POINT CLOUD ATTRIBUTE INPAINTING IN GRAPH SPECTRAL DOMAIN

Ju He, Zeqing Fu, Wei Hu **, Zongming Guo*

Institute of Computer Science & Technology, Peking University

a

ABSTRACT

With the prevalence of depth sensors and 3D scanning devices, point clouds have attracted increasing attention as a format for 3D object representation, with applications in various fields such as tele-presence, navigation for autonomous driving and heritage reconstruction. However, point clouds usually exhibit holes of missing data, mainly due to the limitation of acquisition techniques and complicated structure. Hence, we propose an efficient inpainting method for the attribute (e.g., color) of point clouds, exploiting non-local self-similarity in graph spectral domain. Specifically, we represent irregular point clouds naturally on graphs, and split a point cloud into fixed-sized cubes as the processing unit. We then globally search for the most similar cubes to the target cube with holes inside, and compute the graph Fourier transform (GFT) basis from the similar cubes, which will be leveraged for the GFT representation of the target patch. We then formulate attribute inpainting as a sparse coding problem, imposing sparsity on the GFT representation of the attribute for hole filling. Experimental results demonstrate the superiority of our method.

Index Terms— Graph signal processing, point cloud, attribute inpainting, self-similarity, sparse coding

1. INTRODUCTION

Point clouds have received increasing attention as a basic form of 3D formats. A 3D point cloud consists of a set of points in 3D space, often with attributes such as color. The development of depth sensing and 3D laser scanning techniques enables convenient acquisition of point clouds, which leads to the application in a variety of fields such as autonomous driving, 3D telepresence, augmented reality, heritage reconstruction and so on [1]. However, scanned point clouds often suffer from missing data due to occlusion, low reflectance of the scanned surface, limited number of scans from different viewing directions, etc. Hence, point cloud inpainting is necessary prior to the subsequent applications.

Geometry inpainting has been extensively studied in recent years. Various approaches have been proposed, which mainly include two classes according to the cause of holes: 1) restore holes in the object itself such as heritage and sculptures [2–5], and 2) inpaint holes caused by the limitation of scanning devices [6–8]. For the first class of methods, the main hole-filling data source is online database, as the holes are often large. For example, Sahay et al. [2] attempt to fill big holes using the neighbourhood surface and geometric prior derived from registered similar examples in a library. The other class of methods focus on holes generated due to the limitations of scanning devices. This kind of holes are rather smaller than the aforementioned ones in general, thus the information of the data itself is often enough for inpainting. For example, Fu et al. [8] and Hu et al. [9] exploit the non-local self-similarity in point clouds, and inpaint holes from similar patches, which is regularized by a graph-signal smoothness prior to enforce geometric consistency.

However, few inpainting methods focus on the attributes of point clouds such as color, which is important for applications like 3D telepresence and heritage reconstruction. As far as we know, only Lozes et al. [10] propose point cloud inpainting on color, where they refer to the neighborhood of the hole to compute the geometric structure and color information and deploy partial difference operators to optimize the inpainting result. Consequently, the inpainted point clouds tend to be planar than the ground truth in both geometry and color. Besides, artifacts are likely to occur around the boundary when the structure is complicated.

In order to address the above problems, we propose to inpaint the attributes in point clouds in graph spectral domain. Taking color as an example, the key idea is to exploit the *sparsity in graph spectral domain*, where the dictionary for the sparse representation of the target region with missing color information resorts to *non-locally similar* regions. Due to the irregularity of point clouds, we represent point clouds on graphs naturally. Specifically, we first segment the input point cloud into cubes of the same size. Then we define the similarity metric between two cubes based on the difference of known color information at corresponding positions in two cubes. Based on the similarity metric, we obtain the most similar cube to the target as the final source cube. Next, we formulate color inpainting as a sparse coding problem in graph spectral domain, where the dictionary is the basis of Graph Fourier Transform (GFT) [11] computed from the most similar cube. Finally, we acquire the closed-form solution of the

This work was supported in part by Beijing Natural Science Foundation (4194080) and in part by National R&D project of China under contract No. 2018YFB1003504.

optimization problem, leading to the inpainted color information.

