Transform Coding for Point Clouds Using a Gaussian Process Model

Ricardo L. de Queiroz, *Fellow, IEEE*, and Philip A. Chou, *Fellow, IEEE*

***Abstract*— We propose using stationary Gaussian processes (GPs) to model the statistics of the signal on points in a point cloud, which can be considered samples of a GP at the positions of the points. Furthermore, we propose using Gaussian process transforms (GPTs), which are Karhunen–Loève transforms of the GP, as the basis of transform coding of the signal. Focusing on colored 3D point clouds, we propose a transform coder that breaks the point cloud into blocks, transforms the blocks using GPTs, and entropy codes the quantized coefficients. The GPT for each block is derived from both the covariance function of the GP and the locations of the points in the block, which are separately encoded. The covariance function of the GP is parameterized, and its parameters are sent as side information. The quantized coefficients are sorted by the eigenvalues of the GPTs, binned, and encoded using an arithmetic coder with bin-dependent Laplacian models, whose parameters are also sent as side information. Results indicate that transform coding of 3D point cloud colors using the proposed GPT and entropy coding achieves superior compression performance on most of our data sets.**

***Index Terms*— Gaussian processes, graph signal processing, graph Fourier transform, Karhunen-Loeve transform, point cloud, data compression.**

I. Introduction

COLORED 3D point clouds have recently emerged as an effective means to represent information in computer-graphics-based telepresence systems [1]. In order to represent a person or object in 3D to a remote receiver, instead of sending multiple camera views of the scene (multiview approach), we represent the scene as a collection of colored 3D points arranged in a raster of volumetric elements (voxels) [1], [2]. In general, we are not concerned with transmitting all the voxels in the raster, but only the voxels at the boundaries of the objects, which may be visible from a distant point of view [2]. Dynamic point clouds can be used to convey moving 3D people or objects to the receiver with the aid of special devices such as near eye displays. Figure 1 shows various viewpoints of one frame of a voxelized point cloud sequence.

Manuscript received April 22, 2016; revised September 16, 2016 and February 15, 2017; accepted April 24, 2017. Date of publication April 28, 2017; date of current version May 19, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Feng Wu. *(Corresponding author: Ricardo L. de Queiroz.)*

R. L. de Queiroz is with the Computer Science Department, Universidade de Brasilia, Brasilia 70910-900, Brazil (e-mail: queiroz@ieee.org).

P. A. Chou was with the Microsoft Research, Redmond, WA 98052 USA. He is now with 8i Labs., Bellevue, WA 98004 USA (e-mail: pachou@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2017.2699922

Fig. 1. Various viewpoints of one frame of a voxelized point cloud sequence.

Many works have described methods for the compression of 3D objects in a mesh representation [3]–[14]. Compression of point clouds has also been addressed in a few works [15]–[17]. A method based on the Graph Fourier Transform (GFT) [18] seems to have been the state-of-the-art in compression of 3D point cloud colors, until a method known as RAHT (region adaptive hierarchical ransform) was developed recently [19]. While the GFT requires complex matrix decompositions, RAHT is a much lower-complexity transform, making it suitable for real-time implementation, while being able to match the GFT with an improved entropy coder. The GFT-based work was also extended to explore temporal redundancy [20].

In this paper, we propose a method for compressing the colors of a 3D point cloud, assuming that the geometry of the point cloud has already been compressed and transmitted. Our approach is to model the statistics of the point colors by assuming that they are samples of a stationary Gaussian Process (GP) defined over an infinite 3D space. The samples are taken at the corresponding point positions. The covariance function of the GP is used to derive the covariance matrix of the colors, which in turn is used to derive the Karhunen-Loève transform (KLT) of the color signal. We call this KLT the Gaussian Process Transform (GPT) to distinguish it from other KLTs, such as the GFT, which assumes a different covariance structure than that given by a GP of the color signal. The GPT is in turn used in a transform coding system to code the colors of the point cloud.

Though we apply this technique to the compression of 3D point cloud colors, the same approach can be used for compression of other attributes or signals on the 3D points, as well as on points in higher dimensional spaces.

Section II discusses how we model the color statistics with Gaussian processes. Section III introduces the GPT. Section IV describes how we use the GPT for transform coding, and Section V goes into the details of our entropy coder. Finally, Section VI provides experimental results, and Section VII our conclusions.

II. Modeling Voxel Colors as Gaussian Processes

A point cloud is a collection of points $\{v_i\}$ each representing one or more real-valued attributes at a particular position in space. Attributes typically include color vectors, but may also include normal vectors, motion vectors, etc. For concreteness and without loss of generality, we assume that the attributes are colors. Thus a point in the cloud may be represented by a tuple

$$v_i = (x_i, y_i, z_i, Y_i, U_i, V_i), \tag{1}$$

such that x, y, and z describe the point position (geometry) while Y, U, and V describe the color components of the point. Though it is not required for our method, for simplicity in modeling, we constrain the points to lie on a regular grid. In particular, we divide a bounding cube for the points into $L \times L \times L$ voxels, so that we may take x_i, y_i, and z_i to be integers such that $0 \leq x_i, y_i, z_i < L$. A voxel is said to be *occupied* if it contains a point of the point cloud, and points of the point cloud are identified with occupied voxels. We refer to the set of occupied voxels as a voxelized point cloud (VPC). Whenever it is clear from the context we refer to occupied voxels simply as *voxels*.

Let $\mathbf{v}_1 \ldots, \mathbf{v}_M$ be the points of a voxelized point cloud with integer components, and let $Y(\mathbf{v}_1), \ldots, Y(\mathbf{v}_M)$ be the color signal (luminance, for example, or other signal) defined at those points. We model $Y(\mathbf{v}_1), \ldots, Y(\mathbf{v}_M)$ as being samples of a discrete-space stationary Gaussian Process $\{Y(\mathbf{v})\}$ defined everywhere on an infinite 3D integer lattice with mean $\mu = E[Y(\mathbf{v})]$ and covariance function

$$R(\mathbf{d}) = E[(Y(\mathbf{v}) - \mu)(Y(\mathbf{v} - \mathbf{d}) - \mu)]. \tag{2}$$

That is, $E[(Y(\mathbf{v}_i) - \mu)(Y(\mathbf{v}_j) - \mu)] = R(\mathbf{v}_i - \mathbf{v}_j)$. The mean μ and covariance function $R(\mathbf{d})$ completely characterize $\{Y(\mathbf{v})\}$. As an alternative to the covariance function $R(\mathbf{d})$, the precision function $Q(\mathbf{d})$ may be used to characterize the process. As described later in this section, the precision function is the inverse Fourier transform of the reciprocal of the Fourier transform of the covariance function. The precision function is especially useful for characterizing a stationary Gaussian Process if it is a stationary Gauss Markov random field (GMRF), in which case the precision function has only a finite number of non-zero elements.

In this paper, we model the GP $\{Y(\mathbf{v})\}$ in four different ways. In the first two ways, we model its covariance function $R(\mathbf{d})$ either (i) non-parametrically (NP) or (ii) parametrically as the covariance function of an isotropic Ornstein-Uhlenbeck (OU) random process, which has a single parameter. In the second two ways, we model the precision function $Q(\mathbf{d})$ either (iii) using coefficients that fall off inversely with distance (ID) up to a threshold, as commonly used in graph signal processing, or (iv) using the coefficients of an auto-regressive (AR) process. In the following subsections, we introduce these models and explain how we estimate their parameters from data if necessary.

A. Non-Parametric (NP) Model

Our first model of the process $\{Y(\mathbf{v})\}$ is non-parametric (NP) in the sense that our estimate of $R(\mathbf{d})$ is not constrained to be a member of any family of covariance functions parametrized by a finite number of parameters. However, we do constrain $R(\mathbf{d})$ to have reflective symmetry across the planes $x = 0$, $y = 0$, and $z = 0$; that is, $R(\pm x, \pm y, \pm z)$ are all equal. We enforce this by assuming that $R(\mathbf{d})$ depends only on $|\mathbf{d}|$, which is similar to rotational symmetry except that $\mathbf{d}$ lies on a grid, so $R(\mathbf{d})$ cannot be strictly rotationally symmetric.

To estimate $R(\mathbf{d}) = R(d)$, where $d = |\mathbf{d}|$, we let Φ_d be the set of all pairs of voxels $(\mathbf{v}, \mathbf{m})$ in a data set such that $|\mathbf{v} - \mathbf{m}| = d$, and let $N_d = ||\Phi_d||$ be the number of such pairs. Note that if $(\mathbf{v}, \mathbf{m}) \in \Phi_d$, then $(\mathbf{m}, \mathbf{v})$ is also in the same set. Let the estimate of its mean be

$$\mu_d = \frac{1}{N_d} \sum_{\mathbf{v}:(\mathbf{v},\mathbf{m})\in\Phi_d} Y(\mathbf{v}), \tag{3}$$

such that an unbiased estimate of $R(d)$ is

$$\varphi_d(d) = \frac{1}{N_d - 1} \sum_{(\mathbf{v},\mathbf{m})\in\Phi_d} (Y(\mathbf{v}) - \mu_d)\,(Y(\mathbf{m}) - \mu_d)\,. \tag{4}$$

Likewise

$$\varphi_d(0) = \frac{1}{N_d - 1} \sum_{(\mathbf{v},\mathbf{m})\in\Phi_d} (Y(\mathbf{v}) - \mu_d)^2 \tag{5}$$

is an unbiased estimate of the variance and

$$\varphi(d) = \frac{\varphi_d(d)}{\varphi_d(0)} \tag{6}$$

can be considered an estimate of the normalized correlation coefficient $R(d)/R(0)$, whose magnitude is at most 1 ($\varphi(0) = 1$ is the variance of the normalized unit-variance correlation) by the Cauchy-Schwartz inequality. Our non-parametric covariance function model for $R(\mathbf{d})$ is

$$R_{\mathrm{NP}}(\mathbf{d}) = R_{\mathrm{NP}}(d) = \varphi(d)\sigma_Y^2, \tag{7}$$

which can be simplified to $R_{\mathrm{NP}}(\mathbf{d}) = \varphi(d)$ if the absolute variance is unimportant.

