PROGRESSIVE GRAPH-SIGNAL SAMPLING AND ENCODING FOR STATIC 3D GEOMETRY REPRESENTATION

Mingyuan Zhao [*], *Gene Cheung* [#], *Dinei Florencio* [$], *Xiangyang Ji* [*]

[*] Tsinghua University, [#] National Institute of Informatics, [$] Microsoft Research

ABSTRACT

Compression of arbitrary 3D geometry like a human figure in 3D space is challenging. Existing 3D representations like point cloud require encoding of input-specified 3D coordinates, resulting in a large overhead. In this paper, assuming that there exists an underlying smooth 2D manifold in 3D space that describes the geometric shape of a target object, we develop a new progressive 3D geometry representation that signal-adaptively identifies new samples on the manifold surface and encodes them efficiently as graph-signals. Specifically, at each iteration, using previous encoded samples in 3D space, the encoder and decoder first synchronously interpolate a continuous sampling kernel (a 3D mesh)—an approximation of the target surface. We next distribute new sample locations on the continuous kernel based on locally computed kernel curvatures, and compute the signed distances between sample locations and the target surface as sample values. Finally, we connect new discrete samples into a graph for graph-based transform coding of the sample values, which are transmitted to the decoder to refine 3D reconstruction. Experimental results show that our coding scheme outperforms an existing mesh-bsed approach significantly at the low-bitrate region for two different datasets.

Index Terms— 3D geometry compression, graph signal processing, progressive coding

1. INTRODUCTION

The advent of depth sensing technologies like Microsoft Kinect has enabled real-time capturing of 3D geometry of arbitrarily shaped objects like human figures in a 3D scene[1]. Efficient compression and transmission of captured 3D geometry to a receiver for viewpoint synthesis remains a technical challenge. Existing representations like point cloud require explicit encoding of a pre-specified set of 3D coordinates, which is expensive.

In this paper, assuming that there exists an underlying smooth 2D manifold in 3D space that describes the geometric surface of a target object—a fair assumption given that immersive visual communication typically involves animated subjects like humans—we propose a new progressive 3D geometry representation that iteratively identifies new samples on the manifold and encodes them efficiently as a *graph-signal* [1–3], in order to incrementally improve 3D reconstruction at the decoder. Specifically, at each iteration, we first construct a continuous sampling kernel by linearly interpolating previously coded samples; the constructed kernel (3D mesh) is an approximation of the target 3D surface. To refine the reconstruction, we distribute new sample locations on the continuous kernel based on locally computed curvatures—regions with large gradients are allocated samples. Because kernel curvatures are computed from previously coded samples, sample locations can be deduced at the decoder without explicit signaling. We then compute signed distances between sample locations on the kernel and the target 3D surface as the signal to be encoded.

This work is supported by the Microsoft Research CORE program.

[1]See Holoportation from Microsoft Research as an example of dynamic 3D construction using multiple depth cameras: https://www.youtube.com/watch?v=7d59O6cfaM0&t=14s

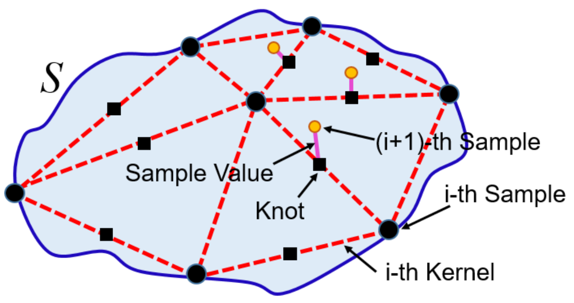


Fig. 1. Illustration of progressive graph-signal sampling & encoding.

As an illustration, in Fig. 1 samples of the i-th iteration (black circles) form a continuous kernel (connected planes denoted by the red lines) that is an approximation of the target surface $\mathcal{S}$. New sample locations called *knots* (squares) are introduced on the kernel surface, and the signed distances between knots and $\mathcal{S}$ are recorded as sample values. Sample values are encoded and transmitted to the decoder to reconstruct samples for the $(i+1)$-th iteration (green circles) that are on $\mathcal{S}$. Unlike previous representations that require coding of pre-specified 3D coordinates (three numbers per sample), our coding system chooses new sample locations freely on the continuous kernel and encodes only samples' signed distances (one number per sample), which is an easier task.

