Integer Alternative for the Region-Adaptive Hierarchical Transform

Gustavo P. Sandri, *Member, IEEE,* Philip A. Chou, *Fellow, IEEE,* Maja Krivokuća, and Ricardo L. de Queiroz *Fellow, IEEE*

***Abstract*—A recently-introduced coder based on region-adaptive hierarchical transform (RAHT) is being considered as a standard for the compression of point cloud attributes at MPEG (Moving Picture Experts Group). The RAHT coefficients can be encoded in many ways and the transform is based on a series of orthogonal 2×2 transform matrices with geometry-dependent floating-point entries. In order to remove computation ambiguity and facilitate deployment, fixed-point operations are often preferred. In this paper, we present an alternative RAHT description that allows for fixed-point implementation of its transform steps. It is based on matrix decompositions akin to lifting steps and a scaling of the quantization steps. Results are presented to show that the new fixed-point transform is, in practical terms, equivalent to the floating-point RAHT. For that we use a reasonable number of precision bits for the integer operations, e.g. 8 bits or more.**

***Index Terms*—point cloud compression, MPEG PCC, RAHT.**

I. Introduction

THE region-adaptive hierarchical transform (RAHT) was recently introduced in order to compress color information and other attributes of point clouds [1],[2]. RAHT is a hierarchical orthogonal sub-band transform that resembles an adaptive variation of a Haar wavelet transform. Unlike images which are dense, point clouds are sparsely occupied [3],[4] and RAHT adapts its coefficients to the location of occupied voxels [1]. RAHT coefficients, like in typical transform coders, are then quantized and entropy coded.

Real-time point cloud compression has received increased attention with recent developments in applications such as 3D telepresence systems [5]–[7]. There is also a recent interest in augmented reality systems and autonomous automotive navigation, which has further sparked interest in the use of point clouds as a means to represent 3D objects and has motivated the need for efficient point cloud compression techniques. The Moving Picture Experts Group (MPEG) and the Joint Photographic Experts Group (JPEG) have identified this trend and have recently initiated activities towards the standardization of point cloud compression technologies [8]–[11]. RAHT has been proposed to MPEG [12] as a means to efficiently encode the attributes of point clouds, and has been accepted by the MPEG-3DG/PCC (MPEG 3D Graphics/Point Cloud Coding) group as part of the Test Model for geometry-based point cloud compression (G-PCC) [13]. RAHT, however, is originally based upon a series of operations on real numbers. In standardization activities, recommendations should be as precise as possible, and operations with real numbers are usually challenging since floating-point operations often vary with hardware and software. In order to remove any ambiguity of floating-point operations inherent to RAHT, in its original proposal, we present a novel RAHT description which is designed to accommodate fixed-point arithmetic.

G. Sandri is with the Department of Electrical Engineering, University of Brasilia, Brazil, e-mail: gustavo.sandri@ieee.org

P. Chou is with Google, Seattle, Wa, USA, e-mail pachou@ieee.org

M. Krivokuća is with INRIA, Rennes, France, e-mail: majakri01@gmail.com

R. de Queiroz is with the Computer Science Dept, University of Brasilia, Brasilia, Brazil, e-mail queiroz@ieee.org

II. Integer-based Transform

RAHT is a variation of the Haar transform, which takes into account empty voxels. If a voxel to be transformed has no immediate neighbor, it is passed to the next level. If two neighbors exist, they are transformed, yielding a low-pass and a high-pass coefficient. The low-pass coefficient is then connected to other low-pass coefficients in a hierarchical and recursive manner. If a coeffcient has no immediate neighbor, it is passed to the next level, otherwise the pair is transformed, generating another pair of low- and high-pass coefficients. The problem is that, unlike with dense data, along this tree two low-pass coeffcients about to be transformed represent averages over different numbers of voxels (we call these numbers weights) [1]. The difference in weights is accounted for in RAHT as follows. Let two neighbor low-pass coefficients, $F_{\ell+1,2n}$ and $F_{\ell+1,2n+1}$ at level $\ell+1$ be combined to form a low- and high-pass pair of coefficients, $F_{\ell,n}$ and $G_{\ell,n}$, at level ℓ. If $w_{\ell+1,2n}$ and $w_{\ell+1,2n+1}$ are the respective weights of the input coefficients, then