B. Ornstein-Uhlenbeck (OU) Model

Our second model of $\{Y(\mathbf{v})\}$ is an Ornstein-Uhlenbeck (OU) process, which is a stationary zero-mean Gaussian Process in which the covariance matrix is isotropic, or radially symmetric, and aside from its magnitude has a single parameter ρ, which models the decay of the covariance,

$$R_{OU}(\mathbf{d}) = \sigma_Y^2 \rho^{|\mathbf{d}|}. \tag{8}$$

Since there is only a single parameter, it is relatively easy using numerical means to choose the parameter to maximize the likelihood

$$p(\mathbf{y}) = \frac{1}{(2\pi)^{\ell/2}|\mathbf{R}|^{1/2}} \exp\left(-\frac{1}{2}\mathbf{y}^T\mathbf{R}^{-1}\mathbf{y}\right), \tag{9}$$

where $\mathbf{y} = [y_i]$ with $y_i = Y(\mathbf{v}_i)$ and $\mathbf{R} = [r_{ij}]$ with $r_{ij} = \sigma_Y^2 \rho^{|\mathbf{v}_i - \mathbf{v}_j|}$.

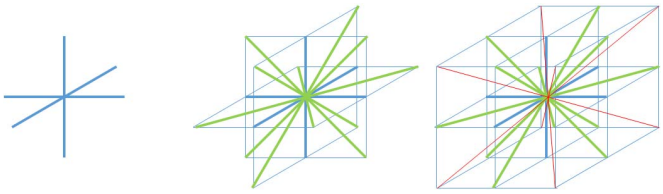


Fig. 2. Local neighborhoods in $\mathbb{Z}^3$ for $d_{\max} = 1, \sqrt{2}, \sqrt{3}$. Blue are $d = 1$ edges; green are $d = \sqrt{2}$ edges; red are $d = \sqrt{3}$ edges. Respective local neighborhoods sizes are 6, 18, 26.

C. Inverse Distance (ID) Model

Our third model of $\{Y(\mathbf{v})\}$ is a stationary zero-mean GMRF defined on an infinite 3D lattice, $\mathbb{Z}^3 = \{\mathbf{v}\}$. Such a GMRF is defined most naturally not in terms of its covariance function $R(\mathbf{d})$ but rather in terms of its precision function $Q(\mathbf{d})$, which is related to $R(\mathbf{d})$ through the inverse Fourier transform of the power spectral density,

$$S(\boldsymbol{\omega}) = \frac{1}{(2\pi)^3} \sum_{\mathbf{d}\in\mathbb{Z}^3} R(\mathbf{d}) e^{-i\mathbf{d}\cdot\boldsymbol{\omega}}, \tag{10}$$

$$Q(\mathbf{d}) = \int_{[-\pi,\pi]^3} \frac{1}{S(\boldsymbol{\omega})} e^{i\mathbf{d}\cdot\boldsymbol{\omega}} d\boldsymbol{\omega}, \tag{11}$$

provided $S(\boldsymbol{\omega}) > 0$ for all $\boldsymbol{\omega} \in [-\pi, \pi]^3$. Note that $R(\mathbf{d})$ is real and symmetric, and hence so is $S(\boldsymbol{\omega})$, $1/S(\boldsymbol{\omega})$, and $Q(\mathbf{d})$. Also note that $R(\mathbf{d})$ and $Q(\mathbf{d})$ are inverses in the sense that their correlation is the unit impulse $\delta(\mathbf{d})$. This can be verified by taking the Fourier transform of each (namely, $S(\boldsymbol{\omega})$ and $1/S(\boldsymbol{\omega})$) and multiplying them in the frequency domain to obtain a constant.

It is convenient to use $Q(\mathbf{d})$ instead of $R(\mathbf{d})$ to characterize a stationary Gaussian Process if $Q(\mathbf{d})$ is non-zero for only a finite number of values of $\mathbf{d}$, because the process would then be finitely parametrized. In this case, the process is a stationary GMRF defined on an infinite lattice, also known as a *conditional autoregression* [25, Sec. 2.6.5].

We model $\{Y(\mathbf{v})\}$ as a conditional autoregression specified by an Inverse Distance (ID) weight function,

$$W(\mathbf{d}) = \begin{cases} 1/|\mathbf{d}|, & \text{whenever } 0 < |\mathbf{d}| \le d_{\max}, \\ 0, & \text{otherwise}, \end{cases} \tag{12}$$

for some distance cutoff $d_{\max}$. The precision function can be defined in terms of the weight function as

$$Q(\mathbf{d}) = \begin{cases} -W(\mathbf{d}) & \text{for } \mathbf{d} \neq 0 \\ \sum_{\mathbf{d}'} W(\mathbf{d}') & \text{for } \mathbf{d} = 0. \end{cases} \tag{13}$$

Thus $Q(\mathbf{d})$ is non-zero only for $|\mathbf{d}| \le d_{\max}$. When applied to nodes on the integer lattice $\mathbb{Z}^3$, typical cutoffs satisfy $d_{\max} = 1, \sqrt{2}, \sqrt{3}$, leading to the local neighborhoods illustrated in Fig. 2.

D. Auto-Regressive (AR) Model

Our last model of $\{Y(\mathbf{v})\}$ is also a conditional autoregression. In particular, it is a stationary zero-mean GP $\{Y(\mathbf{v})\}$ satisfying the auto-regressive equation

$$Y(\mathbf{v}) = X(\mathbf{v}) - \sum_{\mathbf{d}\in\mathcal{N}} a_{\mathbf{d}} Y(\mathbf{v} - \mathbf{d}), \tag{14}$$

where $X(\mathbf{v})$ is an iid $N(0, \sigma_x^2)$ Gaussian white noise, $a_{\mathbf{d}}$ is a coefficient for offset $\mathbf{d}$, and $\mathcal{N}$ is a set of offsets in a neighborhood around $\mathbf{0}$, such as $\{\mathbf{d} : 0 < |\mathbf{d}| \le d_{\max}\}$ (or some subset of it) for some $d_{\max}$. This process is parameterized by σ_x^2 and $\{a_{\mathbf{d}} : \mathbf{d} \in \mathcal{N}\}$.

For the moment, so that we can use finite-dimensional algebra, assume that (14) is defined not on the infinite 3D lattice $\mathbb{Z}^3$ but on the finite 3D torus $\mathbb{Z}_N^3$, and assume that $\mathbf{v} - \mathbf{d}$ in (14) is taken modulo N component-wise. Then we may write

$$\mathbf{x} = \mathbf{A}\mathbf{y}, \tag{15}$$

where $\mathbf{y} = [Y(\mathbf{v}_i)]$ and $\mathbf{x} = [X(\mathbf{v}_i)]$ are vectors of length $\ell = N^3$ and $\mathbf{A}$ is an $\ell \times \ell$ matrix whose (i, j)th entry equals 1 if $i = j$, equals $a_{\mathbf{d}}$ if $\mathbf{v}_i - \mathbf{d} = \mathbf{v}_j$ (modulo N) for $\mathbf{d} \in \mathcal{N}$, and equals 0 otherwise.

Since the probability density of $\mathbf{x}$ is

$$p(\mathbf{x}) = \frac{1}{(2\pi\sigma_x^2)^{\ell/2}} \exp\left(-\frac{1}{2\sigma_x^2}\mathbf{x}^T\mathbf{x}\right), \tag{16}$$

the probability density of $\mathbf{y}$ can be written

$$p(\mathbf{y}) = \frac{|\mathbf{A}|}{(2\pi\sigma_x^2)^{\ell/2}} \exp\left(-\frac{1}{2\sigma_x^2}\mathbf{y}^T\mathbf{A}^T\mathbf{A}\mathbf{y}\right) \tag{17}$$

$$= \frac{1}{(2\pi)^{\ell/2}|\mathbf{R}|^{1/2}} \exp\left(-\frac{1}{2}\mathbf{y}^T\mathbf{R}^{-1}\mathbf{y}\right), \tag{18}$$

where $\mathbf{R} = \sigma_x^2(\mathbf{A}^T\mathbf{A})^{-1}$ is the covariance matrix of $\mathbf{y}$. The inverse of the covariance matrix is called the *precision matrix* [21]. It can be seen that the coefficients q_{ij} in the precision matrix $\mathbf{Q} = \mathbf{R}^{-1} = \sigma_x^{-2}(\mathbf{A}^T\mathbf{A})$ of $\mathbf{y}$ are the coefficients of the autocorrelation of the coefficients in $\mathbf{A}$ (scaled by σ_x^{-2}). Specifically, if we extend the definition of $a_{\mathbf{d}}$ to all values of $\mathbf{d}$ as

$$\bar{a}_{\mathbf{d}} = \begin{cases} 1 & \text{if } \mathbf{d} = \mathbf{0} \\ a_{\mathbf{d}} & \text{if } \mathbf{d} \in \mathcal{N} \\ 0 & \text{otherwise} \end{cases}$$

then we may write $\mathbf{A} = [\bar{a}_{\mathbf{v}_i - \mathbf{v}_j}]$ and

$$q_{ij} = \sigma_x^{-2} \sum_k \bar{a}_{\mathbf{v}_k - \mathbf{v}_i} \bar{a}_{\mathbf{v}_k - \mathbf{v}_j} = \sigma_x^{-2} \sum_{\mathbf{d}'} \bar{a}_{\mathbf{d}'} \bar{a}_{\mathbf{d}' + \mathbf{v}_i - \mathbf{v}_j}, \tag{19}$$

where all subscripts are taken modulo N. Clearly, q_{ij} depends only on $\mathbf{v}_i - \mathbf{v}_j$.