In our implementation, the allocated samples are connected using edges (with weights that reflect pairwise similarities) into a graph for efficient graph-signal compression using graph Fourier transform (GFT) [1]. Experimental results show that our progressive encoding scheme outperforms existing representations significantly at low bitrate regions for two different datasets.

The outline of the paper is as follows. We first discuss related work in Section 2. We overview our coding system in Section 3. In Section 4, we first study convergence of a simpler progressive sampling and coding scheme in 2D space, which motivates our kernel construction and sampling procecdures in 3D space. We describe our graph-signal coding strategy using GFT in Section 5. Finally, experimental results and conclusion are presented in Section 6 and 7, respectively.

2. RELATED WORK

There are numerous existing representations of 3D geometry captured from static natural scenes. In the image processing community, multiple viewpoint images captured by an array of closely spaced cameras were used [4]. However, it is not straightforward to entirely remove redundancies across neighboring views using standard image / video coding tools like HEVC [5]. Alternatively, point cloud—unstructured captured 3D points of an object—has fast become popular. Existing point cloud compression techniques attempt to logically structure the point set before compression, using kd-tree [6], octree [7, 8], and 2D images via projection from 3D to 2D [9] (called "geometry image" in [10]). As previously discussed, no matter what support structure a coding scheme is using, the 3D coordinate of each designated 3D point still requires explicit coding, which typically results in a large overhead.

There exists a vast literature on 3D mesh compression with novel techniques like mesh wavelet transforms (MWTs) [11, 12]; see surveys like [13] for details. There are also progressive mesh compression schemes [14] where points in the original mesh are divided into refinement batches and transmitted in sequence. Nonetheless, the *exact* 3D coordinate of each point still requires coding, which is expensive. In contrast, our scheme freely picks sampling locations on the continuous smooth manifold and encodes only signed distances, resulting in coding gain.

Recent advances in *graph signal processing* (GSP) [15] have led to the development of new coding tools such as graph Fourier transforms (GFT) [1, 2] and wavelets [3]. These tools have been used also for 3D data: compression of human body sequence [16], and compression of dynamic 3D point cloud [17, 18]. Unlike these works, our scheme combines manifold sampling and graph-signal compression into one unified framework for 3D representation.

3. SYSTEM OVERVIEW

We first overview our 3D geometry coding system. We assume that the input to our system is a continuous 3D surface specified by a 3D mesh—connected triangles in 3D space, where each triangle corner is defined by a triple denoting its Cartesian coordinate. A point cloud (collection of 3D points in space) captured by a depth sensing camera like MS Kinect can be converted to a 3D mesh easily using known tools such as MeshLab[2].

Given an input 3D mesh, we perform our proposed progressive graph-signal sampling and coding stretegy as follows. First, a small initial subset of the original set of 3D coordinates are selected and coded explicitly for transmission to the decoder. Because the size of this dataset is kept small, the coding cost is not expensive. We then draw triangles using the received 3D coordinates to build a continuous kernel. Given a continuous kernel, new discrete sample locations (called *knots* in the sequel) on the kernel are identified based on locally computed curvatures. For each knot, a normal vector is computed, and the distance between the knot and the target surface along the normal vector is recorded as a *sample value*. We connect the knots with edges, where an edge weight $w_{i,j}$, $0 < w_{i,j} \leq 1$, is computed based on geometric locations of knots i and j. Because knot locations are known at both encoder and decoder, no extra coding overhead is required to explicitly specify the graph. The collection of connected sample values now compose a graph-signal, which we can encode using any graph-based transform or wavelet tools in the GSP literature [1–3].

[2]http://www.meshlab.net/

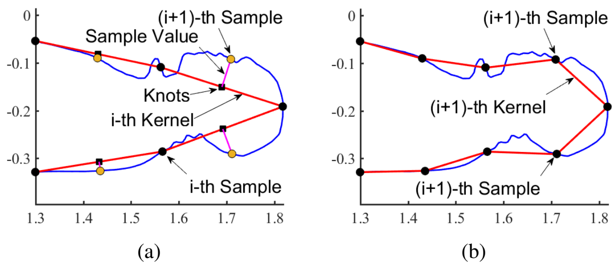


Fig. 2. (a) Sampling kernel at the i-th iteration; (b) Sampling kernel at the $(i+1)$-th iteration.