The outline of the paper is as follows. We first review graph signal processing tools in Section 2. We then elaborate on the proposed method, including problem formulation, cube matching and optimization In Section 3. Experimental results and conclusion are presented in Section 4 and 5, respectively.

2. BACKGROUND

2.1. Graph Laplacian and Graph Fourier Transform

We consider an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ composed of a vertex set $\mathcal{V}$ of cardinality $|\mathcal{V}| = N$, an edge set $\mathcal{E}$ connecting vertices, and a weighted *adjacency matrix* $\mathbf{W}$. $\mathbf{W}$ is a real symmetric $N \times N$ matrix, where $w_{i,j}$ is the weight assigned to the edge (i, j) connecting vertices i and j. We assume non-negative weights, *i.e.*, $w_{i,j} \geq 0$.

The Laplacian matrix, defined from the adjacency matrix, is a fundamental algebraic representation of a graph [12]. Among different variants of Laplacian matrices, the *combinatorial graph Laplacian* used in [13–15] is defined as $\mathcal{L} := \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is the *degree matrix*—a diagonal matrix where $d_{i,i} = \sum_{j=1}^{N} w_{i,j}$.

Graph signal refers to data residing on the vertices of a graph. For example, if we construct a K-nearest-neighbor (K-NN) graph on the point cloud, then the color of each point can be treated as graph signal defined on the K-NN graph. This will be discussed further in the proposed problem formulation in Section 3.1.

Since the graph Laplacian is a real symmetric matrix, it admits a set of real eigenvalues $\{\lambda_l\}_{l=0,1,...,N-1}$ with a complete set of orthonormal eigenvectors $\mathbf{U} = \{\mathbf{u}_l\}_{l=0,1,...,N-1}$, *i.e.*, $\mathcal{L}\mathbf{u}_l = \lambda_l \mathbf{u}_l$, for $l = 0, 1, ..., N-1$. The eigenvectors then define the basis of GFT.

Formally, for any signal $\mathbf{x} \in \mathbb{R}^N$ residing on the vertices of $\mathcal{G}$, its GFT $\boldsymbol{\eta} \in \mathbb{R}^N$ is defined in [11] as

$$\boldsymbol{\eta} = \mathbf{U}^T \mathbf{x}, \tag{1}$$

and the inverse GFT follows as

$$\mathbf{x} = \mathbf{U}\boldsymbol{\eta} = \sum_i \eta_i \mathbf{u}_i. \tag{2}$$

2.2. Sparse Coding

Sparse coding [16] assumes that a patch $\mathbf{y}$ can be represented by a weighted combination of only a few atoms out of a learned dictionary $\mathbf{U}$. In other words, it means finding a dictionary $\mathbf{U}$ and a weight vector $\boldsymbol{\eta}$ for patch $\mathbf{y}$ such that: 1) each patch $\mathbf{y}$ is well approximated by $\mathbf{U}\boldsymbol{\eta}$, and 2) the sparsity of $\boldsymbol{\eta}$, approximated by $||\boldsymbol{\eta}||_1$, is minimized. Mathematically,

$$\min_{\mathbf{U},\boldsymbol{\eta}} \; ||\mathbf{y} - \mathbf{U}\boldsymbol{\eta}||_2 + \beta ||\boldsymbol{\eta}||_1 \tag{3}$$

where β is a Lagrange multiplier trading off the approximation error and sparsity. The l_1 norm regularizer approximates the sparsity of the weight vector $||\boldsymbol{\eta}||_0$.

3. PROPOSED METHOD

We first discuss the proposed problem formulation, leveraging sparse coding and spectral graph theory in Section 2, and then elaborate on the developed algorithm.