We are now in a position to remove the toroidal assumption. We define the precision function in terms of $\{a_{\mathbf{d}} : \mathbf{d} \in \mathcal{N}\}$ as

$$Q(\mathbf{d}) = \sigma_x^{-2} \sum_{\mathbf{d}'} \bar{a}_{\mathbf{d}'} \bar{a}_{\mathbf{d}' + \mathbf{d}}, \tag{20}$$

where now the subscripts are in $\mathbb{Z}^3$. From this, we can see that $Q(\mathbf{d})$ can be non-zero only when $|\mathbf{d}| \le 2d_{\max}$. Thus the process defined by this precision function is a conditional autoregression.

It remains to specify the coefficients $\{a_{\mathbf{d}}\}$. In principle, these can be chosen to fit a set of data $\mathbf{y}$ by maximizing the log-likelihood $\log p(\mathbf{y})$ over the variables $\{a_{\mathbf{d}}\}$. However, this is an "extremely non-linear" problem that is difficult to solve [22]. A close approximation is to maximize the pseudo log-likelihood

$$-\frac{1}{2}\mathbf{y}^T\mathbf{R}^{-1}\mathbf{y}, \tag{21}$$

or equivalently, minimize the squared prediction error $||\mathbf{x}||^2$ over the variables $\{a_{\mathbf{d}}\}$, where the prediction of $Y(\mathbf{v})$ is $-\sum_{\mathbf{d}\in\mathcal{N}} a_{\mathbf{d}} Y(\mathbf{v}-\mathbf{d})$ and, therefore, the prediction error is

$$X(\mathbf{v}) = Y(\mathbf{v}) + \sum_{\mathbf{d}\in\mathcal{N}} a_{\mathbf{d}} Y(\mathbf{v}-\mathbf{d}). \tag{22}$$

The mean squared prediction error can be minimized using the principle of orthogonality in Hilbert spaces, so that the error $X(\mathbf{v})$ in the approximation of $Y(\mathbf{v})$ by its projection onto a span of basis vectors $\{Y(\mathbf{v}-\mathbf{d})\}$ must be orthogonal to each of the basis vectors. That is, for all $\mathbf{d}\in\mathcal{N}$,

$$\begin{aligned} 0 &= \langle X(\mathbf{v}), Y(\mathbf{v}-\mathbf{d})\rangle \\ &= \langle Y(\mathbf{v}) + \sum_{\mathbf{d}'\in\mathcal{N}} a_{\mathbf{d}'} Y(\mathbf{v}-\mathbf{d}'), Y(\mathbf{v}-\mathbf{d})\rangle \\ &= \langle Y(\mathbf{v}), Y(\mathbf{v}-\mathbf{d})\rangle \\ &\quad + \sum_{\mathbf{d}'\in\mathcal{N}} a_{\mathbf{d}'}\langle Y(\mathbf{v}-\mathbf{d}'), Y(\mathbf{v}-\mathbf{d})\rangle \end{aligned} \tag{23}$$

Using $\langle Y(\mathbf{v}-\mathbf{d}'), Y(\mathbf{v}-\mathbf{d})\rangle = K_{\mathrm{NP}}(\mathbf{d}-\mathbf{d}')$ as defined in the previous subsection, we solve for the variables $\{a_{\mathbf{d}}\}$ from the $|\mathcal{N}|$ normal equations,

$$K_{\mathrm{NP}}(\mathbf{d}) = -\sum_{\mathbf{d}'\in\mathcal{N}} a_{\mathbf{d}'} K_{\mathrm{NP}}(\mathbf{d}-\mathbf{d}') \tag{24}$$

for all $\mathbf{d}\in\mathcal{N}$.

III. The Gaussian Process Transform (GPT)

Let $Y_1,\ldots,Y_M$ be zero mean random variables with covariance $r_{ij} = E[Y_iY_j] < \infty$. Since the covariance matrix $\mathbf{R} = [r_{ij}]$ is symmetric and positive definite, its eigen-decomposition can be expressed

$$\mathbf{R} = \Psi\mathbf{S}\Psi^T, \tag{25}$$

where Ψ is an orthogonal matrix whose columns are the right eigenvectors of $\mathbf{R}$ and $\mathbf{S} = \mathrm{diag}\{\sigma_k^2\}$ is a diagonal matrix of corresponding positive eigenvalues, or principal components, sorted as $\sigma_1^2 \geq \cdots \geq \sigma_M^2 > 0$. $\Psi^T\mathbf{y}$ is the KLT of the vector $\mathbf{y} = (Y_1,\ldots,Y_M)^T$.

Different covariance matrices $\mathbf{R}$ result in different transforms. Though they are all KLTs, they are distinguishable by how $\mathbf{R}$ is obtained.

In this paper, we obtain $\mathbf{R} = [r_{ij}]$ by assuming that $Y_i = Y(\mathbf{v}_i)$, $i = 1,\ldots,M$, are samples of a zero-mean GP. In particular, we let

$$r_{ij} = E[Y(\mathbf{v}_i)Y(\mathbf{v}_j)] = R(\mathbf{v}_i - \mathbf{v}_j), \tag{26}$$

where $R(\mathbf{d})$ is the covariance function of the GP, as modeled for example in Section II. As before, we let Ψ denote the matrix of eigenvectors of $\mathbf{R}$.

We call Ψ^T a *Gaussian Process Transform* (GPT) in recognition of the fact that the covariance matrix $\mathbf{R} = [E[Y(\mathbf{v}_i)Y(\mathbf{v}_j)]]$ on which it is based is derived from the covariance function defining a stationary GP, whose elements $Y(\mathbf{v}_i)$ and $Y(\mathbf{v}_j)$ may be considered samples at respective locations $\mathbf{v}_i$ and $\mathbf{v}_j$. The transform itself depends on how the samples are embedded in Euclidean space.

A. Relation to the Graph Fourier Transform

It is important to understand the close relation of the GPT to the Graph Fourier Transform (GFT) [23]. The GFT is defined as follows. Let $\mathcal{G} = (\mathcal{V},\mathcal{E})$ be an *undirected* graph, where $\mathcal{V} = \{v_1,\ldots,v_M\}$ is a finite set of nodes and $\mathcal{E} = \{(v_i,v_j)\}$ is a set of edges between nodes in $\mathcal{V}$. (A graph is undirected, also called bi-directed, if $(v_i,v_j)\in\mathcal{E}$ whenever $(v_j,v_i)\in\mathcal{E}$ [24].) Let $(\mathcal{G},\mathbf{W})$ be a *weighted* undirected graph, where $\mathbf{W} = [w_{ij}]$ is a symmetric non-negative $M\times M$ matrix of weights such that

$$w_{ij} > 0 \text{ if } (v_i,v_j)\in\mathcal{E} \text{ and } w_{ij} = 0 \text{ otherwise.} \tag{27}$$

$\mathbf{W}$ defines the *neighborhood structure* of $\mathcal{V}$ in the graph. Let $\mathbf{D} = [d_{ij}]$ be a diagonal matrix such that $d_{ii} = w_{ii} + \sum_j w_{ij}$ for all i. The *Graph Laplacian* of the weighted graph $(\mathcal{G},\mathbf{W})$ is defined

$$\mathbf{L} = \mathbf{D} - \mathbf{W}. \tag{28}$$

Let $\mathbf{L} = \Psi^T\Lambda\Psi$ be the eigen-decomposition of $\mathbf{L}$, where Λ is the diagonal matrix of eigenvalues and Ψ is the matrix whose columns are the corresponding right eigenvectors. The GFT is the linear transform from $\mathbb{R}^M$ to $\mathbb{R}^M$ represented by the matrix Ψ^T.

We next remark that the GFT is the KLT of a corresponding GMRF. A GMRF is a finite collection of Gaussian random variables whose joint distribution has a covariance structure given by a weighted undirected graph. Specifically, a random vector $\mathbf{y} = (Y(v_1),\cdots,Y(v_M))^T$ is called a GMRF with respect to the undirected graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$ with mean vector μ and a symmetric positive definite precision matrix $\mathbf{Q} = [q_{ij}]$ if and only if its density has the form [25]:

$$p(\mathbf{y}) = (2\pi)^{-\frac{M}{2}}|\mathbf{Q}|^{\frac{1}{2}}\exp\Big(-\frac{1}{2}(\mathbf{y}-\mu)^T\mathbf{Q}(\mathbf{y}-\mu)\Big), \tag{29}$$

and $q_{ij}\neq 0 \Leftrightarrow (v_i,v_j)\in\mathcal{E}$ for all $i\neq j$. (30)

From the above definition, it is clear that a GMRF $\mathbf{y}$ is a multivariate Gaussian distribution with mean vector μ whose covariance matrix $\mathbf{R}$ is the inverse of the precision matrix $\mathbf{Q}$ [21].