The crux in our framework resides in two crucial steps: i) how to construct a continuous kernel from existing samples, and ii) how to select new samples from a constructed continuous kernel. We describe these two steps next.

4. SAMPLING & KERNEL CONSTRUCTION

We first investigate the convergence of our progressive sampling approach by studying the simpler 2D case. Based on the insights we have developed, we describe our method to address the two aforementioned key issues.

4.1. Progressive 2D Sampling

We consider a progressive sampling process for a target planar polygon curve. Let the polygon curve be defined by its M vertices $\mathbf{P} = \{\mathbf{P}_1, \ldots, \mathbf{P}_M\}$, where each vertex $\mathbf{P}_i = (x_i, y_i)$ has a 2D Cartesian coordinate. We can write the curve $\mathbf{P}$ in parametric form: $\mathbf{P}(u)$, $u \in [0, 1]$, is a point along the curve $\mathbf{P}$, where $\mathbf{P}(0) = \mathbf{P}_1$ and $\mathbf{P}(1) = \mathbf{P}_M$.

Our goal is to show that a simple progressive sampling strategy to select samples $\mathbf{s}_j$ on curve $\mathbf{P}$ will result in a continuous sampling kernel $\mathbf{Q}$ (piecewise linear interpolation of samples $\mathbf{s}_i$) that converges to $\mathbf{P}$. The strategy is as follows: at each iteration i, we connect each pair of neighboring samples $\mathbf{s}_j$ and $\mathbf{s}_{j+1}$ with a straight line, and choose the midpoint of the line as the new sample location (called *knot*) $\mathbf{h} = (\mathbf{s}_i + \mathbf{s}_j)/2$. At knot $\mathbf{h}$, we identity a *normal vector* $\mathbf{n}$ based on the orientation of line segment from $\mathbf{s}_j$ to $\mathbf{s}_{j+1}$. We compute the *signed distance* between $\mathbf{h}$ and curve $\mathbf{P}$ along direction $\mathbf{n}$—sample value x—which is encoded and transmitted to the decoder to reconstruct sample $\mathbf{s}'$ on $\mathbf{P}$. At iteration $i + 1$, this new sample $\mathbf{s}'$ is inserted between previous samples $\mathbf{s}_j$ and $\mathbf{s}_{j+1}$, and the procedure repeats. An example is shown in Fig. 2, where the target curve is blue and the approximating continuous kernels are red. We see that new samples are inserted in the $(i + 1)$-th iteration relative to the i-th iteration, resulting in a kernel that is closer to the target curve.

We now sketch a proof to show that using this sampling strategy, continuous kernel $\mathbf{Q}$ will converge to the target $\mathbf{P}$. We first see that at each iteration i, insertion of new sample $\mathbf{s}'$ between previous sample pairs $\mathbf{s}_j$ and $\mathbf{s}_{j+1}$ means that the *curve distance* (Euclidean distance between two points on curve $\mathbf{P}$ along $\mathbf{P}$) between neighboring samples must strictly decrease. This is true because $\mathbf{s}'$ always divides the curve segment from $\mathbf{s}_j$ to $\mathbf{s}_{j+1}$ into two shorter segments.

Given that the curve distance between neighboring samples strictly decreases, there are only two possible cases for two neighboring samples when the number of iterations is sufficiently large: i)

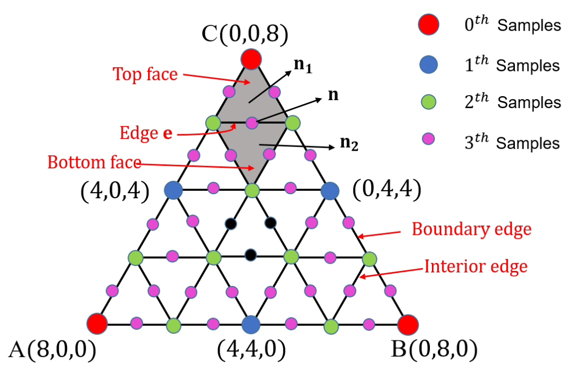


Fig. 3. Barycentric coordinate system, with n = 3.