3.1. Problem Formulation

We formulate point cloud attribute inpainting as sparse coding of a given observation with missing data. In particular, for a target region with holes, we construct a dictionary from its most similar region, and inpaint the target region by finding a sparse representation of the signal given the constructed dictionary. Note that, sparse coding is mainly deployed over patches in 2D images on regular grids, while point clouds are irregular 3D data. In order to address this, we split the entire point cloud into overlapping cubes of the same size, and employ cubes instead of patches as the processing unit.

The dictionary construction is the key to attribute inpainting. We propose to deploy the GFT basis $\mathbf{U}$ of the most similar region as the dictionary, because the basis characterizes the underlying structure of a region. Hence, we formulate point cloud attribute inpainting as

$$\min_{\mathbf{c}_r,\boldsymbol{\eta}} \; ||\Omega\mathbf{c}_r - \Omega\mathbf{c}_t||_2^2 + \alpha||\mathbf{c}_r - \mathbf{U}\boldsymbol{\eta}||_2^2 + \beta||\boldsymbol{\eta}||_1^2, \tag{4}$$

where $\mathbf{c}_r \in R^{M^3 \times 3}$ is the desired cube. Ω is a $M^3 \times M^3$ diagonal matrix with $\Omega_{i,i} \in \{0, 1\}$, $i = 1, ..., M^3$, extracting the known region in $\mathbf{c}_r$ and $\mathbf{c}_t$. α and β are two weighting parameters (empirically $\alpha = 0.5$ and $\beta = 0.5$ in our experiments). $\boldsymbol{\eta}$ is the weight vector in sparse coding, and $\mathbf{U}$ is GFT basis of $\mathbf{c}_s$ separately.

The first term in (4) is a data fidelity term, which ensures the desired cube to be close to $\mathbf{c}_t$ in the known region. The second and third terms are the approximation error and sparsity respectively, as in the formulation of sparse coding introduced in Section 2.2.

3.2. Algorithm Development

Having formulated the attribute inpainting problem, we develop an iterative algorithm to solve it. As shown in Fig. 1, the input data is a point cloud denoted by $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\}$ with $\mathbf{p}_i \in \mathbb{R}^6$ consisting of the coordinates and the color of the i-th point in the point cloud. Firstly, we split it into fixed-sized cubes as the processing unit in the subsequent steps. Secondly, we choose the target cube and inpaint the geometry using the method in [8]. Thirdly, we search the most similar cube to the target cube based on the proposed similarity metric. Finally, we address the formulated optimization problem via the closed-form solution iteratively, which leads to the inpainting result. The details of each step are discussed as follows.

Preprocessing We first split the input point cloud into overlapping cubes $\{\mathbf{c}_1, \mathbf{c}_2, ...\}$ with $\mathbf{c}_i \in \mathbb{R}^{M^3 \times 6}$ (M is the dimension of the cube) as the processing unit. M is empirically set according to the coordinate range of $\mathbf{P}$ ($M = 20$ in our experiments). Then the overlapping step is empirically

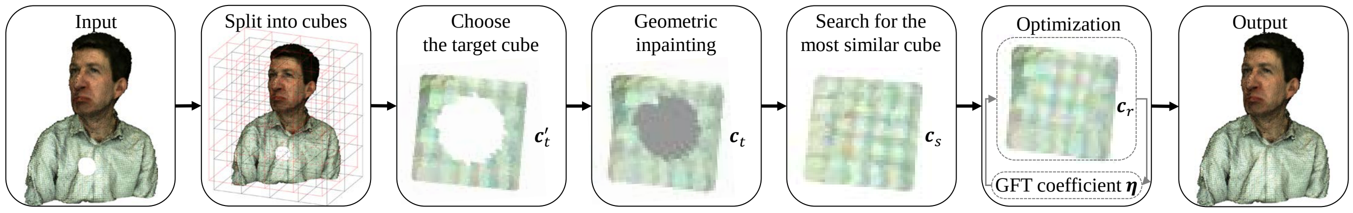

Fig. 1. The framework of the proposed point cloud attribute inpainting method.

set as $\frac{M}{4}$. This is a trade-off between the computational complexity and ensuring enough information available to search for the most similar cube.