It is shown in [26, Sec. 2.1] that there is a one-to-one mapping from the set of symmetric non-negative weight matrices $\mathbf{W}$ satisfying (27) to the set of symmetric positive semi-definite precision matrices $\mathbf{Q}$ satisfying (30), through the mapping

$$q_{ij} = -w_{ij}, \text{for all } i\neq j, \tag{31}$$

$$q_{ii} = \sum_{j=1}^{n} w_{ij}, \text{for all } i, \tag{32}$$

and its inverse

$$w_{ij} = -q_{ij}, \text{for all } i \neq j, \tag{33}$$

$$w_{ii} = \sum_{j=1}^{n} q_{ij}, \text{for all } i. \tag{34}$$

It can be shown that $\mathbf{Q}$ is positive semi-definite if $\mathbf{W}$ is non-negative, and furthermore that $\mathbf{Q}$ is positive definite if $\mathbf{W}$ has at least one self-loop (i.e., $w_{ii} > 0$ for some i) in every connected component [26], [27]. In this paper, for simplicity we deal with only the case where $\mathbf{Q}$ is positive definite. The case where $\mathbf{Q}$ is singular requires more care but results in qualitatively similar conclusions. For details see [26].

It can be seen that every weighted graph $(\mathcal{G}, \mathbf{W})$ corresponds uniquely to a GMRF with zero mean and precision matrix $\mathbf{Q}$ given by (31)-(32). Moreover, it is easy to verify from (28) and (31)-(32) that

$$\mathbf{Q} = \mathbf{L} \tag{35}$$

and therefore $\mathbf{R} = \mathbf{Q}^{-1} = \Psi\Lambda^{-1}\Psi^T$. Hence $\mathbf{S} = \Lambda^{-1}$ is the diagonal matrix of eigenvalues of $\mathbf{R}$, and Ψ is the matrix whose columns are the corresponding eigenvectors. Thus the GFT Ψ^T is the KLT of the GMRF.

For a wide class of applications of Graph Signal Processing, the nodes $\mathbf{v}_1, \ldots, \mathbf{v}_M$ of $\mathcal{V}$ are embedded in a Euclidean domain $\mathbb{R}^N$. (Here we use boldface for the nodes to indicate that they are vectors in $\mathbb{R}^N$.) It is common in this case for the neighborhood structure of $\mathcal{V}$ to be inherited from the neighborhood structure of the containing domain. That is, $w_{ij} = W(\mathbf{v}_i - \mathbf{v}_j)$ for some weight function $W(\mathbf{d})$.

Figure 3 illustrates the relationship between the GPT and GFT. In the center column, there are three ways to specify a stationary GP $\{Y(\mathbf{v})\}$: through the covariance function $R(\mathbf{d})$, the precision function $Q(\mathbf{d})$, or the weight function $W(\mathbf{d})$. $R(\mathbf{d})$ and $Q(\mathbf{d})$ are equivalent via (10)-(11) (assuming the power spectral density is bounded away from zero over the cube $[-\pi, \pi]^3$), while $Q(\mathbf{d})$ and $W(\mathbf{d})$ are equivalent via (13) and its inverse

$$W(\mathbf{d}) = \begin{cases} -Q(\mathbf{d}) & \text{for } \mathbf{d} \neq 0 \\ \sum_{\mathbf{d}'} Q(\mathbf{d}') & \text{for } \mathbf{d} = 0. \end{cases} \tag{36}$$

The functions $R(\mathbf{d})$, $Q(\mathbf{d})$, and $W(\mathbf{d})$ induce what could be regarded as infinite block Toeplitz matrices $\mathbf{R}_\infty$, $\mathbf{Q}_\infty$, and $\mathbf{W}_\infty$, respectively, having row and column indices $\mathbf{v}$ and $\mathbf{v}'$ and elements at $(\mathbf{v}, \mathbf{v}')$ given by $R(\mathbf{v} - \mathbf{v}')$, $Q(\mathbf{v} - \mathbf{v}')$, and $W(\mathbf{v} - \mathbf{v}')$. Given a finite collection of points $\mathbf{v}_1, \ldots, \mathbf{v}_M$, the infinite matrices $\mathbf{R}_\infty$ and $\mathbf{W}_\infty$ can be restricted to the finite matrices $\mathbf{R}_M^{process} = [R(\mathbf{v}_i - \mathbf{v}_j)]$ and $\mathbf{W}_M = [W(\mathbf{v}_i - \mathbf{v}_j)]$ by considering only rows and columns in the collection $\mathbf{v}_1, \ldots, \mathbf{v}_M$. Then, $\mathbf{W}_M$ and $\mathbf{Q}_M^{graph}$ are equivalent via (31)-(34). Note that the finite matrices $\mathbf{R}_M^{process}$, $\mathbf{W}_M$, and $\mathbf{Q}_M^{graph}$ are block Toeplitz. However, the inverse of a finite Toeplitz matrix is generally not Toeplitz, and hence neither $\mathbf{R}_M^{graph}$ (the inverse of $\mathbf{Q}_M^{graph}$) nor $\mathbf{Q}_M^{process}$ the inverse of $\mathbf{R}_M^{process}$) are generally Toeplitz. Thus, $\mathbf{R}_M^{process}$ and $\mathbf{R}_M^{graph}$ cannot be equal (since one is Toeplitz and the other is not), and, hence, their KLTs cannot be equal. One is the GPT Ψ_{GPT}^T and the other is the GFT Ψ_{GFT}^T. These are KLTs of the random variables $Y(\mathbf{v}_1), \ldots, Y(\mathbf{v}_M)$.

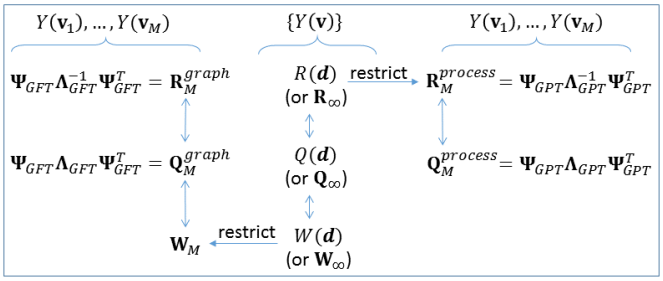


Fig. 3. Relationship between GPT and GFT.

IV. Transform Coding With the GPT

We now wish to use the GPT for coding voxelized point clouds. We segment the $L \times L \times L$ voxel grid containing the VPC into blocks of $N \times N \times N$ voxels. Each block may have anywhere between 0 and N^3 occupied voxels. We are concerned only with *occupied blocks*, that is, blocks with one or more occupied voxels. For a given occupied block, let $M \in \{1, \ldots, N^3\}$ be the number of occupied voxels in the block and let a voxel v_i $(0 \leq i < M)$ within the block have a 3D integer position $\mathbf{v}_i = (x_i, y_i, z_i)$ and a color signal c_i, where c_i can be any of Y_i, U_i or V_i with its global mean removed. We assume the covariance $r_{ij} = E[c_i c_j]$ between the colors of any two voxels in the block (v_i and v_j, $0 \leq i, j < M$) is given by one of the covariance matrices

$$\mathbf{R}_M^{process} = \Psi_{GPT}\mathbf{S}_{GPT}\Psi_{GPT}^T \tag{37}$$

$$\mathbf{R}_M^{graph} = \Psi_{GFT}\mathbf{S}_{GFT}\Psi_{GFT}^T \tag{38}$$

where in turn $\mathbf{R}_M^{process}$ is determined by one of the first two methods in the previous section (NP or OU) and $\mathbf{R}_M^{graph}$ is determined by one of the second two (ID or AR). In general, let $\mathbf{R}_{cc} = [r_{ij}]$ denote the covariance matrix and let Ψ denote the matrix of eigenvectors of $\mathbf{R}_{cc}$. If $\mathbf{c} = [c_i]$ is an $M \times 1$ vector of colors in the block, then

$$\mathbf{f} = \Psi^T\mathbf{c} \tag{39}$$

is the $M \times 1$ vector of its transform coefficients, whose $M \times M$ covariance matrix is given by

$$\mathbf{R}_{ff} = E[\mathbf{f}\,\mathbf{f}^T] = \Psi^T\mathbf{R}_{cc}\Psi = \mathbf{S}, \tag{40}$$

where $\mathbf{S} = \text{diag}\{s_i\}$. Thus $s_i = \sigma_{f(i)}^2$ is the nominal variance of each transform coefficient.

We quantize the transform coefficients using a uniform scalar quantizer with stepsize Q, and entropy code the quantized coefficient. This, however, is not sufficient for the decoder to reconstruct the color vector $\mathbf{c}$. We need also to convey the transform Ψ^T for each cube.