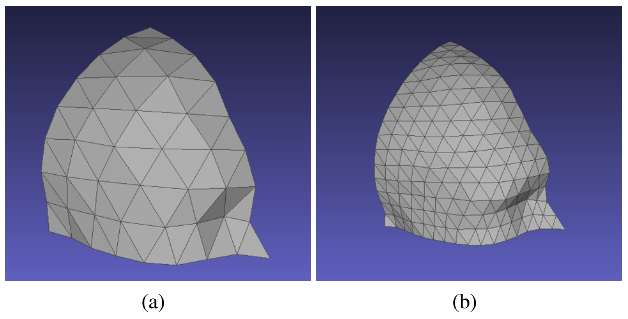


Fig. 4. (a) Sampling kernel in the i^{th} stage; (b) Sampling kernel in the $(i+1)^{th}$ stage

the two neighboring samples are on the same line segment in $\mathbf{P}$, ii) the two neighboring samples are on two neighboring line segments of $\mathbf{P}$. In the first case, the kernel $\mathbf{Q}$ has already converged to $\mathbf{P}$. In the second case, a straight line connecting the two neighboring samples and the portion of curve $\mathbf{P}$ denoted by the two samples form a local triangle. One can then easily show that the area of that triangle will monotonously decrease to zero as more samples are inserted in-between.

4.2. 3D Kernel Construction

Having developed insights into the sampling & reconstruction problem in 2D, we now turn to the more challenging problem in 3D. While in 2D we interpolate every pair of neighboring samples $\mathbf{s}_j$ and $\mathbf{s}_{j+1}$ with a straight line to construct a continuous 1D kernel (1D curve in 2D space), in 3D we identify every neighborhood of three samples and interpolate with a linear plane to construct a continuous 2D kernel (2D manifold in 3D space).

To ease discussion, we use a *Barycentric coordinate system*[3], in which the coordinate of a point of a simplex—in this paper a triangle—is computed as the center of mass. For example, see Fig. 3 where the coordinates of the corners of the large triangle are $\mathbf{A}$: $(2^n, 0, 0)$, $\mathbf{B}$: $(0, 2^n, 0)$, $\mathbf{C}$: $(0, 0, 2^n)$, where $n = 3$. If a point is inside this triangle, then it has positive coordinates. If all coordinates of a point are integers, we call this point a grid point. We can easily compute the coordinate of a grid point, which can be viewed as the weighted sum of mass of the corners. We can then index it by that coordinate.

Specifically, at each iteration i, we first assume that samples from the previous iteration collectively form a triangular mesh (not necessarily watertight). We traverse each edge $\mathbf{e}$ in the mesh with endpoints $\mathbf{p}_j$ and $\mathbf{p}_k$. We compute the midpoint $\mathbf{m} = (\mathbf{p}_j + \mathbf{p}_k)/2$, which is a potential knot to collect a new sample for this iteration. If edge $\mathbf{e}$ is in the mesh interior, then it is incident on two triangles with respective surface normals $\mathbf{n}_1$ and $\mathbf{n}_2$. We assign normal direction to knot $\mathbf{m}$ as $\bar{\mathbf{n}} = (\mathbf{n}_1 + \mathbf{n}_2)/2$. As an example, in Fig. 3 the edge $\mathbf{e}$ is incident to triangles *Top face* and *Bottom face* with respective normals $\mathbf{n}_1$ and $\mathbf{n}_2$. For $\mathbf{e}$, normal vector $\bar{\mathbf{n}}$ is computed.

If a sample at knot $\mathbf{m}$ is collected according to a local criteria (to be discussed in Section 4.3), then the signed distance between knot $\mathbf{m}$ and the target surface along direction $\bar{\mathbf{n}}$ is recorded as a sample value. The recorded sample values are coded as a graph-signal using GFT that is discussed in Section 5.

[3]https://en.wikipedia.org/wiki/Barycentric_coordinate_system

Because we do not assume a watertight mesh, there may be boundary edges in the mesh with only one incident triangle (see Fig. 3), and we cannot compute a normal direction $\bar{\mathbf{n}}$ for these edges in the same manner. To simplify this problem, we assume that the boundary of the original mesh is described by a planar curve that is coded separately, using which it would be easy to compute a normal for these boundary edges.