$$\begin{bmatrix} F_{\ell,n} \\ G_{\ell,n} \end{bmatrix} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \begin{bmatrix} F_{\ell+1,2n} \\ F_{\ell+1,2n+1} \end{bmatrix} \tag{1}$$

where

$$a^2 = \frac{w_{\ell+1,2n}}{w_{\ell+1,2n} + w_{\ell+1,2n+1}},\ b^2 = \frac{w_{\ell+1,2n}}{w_{\ell+1,2n+1} + w_{\ell+1,2n+1}}. \tag{2}$$

Note that $a^2+b^2=1$ so that the transform is orthonormal, i.e. a Givens rotation. The typical *butterfly* used to describe its implementation is depicted in Fig. 1. If any of the weights are zero, the transform does not need to be computed, since either $a=1$ and $b=0$ or vice-versa. If the weights are equal, $a=b=1/\sqrt{2}$, and the transform is a 45 degree rotation, which is the Haar transform [14]. The Haar transform is popular for dense data like in images. This would be the case if all voxels are occupied. RAHT is intended for sparse data, where only some of the voxels in the voxel space are occupied, hence the need to track the weights of the coefficients.

Only the low-pass (F) coefficients are passed to further transforms. If we define, for simplicity of notation, $w_0 = w_{\ell+1,2n}$ and $w_1 = w_{\ell+1,2n+1}$, then

$$\begin{bmatrix} F_{\ell,n} \\ G_{\ell,n} \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{w_0}}{\sqrt{w_0+w_1}} & \frac{\sqrt{w_1}}{\sqrt{w_0+w_1}} \\ -\frac{\sqrt{w_1}}{\sqrt{w_0+w_1}} & \frac{\sqrt{w_0}}{\sqrt{w_0+w_1}} \end{bmatrix} \begin{bmatrix} F_{\ell+1,2n} \\ F_{\ell+1,2n+1} \end{bmatrix}. \quad (3)$$

This equation can be rewritten as

$$\begin{bmatrix} \frac{F_{\ell,n}}{\sqrt{w_0+w_1}} \\ G_{\ell,n}\frac{\sqrt{w_0+w_1}}{\sqrt{w_0 w_1}} \end{bmatrix} = \begin{bmatrix} \frac{w_0}{w_0+w_1} & \frac{w_1}{w_0+w_1} \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \frac{F_{\ell+1,2n}}{\sqrt{w_0}} \\ \frac{F_{\ell+1,2n+1}}{\sqrt{w_1}} \end{bmatrix} \quad (4)$$

or, assuming

$$F'_{\ell,n} = \frac{F_{\ell,n}}{\sqrt{w_0+w_1}} \quad (5)$$

and

$$G'_{\ell,n} = G_{\ell,n}\frac{\sqrt{w_0+w_1}}{\sqrt{w_0 w_1}}, \quad (6)$$

then the direct transform can be written as

$$\begin{bmatrix} F'_{\ell,n} \\ G'_{\ell,n} \end{bmatrix} = \begin{bmatrix} a^2 & b^2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} F'_{\ell+1,2n} \\ F'_{\ell+1,2n+1} \end{bmatrix}. \quad (7)$$

Similarly, for the inverse,

$$\begin{bmatrix} F'_{\ell+1,2n} \\ F'_{\ell+1,2n+1} \end{bmatrix} = \begin{bmatrix} 1 & -b^2 \\ 1 & a^2 \end{bmatrix} \begin{bmatrix} F'_{\ell,n} \\ G'_{\ell,n} \end{bmatrix}. \quad (8)$$