We next examine what information about the transform needs to be conveyed to the decoder for each of models described in the previous section. We begin by assuming that the geometry information $[\mathbf{v}_i]$ has already been encoded and conveyed to the decoder. The geometry is of interest in its

own right and must be conveyed for other reasons. How the geometry is encoded is outside the scope of this paper.

1) Non-Parametric GPT (NP-GPT): In the non-parametric covariance model, $r_{ij} = R_{NP}(d_{ij})$, where $d_{ij} = |\mathbf{v}_i - \mathbf{v}_j|$ is the Euclidean distance between voxels v_i and v_j. In order for the decoder to reconstruct Ψ and $\{s_k\}$ (from the r_{ij}), we convey $R_{NP}(d)$ to the receiver. The distance between voxel centers in the $N \times N \times N$ block can be no greater than $d_{\max} = \lceil (N-1)\sqrt{3} \rceil$. Thus it suffices to convey $R_{NP}(d)$ only for d between 0 and $d_{\max}$. In our tests, it was even sufficient to convey values of $R_{NP}(d)$ for values of d in steps of 0.5 from 0 to $d_{\max}$, and to interpolate the intermediate values from those. Thus, for $N = 8$, we convey only 25 values of $R_{NP}(d)$ to the decoder, which fit within a hundred bytes without coding. Once the decoder recovers $R_{NP}(d)$ for $d \in [0, d_{\max}]$, it assembles, for each block, $\mathbf{R} = [R_{NP}(d_{ij})]$ and performs an eigen-decomposition on $\mathbf{R}$ to obtain Ψ and $\{s_k\}$.

2) Ornstein-Uhlenbeck GPT (OU-GPT): In the OU model, $r_{ij} = R_{OU}(d_{ij}) = \sigma^2 \rho^{d_{ij}}$. The scale factor σ^2 can be ignored, as it does not affect the GPT. Thus we convey only ρ to the decoder, requiring only four bytes without coding. Once the decoder recovers ρ, it forms the covariance matrix $\mathbf{R} = [\rho^{d_{ij}}]$, where $d_{ij} = |\mathbf{v}_i - \mathbf{v}_j|$, and performs the eigen-decomposition of $\mathbf{R}$ to determine the GPT.

3) Inverse Distance GFT (ID-GFT): The GFT is given by weights on the graph of nodes corresponding to occupied voxels, such that the weight of the edge between voxels at $\mathbf{v}_i$ and $\mathbf{v}_j$ is w_{ij}. We use weights determined by the inverse distance model (12) for various values of $d_{\max}$. The number of such weights is the number of possible distances d_{ij} satisfying $0 < d_{ij} \leq d_{\max}$, which for $d_{\max} = 1$ is one, for $d_{\max} = \sqrt{2}$ is two, and for $d_{\max} = \sqrt{3}$ is three. Again the necessary information fits into four bytes per weight without coding. The decoder calculates the weights and forms the adjacency matrix $\mathbf{W} = [w_{ij}]$, the diagonal matrix $\mathbf{D} = [\text{diag}(\sum_j w_{ij})]$, and the Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$. The covariance matrix $\mathbf{R}$ is the inverse of the Laplacian matrix and hence has the same eigen-structure: $\mathbf{R} = \Psi \mathbf{S} \Psi^T$ and $\mathbf{L} = \mathbf{R}^{-1} = \Psi \Lambda \Psi^T$, where $\Lambda = \mathbf{S}^{-1}$. For this reason the decoder can obtain the GPT from an eigen-decomposition of $\mathbf{L}$ directly.

4) Auto-Regressive GFT (AR-GFT): In the Auto-Regressive model, we convey the coefficients $\{a_{\mathbf{d}}\}$ of the conditional autoregression to the decoder. The coefficients are determined by solving the normal equations (24) involving K_{NP}, thereby reducing the number of coefficients to at most $|\mathcal{N}|$. The number of coefficients is actually substantially fewer than that, due to symmetry. For example, if $\mathcal{N} = \{\mathbf{d} : 0 < |\mathbf{d}| \leq d_{\max}\}$, the number of independent coefficients for $d_{\max} = 1$ is one, for $d_{\max} = \sqrt{2}$ is two, and for $d_{\max} = \sqrt{3}$ is three. These fit into four bytes per coefficient without coding. Once the decoder recovers the coefficients $\{a_{\mathbf{d}}\}$, it computes $Q(\mathbf{d})$ from the autocorrelation (20), determines $W(\mathbf{d})$ using (36), extracts the finite weight matrix $\mathbf{W}_M$, computes $\mathbf{Q}_M^{graph}$ using (33)-(32), and performs an eigen-decomposition on this submatrix to determine the GFT.

In terms of computation, in addition to the obvious requirement of performing an $M \times M$ eigen-decomposition for each $N \times N \times N$ block having M occupied voxels, the non-parametric and auto-regressive models require computation of the covariance function $R_{NP}(d)$. If there are N_0 occupied voxels in the entire VPC, computing $R_{NP}(d)$ involves a number of computations proportional to N_0^2, which can be quite demanding. The OU model requires determination of ρ, which in principle should be optimized, e.g., by maximizing the likelihood over the N_0 voxels. However, we have found in practice that performance is not sensitive to the exact value of ρ and a value of 0.95 works well across all our data sets. Hence the OU-GPT and the ID-GFT are much less complex than the NP-GPT and AR-GFT.

V. The Entropy Coder

We now turn our attention to the entropy coder for the transform coefficients. The tricky part here is that the number of transform coefficients in each block is variable. Moreover, even for blocks with the same number of transform coefficients, the coefficients may not correspond to the same set of "frequencies" or eigenvalues. Thus there is no natural grouping of coefficients with the same set of statistics, which is traditional for entropy coding. To solve the problem, we bin the coefficients by their eigenvalues λ into N_b bins. Then for each bin we calculate the standard deviation of the coefficients in that bin. These standard deviations are uniformly scalar quantized with stepsize Q_s and are entropy coded and transmitted as side information. For each bin, the quantized standard deviation is used as the parameter of a Laplacian distribution to model the distribution of the coefficients in the bin. The coefficients in the bin are uniformly scalar quantized with stepsize Q and are arithmetic coded according to a discrete distribution induced by the Laplacian distribution for the bin.

Detailed instructions for encoding the colors $\{c_i\}$ for all N_v occupied voxels in the voxelized point cloud are as follows:

1) Decide on encoder parameters Q, Q_s, N_b. Encode and transmit them.
2) Encode the geometry information $\{\mathbf{v}_i\}$ using, for example, octrees.
3) From the geometry, derive the GPT Ψ^T and the corresponding eigenvalues $\{\lambda_i\}$ for each occupied block.
4) Transform all blocks and assemble all $\{\lambda_i\}$, and $\{f_i\}$, $0 \leq i < N_v$, for the whole cloud.
5) Find $\lambda_{max} = \max_i(\lambda_i)$.
6) Divide the λ_i into N_b bins as $\lambda_i^q = \text{round}(\lambda_i N_b / \lambda_{max})$.
7) Compute the standard deviation of the coefficients whose λ fall in each bin, i.e., $\eta_k = \sqrt{E[f_i^2 | \lambda_i^q = k]}$.
8) Quantize all η_k, $0 \leq k < N_b$, as η_k^q, e.g. $\eta_k^q = \text{round}(\eta_k / Q_s)$.
9) Encode and transmit all η_k^q.
10) Reconstruct standard deviations as $\hat{\eta}_k$, e.g. $\hat{\eta}_k = \eta_k^q Q_s$.
11) Quantize all f_i, $0 \leq i < N_v$, as $f_i^q = \text{round}(f_i / Q)$.
12) Encode and transmit all f_i^q using an arithmetic coder with a model of a Laplacian distribution with standard deviation $\hat{\eta}_k$ as

$$p(x) = \frac{1}{\sqrt{2}\hat{\eta}_k} e^{-\frac{|x|\sqrt{2}}{\hat{\eta}_k}}, \tag{41}$$

such that the ℓ-th quantizer bin has probability

$$p_\ell = e^{-\frac{|\ell| Q\sqrt{2}}{\hat{\eta}_k}} \sinh\left(\frac{Q}{\sqrt{2}\hat{\eta}_k}\right), \tag{42}$$

while the central one has probability

$$p_0 = 1 - e^{-\frac{Q}{\sqrt{2}\hat{\eta}_k}}. \tag{43}$$

Detailed instructions for decoding the colors $\{c_i\}$ for all N_v occupied voxels in the voxelized point cloud are as follows:

1) Decode the parameters Q, Q_s, N_b.
2) Decode the geometry information $\{\mathbf{v}_i\}$.
3) From the geometry, derive the GPT Ψ^T and the corresponding eigenvalues $\{\lambda_i\}$ for each occupied block.
4) Decode all N_b quantized values of η_k^q.
5) Reconstruct all standard deviations $\hat{\eta}_k$ (e.g. $\hat{\eta}_k = \eta_k^q Q_s$), and with them build all the probability tables $\{p_\ell\}$.
6) Find $\lambda_{max} = \max_i(\lambda_i)$.
7) In order to decode the i-th coefficient, from λ_i, find $k = \text{round}(\lambda_i N_b/\lambda_{max})$.
8) Look-up the standard deviation $\hat{\eta}_k$ of the k-th bin.
9) Decode f_i^q with an arithmetic coder with the previously constructed probability table for $\hat{\eta}_k$.
10) Reconstruct all coefficients $\hat{f}_i = f_i^q Q$.
11) Inverse transform all blocks to obtain $\{\hat{c}_i\}$.