We construct a triangular mesh (and hence a continuous kernel) using the coded samples in each iteration as follows. As shown in Fig. 3, new samples (blue points $(4, 0, 4)$, $(0, 4, 4)$ and $(4, 4, 0)$) are midpoints of edges from the previous iteration (edges AB, BC and CA). If all samples on the midpoints are collected, we connect them to a triangle. If any sample is missing due to selection criteria described in Section 4.3, then we use the knot (edge midpoint) as a replacement to connect a triangle nonetheless. The idea is that a sample is not selected because the local kernel curvature is small, and hence the difference between the knot location and the actual manifold sample at the knot should be small. As an example, in Fig. 4 we see improvement of 3D surface reconstruction from iteration i to iteration $i + 1$.

4.3. 3D Samples Selection

If we collect samples for every identified knot as described in Section 4.2, then the number of samples will increase exponentially, resulting in a large coding cost. Instead, we determine if an identified knot should collect a new sample in a *signal-adaptive* manner based on a local curvature criteria. Specifically, we consider a triple of midpoints on three edges of a triangle at the same time. For example, see the three block dots in Fig. 3. For a given edge $\mathbf{e}_i$ of the triangle, we compute the normal difference $\mathbf{d}_i$ between the surface normals $\mathbf{n}_1$ and $\mathbf{n}_2$ of the two triangles incident on $\mathbf{e}_i$. Finally, we compute the sum of the absolute normal differences for the three edges: $\sum |\mathbf{d_i}|$. This sum reflects the local kernel smoothness at this triangle. If $\sum |\mathbf{d_i}|$ is larger than a threshold τ, then we allocate a sample for the midpoint of each edge in this triangle. Otherwise, it means that the local kernel at this triangle is sufficiently smooth, hence no more sampling is needed.

5. GRAPH-BASED SAMPLE ENCODING

5.1. Graph Construction

Given a set of selected new sample values in vector form $\mathbf{x}$, we first construct a graph $\mathcal{G}$ to connect them as nodes, so that the sample values together can be interpreted as a graph-signal. Specifically, we first connect each sample knot to its k nearest neighboring knots (k-nn) in Euclidean distance. That means each node (knot) is connected to at least k other nodes. For each edge connecting nodes i and j, we compute its edge weight $w_{i,j}$ using a Gaussian kernel:

$$w_{i,j} = \exp\left(-\frac{\|\mathbf{l}_i - \mathbf{l}_j\|_2^2}{\sigma^2}\right) \tag{1}$$

where $\mathbf{l}_i$ is the 3D coordinate of sample knot i and σ is a pre-chosen parameter. The idea is that edge weight $w_{i,j}$ should reflect the similarity between signal sample values x_i and x_j on the two connected nodes. Gaussian kernel is commonly used in the GSP literature to compute edge weights [1–3].

5.2. Graph Fourier Transform

Having define edges weights $w_{i,j}$, we can define an *adjacency matrix* $\mathbf{A}$ where $A_{i,j} = w_{i,j}$, and a diagonal *degree matrix* $\mathbf{D}$ where $D_{i,i} = \sum_j A_{i,j}$. The *combinatorial graph Laplacian* $\mathbf{L} = \mathbf{D} - \mathbf{A}$ can then be defined. Performing eigen-decomposition on $\mathbf{L}$ leads to a set of eigenvalues λ_i (interpreted as graph frequencies) and eigenvectors $\boldsymbol{\phi}_i$ (together compose a *graph Fourier transform* (GFT)) [15]. We can then compute GFT coefficients $\mathbf{a} = \mathbf{\Phi x}$, perform quantization (using a chosen quantization parameter) and entropy encoding for transmission of the sample values to the decoder.

At the decoder, the graph $\mathcal{G}$ can be first constructed based on knot locations, which are known from previous samples. From $\mathcal{G}$, Laplacian $\mathbf{L}$ and GFT $\mathbf{\Phi}$ can be computed, so that inverse GFT can be performed from quantized coefficients $\hat{\mathbf{a}}$: $\hat{\mathbf{x}} = \mathbf{\Phi}^{-1}\hat{\mathbf{a}}$. The recovered sample values $\hat{\mathbf{x}}$ can be placed at their respective knots to reconstruct this iteration of samples on the target surface.

For complexity reason, instead of performing eigen-decomposition for $\mathbf{\Phi}$ and computing $\mathbf{a} = \mathbf{\Phi x}$, one can choose a lifting implementation of GFT [19], or employ a graph wavelet instead [3], where the complexity in both cases is $O(n \log n)$.