If we consider that $a^2 = 1 - b^2$ and the following matrix decomposition [14]

$$\begin{bmatrix} a^2 & b^2 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & b^2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}, \quad (9)$$

then

$$\begin{bmatrix} F'_{\ell,n} \\ G'_{\ell,n} \end{bmatrix} = \begin{bmatrix} 1 & \Phi \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} F'_{\ell+1,2n} \\ F'_{\ell+1,2n+1} \end{bmatrix}, \quad (10)$$

which is implemented with two additions and one multiplication, and where Φ is a fixed-point approximation of b^2, and $b^2 = w_1/(w_0+w_1)$. The *butterfly* to implement the Givens rotation corresponding to each RAHT step is then simplified to the one depicted in Fig. 1. The fixed-point approximation involves an integer division and a choice of the number of precision bits. The result is a fast transform without using floating point arithmetic.

There is a problem, though, with the proposed transform. Since there are scalings in both input and output samples, as we use $F'_{\ell,n}$ and $G'_{\ell,n}$ (equations 5 and 6), the transform itself is no longer orthonormal. Coefficients $F'_{\ell,n}$ and $G'_{\ell,n}$ have an embedded gain compared to the original $F_{\ell,n}$ and $G_{\ell,n}$

Fortunately, we can compensate them during quantization. If we assume, for simplicity of notation, a constant step size Δ for all levels and coefficients $F_{\ell,n}$ and $G_{\ell,n}$, the corrected stepsizes for the scaled coefficients should be

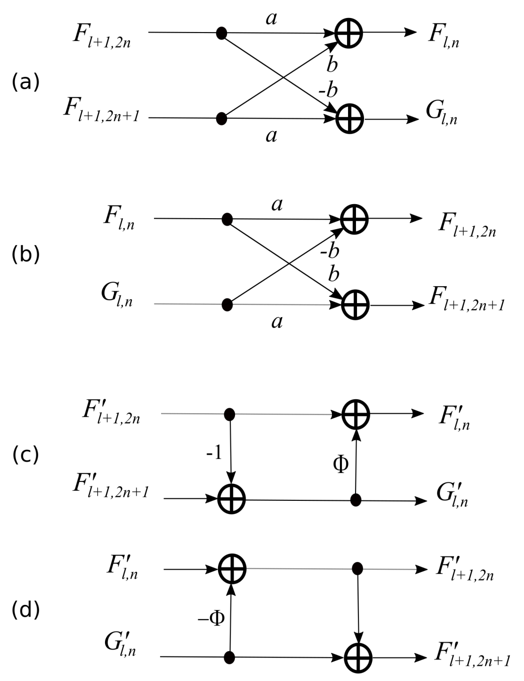


Fig. 1. Lifting-style integer computation of the RAHT butterflies: (a) original butterfly for the forward transform; (b) the butterfly for the inverse transform; (c) the simplified fixed-point butterfly; with (d) its inverse counterpart.

$$F'_{\ell,n} \to \Delta\frac{1}{\sqrt{w_0+w_1}}, \quad G'_{\ell,n} \to \Delta\frac{\sqrt{w_0+w_1}}{\sqrt{w_0 w_1}}. \quad (11)$$

Recall that w_0 and w_1 are notational simplifications. If w_0 and w_1 are the weights of the input low-pass (F) coefficients, w_0+w_1 is the weight of the output low-pass coefficient. Hence,

$$F'_{\ell,n} = \frac{F_{\ell,n}}{\sqrt{w_{\ell+1,2n} + w_{\ell+1,2n+1}}} = \frac{F_{\ell,n}}{\sqrt{w_{\ell,n}}}, \quad (12)$$

and the scaled output of one transform stage is at the proper scale to be input to the next stage. When we reach the top of the tree, for example at level 0, the overall DC coefficient for the whole point cloud, $F_{0,n}$ is quantized using step size $\Delta/\sqrt{w_{0,n}}$. This requires us to compute one square root per point cloud. However, the high-pass coefficients, $G'_{\ell,n}$, are always quantized and encoded. For $G'_{\ell,n}$, the step size should be

$$\Delta \sqrt{\frac{w_{\ell+1,2n} + w_{\ell+1,2n+1}}{w_{\ell+1,2n} w_{\ell+1,2n+1}}}, \quad (13)$$

which demands the computation of a fixed-point square root per coefficient. Along with the square root, it also demands an addition, a division and a multiplication, all using fixed-point arithmetic.