We have found it sufficient to use $Q_s = 1$ for large η_k, with finer steps for small deviations like $\eta_k < 2$. One can use smaller Q_s but we used a coarse floating point representation spending 20 bits per sample. We also found $N_b = 60$ to be adequate such that 3600 bits are used for encoding all η_k^q for the 3 color components.

The proposed entropy coder uses arithmetic coding and Laplacian models just as in [18] and [19]. However, in [18], each quantized GFT coefficient is entropy coded using a variance inversely proportional to its corresponding eigenvalue, while in [19], the RAHT coefficients are bucketed according to their "weights" (the number of points contributing to the coefficients), and for each bucket the actual variance of its coefficients is encoded and transmitted as side information before entropy encoding the coefficients using that variance. This was shown to be more effective than using the variances predicted by the eigenvalues. In contrast, in this paper, we use a third approach. In the absence of "weights" for the coefficients, the coefficients are bucketed according to their corresponding eigenvalues, and for each bucket the actual variance of the coefficients is explicitly conveyed to the receiver.

VI. Experimental Results

We evaluate the proposed GPT and the entropy coding over point cloud colors of five data sets, known as *Man*, *Andrew*, *Phil*, *Ricardo*, and *Sarah*. *Man* is shown in Fig. 1; the rest are shown in Fig. 4. The data sets are point clouds each corresponding to one frame in a sequence of point clouds of the same name.[1] For these point clouds, the numbers of occupied voxels N_v are 223617, 286934, 325077, 207077, and 301336, respectively. Point cloud *Man* is a full body captured as a mesh using 24 RGBD pods in a non-realtime capture system [28], then voxelized into a point cloud. The other point clouds are front-half bodies captured directly into voxels using 4 RGBD pods in a realtime telepresence system [2]. All point clouds are represented by voxels in an $L \times L \times L$ integer grid with $L = 512$. We break the $L \times L \times L$ grid into smaller $N \times N \times N$ blocks with $N = 8$. This value was decided upon comparing against values of $N = 4$ and $N = 16$ in simple tests. Larger blocks take even longer to compute with no evidence of significant advantage, while smaller blocks have slightly inferior performance. An adaptive block size algorithm seems to be an interesting alternative, left for future studies. With $N = 8$, the number of occupied voxels in each occupied block may vary from 1 to 512. Nevertheless, we have found a mean of 58 occupied voxels per block, across all the point clouds tested.

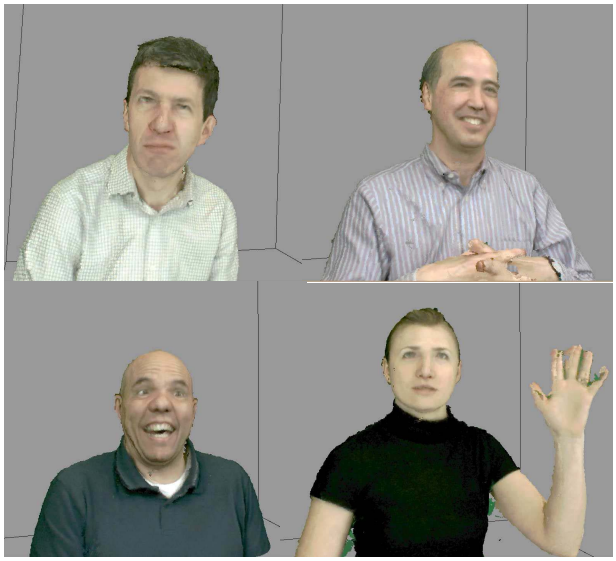


Fig. 4. Renderings of *Andrew* (top left), *Phil* (top right), *Ricardo* (bottom left), and *Sarah* (bottom right).

In our tests, we opted for an unassuming Matlab® implementation that takes tens of seconds to compute each frame, with modern CPUs. This is no parameter for a real-time implementation, wherein GPU programmers need to make the encoding to fit within 1/30 of a second.

The non-parametric covariance functions $K_{NP}(d)$ measured for each data set are shown in Fig. 5. The covariance functions appear to be diverse, both in slope and curvature, which is an indication of how hard it may be for any model to work well across all point clouds.

Our experiments were, then, divided into two parts. Firstly, we evaluated the potential compression of all the 4 GPTs (NP-GPT, OU-GPT, ID-GFT, AR-GFT) using traditional theoretical criteria such as coding gain and energy compaction. These tests are independent of the quantization and entropy coding schemes. Energy compaction is measured by ordering all coefficients according to their eigenvalues. Eigenvalues are associated with coefficients via the eigenvectors used to obtain the coefficients. The larger the eigenvalue of $\mathbf{R}$ the lower the

[1] frames 37, 622, 244, 39, and 234, respectively

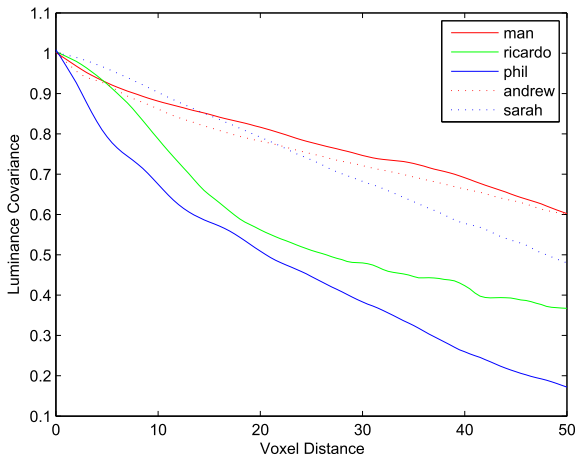


Fig. 5. Covariance function $K_{NP}(d)$ for each data set.

frequency associated to its corresponding eigenvector and we arrange eigenvalues in decreasing magnitude. The total energy in the transform coefficients $\{f_i\}$ is the same as that in the signal $\{c_i\}$, due to orthogonality of each transform. That is, $\sum_{i=1}^{N_v} f_i^2 = \sum_{i=1}^{N_v} c_i^2$.

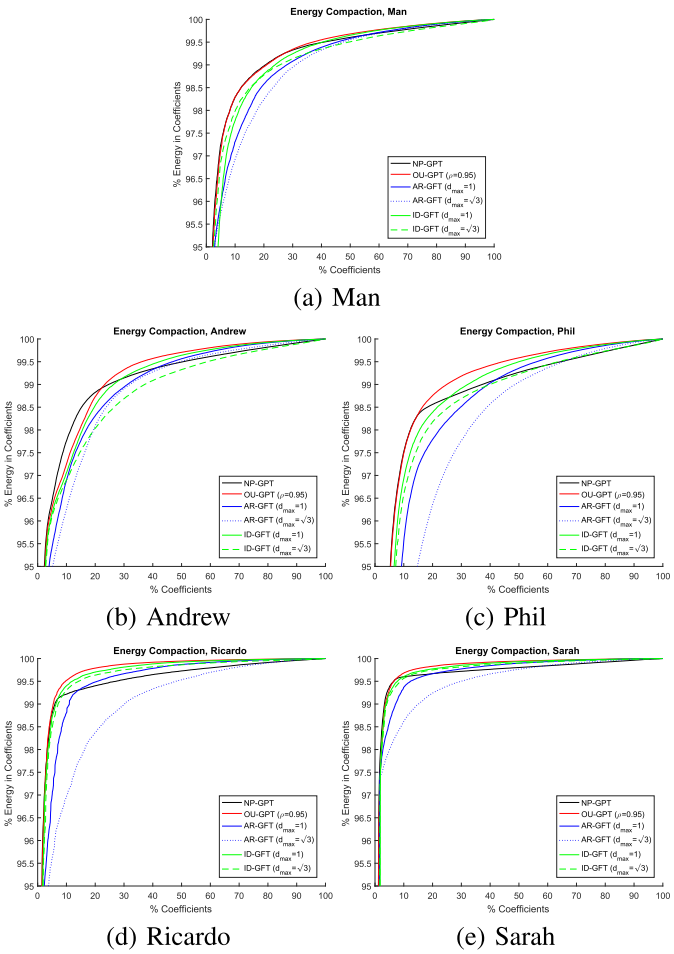
We only evaluate the luminance component Y_i, with the global mean removed, i.e., $c_i = Y_i - \mu$, where $\mu = (\sum_{i=1}^{N_v} Y_i)/N_v$. We tested for all images in our dataset. Figure 6 shows the fraction $\sum_{i=1}^{\alpha N_v} f_i^2 / \sum_{i=1}^{N_v} f_i^2$ of the total energy captured in the bottommost (lowest-frequency) fraction α of transform coefficients, as a function of the fraction α. It can be seen that the NP-GPT and OU-GPT generally have the highest energy compaction, with the OU-GPT edging out the NP-GPT in many cases. The AR-GFT ($d_{\max} = 1$ and $d_{\max} = \sqrt{3}$) does not perform as well as the others. One reason that the NP-GPT may sometimes perform worse than the OU-GPT is that the method of estimating $R_{NP}(\mathbf{d})$ described in II-A is not guaranteed to produce a non-negative definite covariance matrix $\mathbf{R}$; negative components must be truncated. This is due to the sparse nature of the data, which necessitates using different subsets of data to estimate $R_{NP}(\mathbf{d})$ at different $\mathbf{d}$, and also interpolation of $R_{NP}(d)$ at different $d = |\mathbf{d}|$. In any case, the difference between NP-GPT and OU-GPT is slight.

