress SPIHT’s significance, sign, and refinement bits using a logistic probability model driven by previous context. For example, using the sigmoidal logistic function $\sigma(x) = 1/(1 + e^{-x})$, we used a function of the form

$$p = \sigma(a + a' \log |\hat{k}'| + a'' \log |\hat{k}''| + \log |\hat{k}'''|) \quad (2)$$

to predict the probability p that the set rooted at a node

Fig. 2: Illustration of projections of the point cloud sequence “Ricardo” decoded using SPIHT, and cutting the bit-stream at different bit-planes. From left to right: original (14 bit-planes), 11 bit-planes, 10 bit-planes, 9 bit-planes, and 8 bit-planes.

Fig. 3: Rate-distortion comparison among SPIHT, RLGR and G-PCC for the first frame of PC “Ricardo”.

Fig. 4: Rate-distortion comparison among SPIHT with context modeling (SPIHT-AC), SPIHT, and RLGR for the first frame of PC “Ricardo”.

is newly significant, given the compressed coefficients $\hat{k}'$, $\hat{k}''$, and $\hat{k}'''$ of its parent, grandparent, and great-grandparent. This approach significantly reduced the rate needed to code SPIHT’s significance, sign, and refinement bits, as shown in Fig. 4. Unfortunately the side information needed to represent the model parameters a , a' , a'' , a''' did not provide a net benefit for this approach. More investigation will be needed, for example using backward adaptive learning of the parameters, to further close the gap between embedded SPIHT entropy coding and the non-embedded entropy coding used in G-PCC. Another possible direction might be to replace the prediction gain in G-PCC with the transform gain from higher-order RAHT [13], for which it would be more straightforward to perform bit-plane coding.

V. CONCLUSIONS

This work presented an embedded coder for point cloud attributes. We proposed to encode RAHT coefficients with SPIHT and to use as hierarchical set partitions the binary tree that defines the coefficients. Our experimental results show that the use of SPIHT for RAHT can outperform the RLGR-based RAHT, while providing a fully embedded functionality to the codec. This, without use of any entropy coding apart from bit-plane and set partitions. As far as we know, this

is the first embedded PC coder. We plan to further research the use of context modeling for an arithmetic-coded SPIHT and to develop longer RAHT basis functions that mimic the coefficient prediction steps developed for G-PCC.

REFERENCES

- [1] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243–250, 1996.
- [2] 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, August 2016.
- [3] 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.
- [4] 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.
- [5] 3DG, "G-PCC Codec Description v12," ISO/IEC MPEG JTC1/SC29/WG11, Geneva, CH, Approved WG 11 document N18891, October 2020.
- [6] H. Malvar, "Adaptive Run-Length/Golomb-Rice Encoding of Quantized Generalized Gaussian Sources with Unknown Statistics," in *Data Compression Conference*, April 2006, pp. 23 – 32.
- [7] G. Sandri, R. L. de Queiroz, and P. A. Chou, "Comments on "Compression of 3D Point Clouds Using a Region-Adaptive Hierarchical Transform","" *ArXiv e-prints*, *arXiv: 1805.09146v1 [eess.IV]*.
- [8] G. M. Morton, "A computer oriented geodetic data base; and a new technique in file sequencing," IBM, Ottawa, Canada, Technical Report, 1966.
- [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 document m38673/M72012, May 2016.
- [10] 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.
- [11] G. Bjontegaard, "document VCEG-M33: Calculation of Average PSNR Differences between RD-Curves," in *ITU-T VCEG Meeting*, Austin, Texas, USA, 2001.
- [12] S. Lasserre and D. Flynn, "On an Improvement of RAHT to Exploit Attribute Correlation," ISO/IEC MPEG JTC1/SC29/WG11, Geneva, CH, Tech. Rep. m47378, March 2019.
- [13] P. A. Chou, M. Koroteev, and M. Krivokuća, "A volumetric approach to point cloud compression—part i: Attribute compression," *IEEE Transactions on Image Processing*, vol. 29, pp. 2203–2216, 2020.