Both the proposed fixed-point implementation and the floating-point one have the same complexity $O(nlog(n))$. They differ in how the transform is computed. In the fixed-point implementation we need to convert the point cloud color to a fixed-point representation prior to the transform. Furthermore, the transform is executed using only integer operations (sum, subtraction, multiplication, division and square root). We need to compute as many fixed-point square roots as RAHT coefficients, which is the same number as occupied voxels in the point cloud. Therefore, we recommend using a fast fixed-point square root approximation.

III. Experiments

We compared our fixed-point RAHT implementation to the floating-point one, within the context of a RAHT-based point cloud coder. In our naive implementation, the fixed-point RAHT is actually slower than the floating-point one. It actually changes from computer to computer and depends on architecture and compilers. At the same time, the fixed-point implementation heavily depends on the integer-only square-root computation, which, in our implementation, is recursive. Since we used a simple algorithm, far from optimized to any architecture, we believe this time could be significantly reduced.

In order to analyse the impact of fixed-point arithmetic we measure the effect of the change on the coder performance in terms of rate-distortion (RD) curves. Rate is computed as the number of bits used to write the encoded file to disk divided by the number of occupied voxels in the point cloud, *i.e.*, the number of bits to encode all YUV color components, while distortion is measured as peak signal-to-noise ratio (PSNR) comparing the original and reconstructed Y-channel attributes. The PSNR for the U- and V-channels are very similar to that of the Y-channel, being around 91% and 87% the value of the Y-channel PSNR, respectively, regardless of the rate. In our tests, we used the same number of precision bits for both Φ and the square root computation.

Our test set is illustrated in Fig. 2 [15],[16]. Rate-distortion curves are shown in Fig. 3, comparing the floating-point implementation against fixed-point implementations with varying precision. The two sets of curves are very similar and show the curve for an 8-bit implementation to superimpose the floating-point one. The curve sets for the other point clouds were omitted because, qualitatively, they are the same as those two. In order to better appreciate how close the curves are, the average PSNR difference between fixed-point curves and the floating-point one was computed for every point cloud and the results are shown in Table I.

IV. Conclusion

We have introduced a new RAHT definition that allows for fixed-point implementation (that is emulated by integer operations) without impacting the compression performance. Its applications are in real-time point cloud transmission (e.g. augmented reality, autonomous navigation, etc.). This result can effectively change the RAHT coder description in the context of point cloud compression standardization. Results show that 8-bit fixed-point precision may be good enough for our compression applications, and any representation using 10 bits and above yields negligible compression performance impact. The proposed fixed-point RAHT implementation has already been incorporated into the software base (Test Model 13) of the MPEG G-PCC point cloud codec [13]).