The transform coding gain [30] summarizes each energy compaction curve in a single number, as the ratio of the arithmetic to geometric means of the energy in each frequency band. Since our "frequency bands" are not fixed, we partition the coefficients, ordered (as above) by their eigenvalues, into contiguous bins with 100 coefficients per bin. The energy is computed for each bin, and the ratio of the arithmetic and geometric means of the bin energies is computed to estimate the transform coding gain. The estimate is insensitive to the number of coefficients per bin. Table I shows the coding gain in dB for the four transforms where both GFT types are evaluated with $d_{\max} = 1$ and $d_{\max} = \sqrt{3}$. Again the OU-GPT has a distinguished performance compared to the others.

Preliminary results of the energy compaction and transform coding gain were presented in [29]. However, the experimental results in [29] are incorrect. This paper corrects those results.

TABLE I
TRANSFORM CODING GAIN (dB)

Data set	NP-GPT	OU-GPT	AR-GFT(1)	AR-GFT($\sqrt{3}$)	ID-GFT(1)	ID-GFT($\sqrt{3}$)
Man	18.9	**19.3**	18.2	17.8	18.6	18.3
Andrew	17.5	**18.7**	17.2	16.6	17.9	16.3
Phil	16.2	**17.5**	15.4	13.6	16.6	15.5
Ricardo	23.7	**27.5**	24.5	20.7	26.5	25.5
Sarah	27.0	**29.8**	26.6	23.0	28.8	28.0
Average	20.6	**22.6**	20.4	18.4	21.7	20.7



(a) Man (b) Andrew (c) Phil (d) Ricardo (e) Sarah

Fig. 6. Energy compaction for NP-GPT, GMRF-GPT, OU-GPT, and GT for each data set. (a) Man. (b) Andrew. (c) Phil. (d) Ricardo. (e) Sarah.

The results in Fig. 5 and Table I should replace those in [29].

Beyond correcting some results in [29] and being more complete theoretically than [29], this paper focuses on using the Gaussian Process transforms for transform coding, which includes quantization and entropy coding.

Thus, in the second part of the experiments we examine the point cloud compression performance of the various transforms when paired with our entropy coder. Because the AR-GFT is much more complex to implement than the ID-GFT, and because of the above mentioned energy compaction tests, we decided to run compression tests comparing the NP-GPT, OU-GPT, and the ID-GFT for ($d_{max} = \sqrt{3}$). We also tested the RAHT coder of [19]. RAHT is an extremely

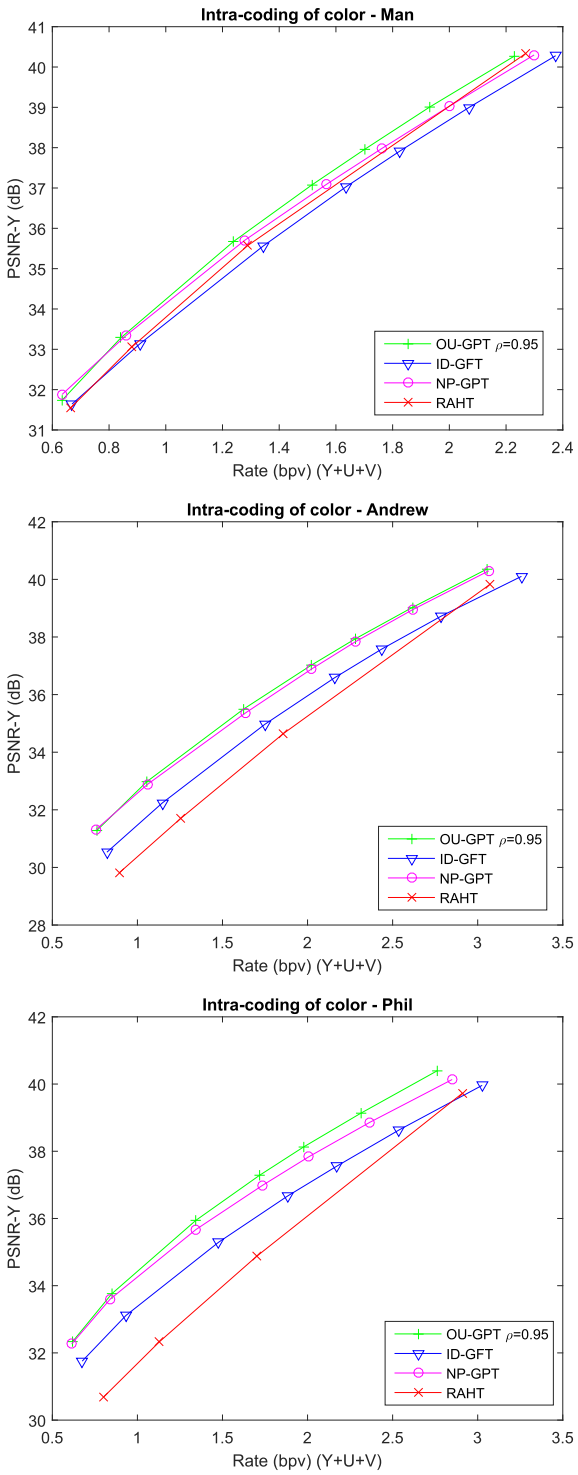


Fig. 7. Rate (bpv) vs. distortion (PSNR in dB for the Y component) for *Man*, *Andrew*, and *Phil*.

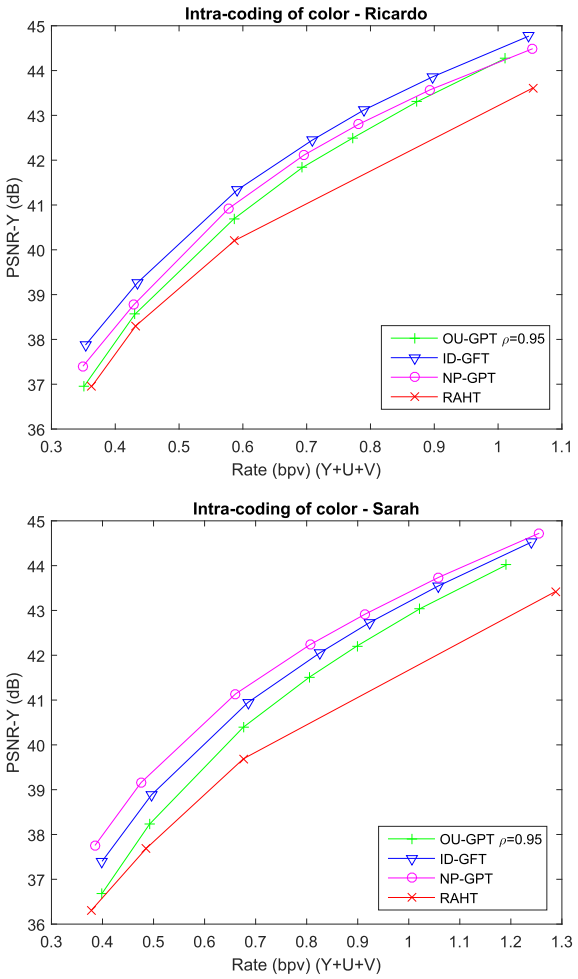


Fig. 8. Rate (bpv) vs. distortion (PSNR in dB for the Y component) for *Ricardo* and *Sarah*.

low-complexity transform, similar to the Haar Transform, suitable for point clouds, which, when paired with an entropy coder similar to the one in this paper, has been shown to match the performance of the graph transform and entropy coder that was the state-of-the-art prior to [19]. The side information required for the transform parameters, if any, is included in the bit rate. The bit rate accounts for all the three color components (Y, U and V) and is given in bits per occupied voxel (bpv).

Figures 7 and 8 show the distortion-rate performances of all transform coders on the five data sets. The GPT (and GFT) with the new entropy coder usually outperforms RAHT. It is not surprising that RAHT is outperformed by the other transforms, as it is a very low computational complexity coder based on the Haar Transform, and it is not adaptive to the signal statistics. Moreover, it is based on a different entropy coder. OU-GPT seems to perform better on the data sets whose covariance functions conform to an exponential 0.95^n model. The better the exponential fit to the covariance function, the closer the performance of OU-GPT to that of NP-GPT. Only in one case the GFT was the best, compared to the GPTs. NP-GPT is better than ID-GFT in 4 out of 5 cases. The drawback of NP-GPT is the extra computation necessary to compute the covariance function $K_{NP}(d)$. Comparing OU-GPT with ID-GFT is a 3-to-2 split decision in favor of OU-GPT. Overall, NP-GPT is in all cases either the best or second best and if we were to rank them somehow, despite the non-unanimous results, the NP-GPT may come up on top. Hence, we are confident to say that GPT (in its NP- or OU- forms) may be the new state-of-the-art for compressing point cloud attributes.