6. EXPERIMENTS

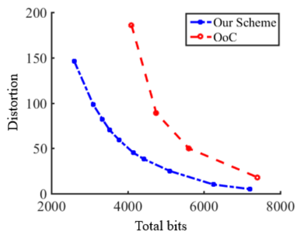
6.1. Experimental Setup

We test our scheme on two different datasets. The MIT dataset[4] provides mesh in .obj format, which include the 3D coordinate of each vertex and a list of triangle faces defined by vertices. A second dataset from [20] contains 15 different facial expressions, each of which has roughly 2000 vertices. We include both datasets for diversity, since the second dataset has density that is much higher than that of MIT's.

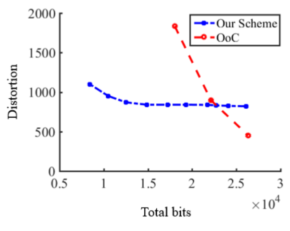
We execute our coding scheme (graph-signal sampling or `GSS`) as follows. We fix the number of coding layers (iterations) to 5. For each layer, we tune the threshold τ for signal-adaptive sampling for optimal performance. When coding GFT coefficients, we use different quantization parameters (QP) to induce different rate-distortion (RD) tradeoffs. We construct the convex hull of all data points generated using different combinations of parameters to discover the best performance points of our coding scheme.

For competitor, we choose a mesh coding scheme called Out-of-Core (`OoC`) [21]—a popular lossless mesh codec—which encodes the original mesh directly. `OoC` does not perform progressive coding. We execute `OoC` with different input bit-depths to induce different RD tradeoffs. In contrast, our progressive scheme `GSS` does not encode 3D coordinates directly, but places knots on continuous kernels from which signed distances to the mesh are computed and encoded. The distortion metric we employ is the method proposed in [22], which is commonly used in the mesh compression literature.

[4]http://people.csail.mit.edu/drdaniel/mesh_animation/



(a)



(b)

Fig. 5. (a) R-D curve of dateset1; (b) R-D curve of dataset2.

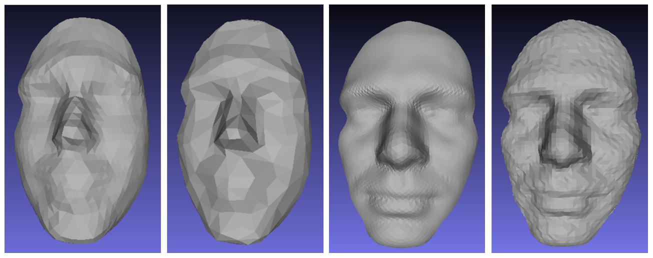

(a) (b)

Fig. 6. (a) Decoded meshes for dataset1; (b) Decoded mesh for dataset2. Left is `GSS` and right is `OoC`, and both encoded at roughly the same rate.

6.2. Experimental Results

The RD-curves for the two datasets are shown in Fig. 5. We observe that our proposed `GSS` outperforms `OoC` noticeably at the low-bitrate region for both datasets. At the high-bitrate region, because our scheme does not reproduce the exact position of each vertex, it is much harder for our proposal to reduce distortion to close to zero. However, as we can observe in Fig. 6, encoded 3D surfaces by `GSS` are actually very reasonable and visually pleasing; compared to 3D surfaces encoded by `OoC` at roughly the same bitrates, our surfaces are more natural and smooth. Thus, one future work is to identify a metric for evaluation that is more suitable for surface-wise comparison rather than point-wise comparison.

7. CONCLUSION

Previous 3D geometry representations like point cloud require explicit coding of a pre-defined discrete set of 3D coordinates, which is costly. In this paper, assuming that there exists a smooth underlying 2D manifold that describes the shape of a target object, we propose a new progressive representation, where, at each iteration, new samples on the continuous manifold are introduced in a signal-adaptive manner to refine previous 3D reconstruction. Each layer of samples are connected using edges (with weights that reflect inter-sample similarities) for efficient graph-based transform coding. Experimental results show that our representation outperforms previous coding schemes significantly at the low bitrate region.