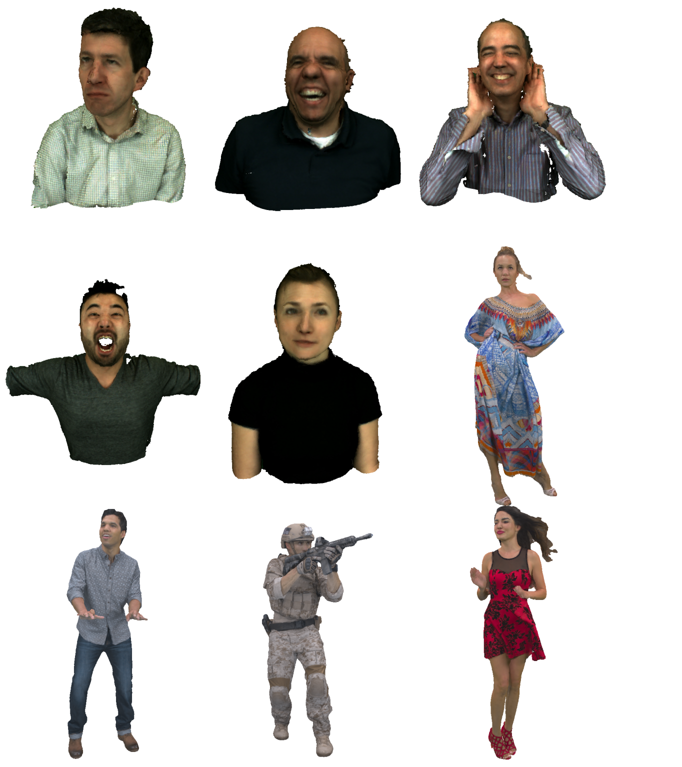


Fig. 2. Rendering of a view of our point cloud test set. From left to right, top to bottom: Andrew, Ricardo, Phil, David, Sarah, longdress, loot, soldier, and redandblack.

References

[1] 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.

[2] G. Sandri, R. L. de Queiroz and P. A. Chou, "Comments on 'Compression of 3D Point Clouds Using a Region-Adaptive Hierarchical Transform'," arXiv:1805.09146v1 [eess.IV], 23 May 2018.

[3] M. Gross and H. Pfister (eds.), *Point-Based Graphics*, Elsevier, Burlington, MA, USA, 2007.

[4] R. B. Rusu and S. Cousins, 3D is here: Point Cloud Library (PCL), *Proc. IEEE Intl. Conf. on Robotics and Automation*, ICRA, Shanghai, China, pp. 1-4, May 2011.

[5] S. Orts-Escolano et al., "Holoportation: virtual 3D teleportation in real-time," *Proc. 29th ACM User Interface Software and Technology Symposium,* Tokyo , Japan, Oct. 2016.

[6] C. Zhang, Q. Cai, P. Chou, Z. Zhang, and R. Martin-Brualla, "Viewport: a fully distributed immersive teleconferencing system with infrared dot pattern," *IEEE Multimedia*, vol. 20, no. 1, pp. 17–27, 2013.

[7] A. Collet, et al., "High-quality streamable free-viewpoint video," *ACM Trans. on Graphics*, Vol. 34, No. 4, Article 69, Aug. 2015.

[8] C. Tulvan, R. Mekuria, Z. Li, and S. Laserre, Use cases for point cloud compression, ISO/IEC JTC1/SC29/WG11 MPEG, document N16331, Jun. 2016.

[9] R. Mekuria, C. Tulvan, and Z. Li, Requirements for point cloud compression, ISO/IEC JTC1/SC29/WG11 MPEG, document N16330, Jun. 2016.

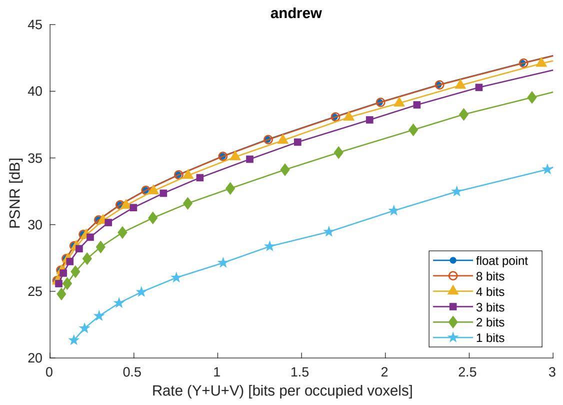
[10] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. Tourapis, and V. Zakharchenko, "Emerging MPEG Standards for Point Cloud Compression," *IEEE J. Emerging Topics in Circuits and Systems,* accepted for publication, 2018.