VII. Conclusions

We presented an approach to compress the colors or other attributes of static 3D point clouds based on transform coding, using what we call a *Gaussian Process Transform* or GPT. The GPT is the KLT of the points as if they were samples of a 3D zero-mean Gaussian process. We studied four models for representing the covariance function of the Gaussian process, and discussed the connection between GPT and GFT (Graph Fourier Transform). In particular we discussed in detail two GPT models (the non-parametric or NP model, which is based on direct measurement of the covariances, and the Ornstein-Uhlenbeck or OU model) and two GFT models (inverse-distance or ID, and autoregressive or AR). All models require transmitting a small amount of side information, representing the model, to the decoder. For entropy coding the quantized transform coefficients, we bin the coefficients according to their corresponding eigenvalues, and use arithmetic coding driven by models of Laplacian distributions with different variances for each bin. The variances are sent as side information to the decoder.

Experimental results show that the NP-GPT and OU-GPT outperform both the ID-GFT and AR-GFT, in terms of energy compaction, transform coding gain, and distortion-rate performance when using the proposed entropy coder. The ID-GFT was the previous state-of-the-art in compression of voxelized point clouds. One hypothesis as to why the NP-GPT and OU-GPT outperform the ID-GFT is that the former are matched to the statistics of the data, whereas the ID-GFT is based on the Laplacian of a graph whose vertices are connected to each other based only on Euclidean distance. However, we introduced the AR-GFT to show that matching the weights of the graph transform to the data by maximizing the pseudo-likelihood does not improve the performance of the GFT.

Thus, transform coding using the GPT (in its OU- or NP- forms) with the proposed entropy coder is the new state-of-the-art in the compression of voxelized point clouds. The method may also be applied to the processing of arbitrary signals on point clouds in higher dimensional spaces.

Acknowledgment

The authors thank Charles Loop and Qin Cai of the Microsoft I3D group for providing the *Andrew*, *Phil*, *Ricardo*, and *Sarah* sequences, and thank the Microsoft HCap group for providing the *Man* sequence.

References

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

[2] C. Loop, C. Zhang, and Z. Zhang, "Real-time high-resolution sparse voxelization with application to image-based modeling," in *Proc. High-Perform. Graph. Conf.*, Jul. 2013, pp. 73–79.

[3] O. Devillers and P.-M. Gandoin, "Geometric compression for interactive transmission," in *Proc. Vis.*, Oct. 2000, pp. 319–326.

[4] X. Gu, S. J. Gortler, and H. Hoppe, "Geometry images," in *Proc. Annu. Conf. Comput. Graph. Interact. Techn.*, Jul. 2002, pp. 355–361.

[5] H. M. Briceno, P. V. Sander, L. McMillan, S. Gortler, and H. Hoppe, "Geometry videos: A new representation for 3D animations," in *Proc. ACM SIGGRAPH/Eurograph. Symp. Comput. Animation*, Jul. 2003, pp. 136–146.

[6] S. Gupta, K. Sengupta, and A. Kassim, "Registration and partitioning-based compression of 3-D dynamic data," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 11, pp. 1144–1155, Nov. 2003.

[7] T. Ochotta and D. Saupe, "Compression of point-based 3D models by shape-adaptive wavelet coding of multi-height fields," in *Proc. Eurograph. Symp. Conf. Point-Based Graph.*, 2004, pp. 103–112.

[8] H. Habe, Y. Katsura, and T. Matsuyama, "Skin-off: Representation and compression scheme for 3D video," in *Proc. Picture Coding Symp.*, 2004, pp. 301–306.

[9] J. Peng, C.-S. Kim, and C.-C. J. Kuo, "Technologies for 3D mesh compression: A survey," *J. Vis. Commun. Image Represent.*, vol. 16, no. 6, pp. 688–733, Dec. 2005.

[10] S.-R. Han, T. Yamasaki, and K. Aizawa, "Time-varying mesh compression using an extended block matching algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 11, pp. 1506–1518, Nov. 2007.

[11] L. Váša and V. Skala, "Geometry-driven local neighbourhood based predictors for dynamic mesh compression," *Comput. Graph. Forum*, vol. 29, no. 6, pp. 1921–1933, Sep. 2010.

[12] H. Q. Nguyen, P. A. Chou, and Y. Chen, "Compression of human body sequences using graph wavelet filter banks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2014, pp. 6152–6156.

[13] R. Mekuria, M. Sanna, S. Asioli, E. Izquierdo, D. C. A. Bulterman, and P. Cesar, "A 3D tele-immersion system based on live captured mesh geometry," in *Proc. ACM Multimedia Syst. Conf.*, Oslo, Norway, Feb. 2013, pp. 24–35.

[14] J. Hou, L. Chau, N. Magnenat-Thalmann, and Y. He, "Compressing 3-D human motions via keyframe-based geometry videos," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 1, pp. 51–62, Jan. 2015.

[15] R. Schnabel and R. Klein, "Octree-based point-cloud compression," in *Proc. Symp. Point-Based Graph.*, Jul. 2006, pp. 111–120.

[16] Y. Huang, J. Peng, C.-C. J. Kuo, and M. Gopi, "A generic scheme for progressive point cloud coding," *IEEE Trans. Vis. Comput. Graphics*, vol. 14, no. 2, pp. 440–453, Mar./Apr. 2008.

[17] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 778–785.

[18] C. Zhang, D. Florêncio, and C. Loop, "Point cloud attribute compression with graph transform," in *Proc. IEEE Int. Conf. Image Process.*, Paris, France, Oct. 2014, pp. 2066–2070.

[19] R. L. D. Queiroz and P. A. Chou, "Compression of 3D point clouds using a region-adaptive hierarchical transform," *IEEE Trans. Image Process.*, vol. 25, no. 8, pp. 3947–3956, Aug. 2016.

[20] D. Thanou, P. A. Chou, and P. Frossard, "Graph-based motion estimation and compensation for dynamic 3D point cloud compression," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2015, pp. 3235–3239.

[21] Y. Dodge, *The Oxford Dictionary of Statistical Terms*. London, U.K.: Oxford Univ. Press, 2003.

[22] J. D. Markel and A. H. Gray, *Linear Prediction of Speech*. Berlin, Germany: Springer-Verlag, 1976.

[23] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[24] J. Edmonds and E. L. Johnson, "Matching: A well-solved class of integer linear programs," in *Combinatorial Optimization—Eureka, You Shrink!* (Lecture Notes in Computer Science), vol. 2570. Jan. 2003, pp. 27–30.

[25] H. Rue and L. Held, *Gaussian Markov Random Fields: Theory and Applications*. Boston, MA, USA: Chapman, 2005.

[26] C. Zhang, D. Florencio, and P. A. Chou, "Graph signal processing—A probabilistic framework," Microsoft Res., Redmond, WA, USA, Tech. Rep. MSR-TR-2015-31, Apr. 2015.

[27] F. Dorfler and F. Bullo, "Kron reduction of graphs with applications to electrical networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 1, pp. 150–163, Jan. 2013.

[28] A. Collet *et al.*, "High-quality streamable free-viewpoint video," *ACM Trans. Graph.*, vol. 34, no. 4, Aug. 2015, Art. no. 69.

[29] P. A. Chou and R. L. de Queiroz, "Gaussian process transforms," in *Proc. IEEE Int. Conf. Image Process.*, Phoenix, AZ, USA, Sep. 2016, pp. 1524–1528.

[30] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Norwell, MA, USA: Kluwer, 1992.

Ricardo L. de Queiroz (F'–) received the Engineer degree from the Universidade de Brasilia, Brazil, in 1987, the M.Sc. degree from the 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.

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

He was a past elected member of the IEEE Signal Processing Society's Multimedia Signal Processing and the Image, Video and Multidimensional Signal Processing Technical Committees. He is an Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING and a past Editor for the *EURASIP Journal on Image and Video Processing*, the IEEE SIGNAL PROCESSING LETTERS, and the 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 the Rochester chapter of the IEEE Signal Processing Society, where he served as the Chair and organized the Western New York Image Processing Workshop since its inception until 2001. He helped organizing the IEEE SPS Chapters in Brazil and chaired the Brasilia IEEE SPS Chapter. He was the General Chair of ISCAS'2011, MMSP'2009, and SBrT'2012. He was also part of the organizing committee of ICIP'2002, ICIP'2012, ICIP'2014, and ICIP'2016. His research interests include image and video compression, multirate signal processing, and color imaging. He is a member of the Brazilian Telecommunications Society.

Philip A. Chou received the B.S.E. degree in electrical engineering and computer science from Princeton University, Princeton, NJ, USA, in 1980, and the M.S. degree in electrical engineering and computer science from the University of California at Berkeley, Berkeley, USA, in 1983, and the Ph.D. degree in electrical engineering from Stanford University in 1988. From 1988 to 1990, he was a member of Technical Staff with AT& T Bell Laboratories in Murray Hill, NJ, USA. From 1990 to 1996, he was a member of Research Staff with the Xerox Palo Alto Research Center in Palo Alto, CA, USA. In 1997, he was the Manager of the Compression Group, VXtreme, an Internet video startup in Mountain View, CA, USA, before it was acquired by Microsoft. From 1998 to 2016, he was a Principal Researcher with the Microsoft Research, in Redmond, Washington, DC, USA, managing the Communication and Collaboration Systems Research Group from 2004 to 2011. He served as a Consulting Associate Professor with Stanford University from 1994 to 1995, an Affiliate Associate Professor with the University of Washington from 1998 to 2009, and an Adjunct Professor with The Chinese University of Hong Kong since 2006. He is currently with 8i Labs., where he leads the effort to compress and communicate volumetric media for augmented and virtual reality.