8. REFERENCES

[1] W. Hu, G. Cheung, A. Ortega, and O. Au, "Multi-resolution graph Fourier transform for compression of piecewise smooth images," in *IEEE Transactions on Image Processing*, January 2015, vol. 24, no.1, pp. 419–433.

[2] W. Hu, G. Cheung, and A. Ortega, "Intra-prediction and generalized graph Fourier transform for image coding," in *IEEE Signal Processing Letters*, November 2015, vol. 22, no.11, pp. 1913–1917.

[3] S. Narang and A. Ortega, "Lifting based wavelet transforms on graphs," in *APSIPA ASC*, Sapporo, Japan, October 2009.

[4] T. Fujii, K. Mori, K. Takeda, K. Mase, M. Tanimoto, and Y. Suenaga, "Multipoint measuring system for video and sound—100 camera and microphone system," in *IEEE International Conference on Multimedia and Expo*, Toronto, Canada, July 2006.

[5] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," in *IEEE Transactions on Circuits and Systems for Video Technology*, December 2012, vol. 22, no.12.

[6] O. Devillers and P.-M. Gandoin, "Geometry compression for interactive transmission," in *IEEE Visualization 2000*, Salt Lake City, UT, October 2000.

[7] R. Schnabel and R. Klein, "Octree-based point-cloud compression," in *3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics*, Boston, MA, July 2006.

[8] Y. Huang, J. Peng, C.-C. J. Kuo, and M. Gopi, "A generic scheme for progressive point cloud coding," in *IEEE Transactions on Visualization and Computer Graphics*, March / April 2008, vol. 14, no.2, pp. 440–453.

[9] J.-K. Ahn, K.-Y. Lee, J.-Y. Sim, and C.-S. Kim, "Large-scale 3D point cloud compression using adaptive radial distance prediction in hybrid coordinate domains," in *IEEE Transactions on Selected Topics in Signal Processing*, April 2015, vol. 9, no. 3, pp. 422–434.

[10] X. Gu, S. Gortler, and H. Hoppe, "Geometry images," in *ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'02)*, San Antonio, TX, July 2002.

[11] I. Guskov, W. Sweldens, and P. Schröder, "Multiresolution signal processing for meshes," in *ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'99)*, Los Angeles, CA, August 1999.

[12] A. Khodakovsky, P. Schröder, and W. Sweldens, "Progressive geometry compression," in *ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'00)*, New Orleans, LA, July 2000.

[13] J. Peng, C.-S. Kim, and C.-C. J. Kuo, "Technologies for 3D mesh compressions: A survey," in *Journal of Visual Communication and Image Representation*, December 2005, vol. 16, no.6, pp. 688–733.

[14] R. Pajarola and J. Rossignac, "Compressed progressive meshes," in *IEEE Transcations on Visualization and Computer Graphics*, January 2000, vol. 6, no.1, pp. 79–93.

[15] 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," in *IEEE Signal Processing Magazine*, May 2013, vol. 30, no.3, pp. 83–98.

[16] H. A. Nguyen, P. A. Chou, and Y. Chen, "Compression of human body sequences using graph wavelet filter banks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Florence, Italy, May 2014.

[17] D. Thanou, P. Chou, and P. Frossard, "Graph-based compression of dynamic 3D point cloud sequences," in *IEEE Transactions on Image Processing*, April 2016, vol. 25, no.4, pp. 1765–1778.

[18] A. Anis, P. A. Chou, and A. Ortega, "Compression of dynamic 3D point cloud using subdivisional meshes and graph wavelet transforms," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Shanghai, China, March 2016.

[19] Y.-H. Chao, A. Ortega, W. Hu, and G. Cheung, "Edge-adaptive depth map coding with lifting transform on graphs," in *31st Picture Coding Symposium*, Cairns, Australia, May 2015.

[20] A. Bronstein, M. Bronstein, and R. Kimmel, "Calculus of nonrigid surfaces for geometry and texture manipulation," in *IEEE Transactions on Visualization and Computer Graphics*, May 2008, vol. 13, no.5, pp. 902–913.

[21] M. Isenburg and S. Gumhold, "Out-of-core compression for gigantic polygon meshes," in *ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'03)*, San Diego, CA, July 2003.

[22] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno, "Metro: Measuring error on simplified surfaces," in *Computer Graphics Forum*, 1998, vol. 17, no.2, pp. 167–174.