[11] T. Ebrahimi, S. Foessel, F. Pereira, P. Schelkens, "JPEG Pleno: toward an efficient representation of visual reality," *IEEE Multimedia,* Vol. 23, No. 4, pp. 14–20, Nov. 2016.

TABLE I

AVERAGE PSNR DIFFERENCE BETWEEN FIXED- AND FLOATING-POINT RAHT CODER IMPLEMENTATIONS. THE AVERAGE WAS COMPUTED IN THE RATE RANGE FROM 0.3 TO 3 BITS PER OCCUPIED VOXELS. PRECISION OF THE FIXED-POINT IMPLEMENTATION IS INDICATED.

	andrew	david	phil	ricardo	sarah	longdress	loot	redandblack	soldier
1 bit	8.200	8.571	8.251	8.602	9.252	7.832	8.559	7.771	8.189
2 bits	2.685	2.561	2.615	2.671	2.856	2.518	2.681	2.405	2.622
3 bits	0.998	1.001	0.952	1.009	1.106	0.937	0.977	0.855	0.947
4 bits	0.430	0.497	0.414	0.472	0.481	0.390	0.424	0.348	0.391
5 bits	0.193	0.170	0.174	0.210	0.234	0.179	0.186	0.158	0.173
6 bits	0.095	0.081	0.085	0.076	0.098	0.084	0.080	0.074	0.083
7 bits	0.044	0.043	0.040	0.043	0.058	0.042	0.039	0.036	0.040
8 bits	0.022	0.027	0.021	0.030	0.037	0.020	0.018	0.018	0.016
9 bits	0.014	0.016	0.013	0.019	0.030	0.010	0.012	0.009	0.010
10 bits	0.007	0.013	0.007	0.016	0.025	0.005	0.007	0.005	0.005
11 bits	0.004	0.018	0.005	0.016	0.039	0.002	0.005	0.003	0.003



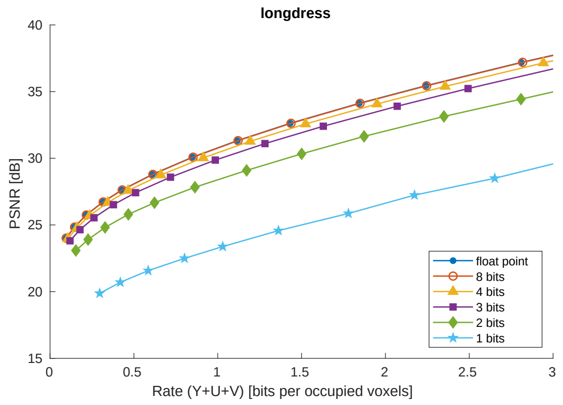


Fig. 3. PSNR curves comparing the floating-point implementation with the fixed-point ones for two point clouds and different numbers of precision bits. Qualitatively, all sets of curves for all nine point clouds in our test set are very similar.

[12] P. A. Chou and R. L. de Queiroz, Transform coder for point cloud attributes, ISO/IEC JTC1/SC29/WG11 MPEG, input document m38674, May 2016.

[13] K. Mammou and P. Chou, “PCC Test Model Category 13 v2,” ISO/IEC JTC1/SC29/WG11 MPEG document N17519, Apr. 2018.

[14] G. Strang and T. Nguyen, *Wavelets and Filter Banks,* Wellesley-Cambridge Press, Wellesley, MA, USA, 1996.

[15] C. Loop, Q. Cai, S.O. Escolano, and P.A. Chou, “Microsoft voxelized upper bodies - a voxelized point cloud dataset,” in *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document m38673/M72012*, May 2016.

[16] E. d’Eon, B. Harrison, T. Myers, and P. A. Chou, “8i Voxelized Full Bodies - A Voxelized Point Cloud Dataset,” in *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006*, Geneva, January 2017.