Having obtained cubes, we choose the cube with missing data as the target cube $\mathbf{c}_t^{'}$. Then we adopt the method proposed in [8] to inpaint the geometry, resulting in the geometry-inpainted cube $\mathbf{c}_t$. Besides, in order to save the computation complexity and increase the accuracy of the subsequent cube matching, we choose candidate cubes $\mathbf{c}_c$ by filtering out cubes with the number of points less than 80% of that of $\mathbf{c}_t$, which will be used in the next step as follows.

Cube Matching In order to search for the most similar cube to the target, we define the color similarity metric $\delta(\mathbf{c}_t, \mathbf{c}_c)$ between the target cube $\mathbf{c}_t$ and candidate cubes $\mathbf{c}_c$ as

$$\delta(\mathbf{c}_t, \mathbf{c}_c) = \delta(R) + \delta(G) + \delta(B), \tag{5}$$

$$\delta(R) = \sum_{j=1}^{M^3} [(\mathbf{c}_{t,j})_R - (\mathbf{c}_{c,j'})_R]^2, \tag{6}$$

where $\mathbf{c}_{t,j}$ is the jth point in $\mathbf{c}_t$ and $\mathbf{c}_{c,j'}$ is the j'th point in the candidate cube $\mathbf{c}_c$. $\{j, j'\}$ denotes a corresponding pair of points in $\mathbf{c}_t$ and $\mathbf{c}_c$. We find this correspondence by searching the nearest point of $\mathbf{c}_{t,j}$ in $\mathbf{c}_c$ in terms of Euclidean distance of the relevant position. $(\mathbf{c}_{t,j})_R$ and $(\mathbf{c}_{c,j'})_R$ represent the color information in channel R of $\mathbf{c}_{t,j}$ and $\mathbf{c}_{c,j'}$, separately. $\delta(G)$ and $\delta(B)$ are defined in the same way as in (6).

Having computed the similarity metric in (5) between the target cube and candidate cubes, we choose the candidate cube with the largest similarity as the source cube $\mathbf{c}_s$.

Graph Construction With the source cube $\mathbf{c}_s$ chosen, we construct a K-NN graph over $\mathbf{c}_s$ based on the affinity of geometric distance among points in $\mathbf{c}_s$, from which we compute the GFT basis $\mathbf{U}$ as the dictionary in the final formulation. The number of nearest neighbors K is considered to be related to the number of existing points in the cube. Specifically, we build unweighted graphs for simplicity. Hence, $w_{k,l}$ is assigned as

$$w_{k,l} = \begin{cases} 1, & k \sim l \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Optimization Considering the computational complexity of optimizing l_1 norm, we further relax the l_1 regularizer in (4) as l_2 norm, leading to the final formulation

$$\min_{\mathbf{c}_r, \boldsymbol{\eta}} \; ||\Omega\mathbf{c}_r - \Omega\mathbf{c}_t||_2^2 + \alpha||\mathbf{c}_r - \mathbf{U}\boldsymbol{\eta}||_2^2 + \beta||\boldsymbol{\eta}||_2^2. \tag{8}$$

This is a quadratic programming problem. Taking derivative of the objective with respect to $\mathbf{c}_r$ and $\boldsymbol{\eta}$ separately, we have the closed-form solution:

$$\hat{\mathbf{c}_r} = (\Omega^T\Omega + I)^{-1}(\Omega^T\Omega\mathbf{c}_t + \alpha\mathbf{U}\boldsymbol{\eta}), \tag{9}$$

$$\hat{\boldsymbol{\eta}} = (\alpha + \beta)^{-1}(\mathbf{U}^T\mathbf{c}_r). \tag{10}$$

As $\Omega^T\Omega = \Omega$, the first multiplier in (9) is a diagonal matrix with diagonal element 0.5 for known regions and 1 for unknown regions. The second multiplier combines the information from known regions and the dictionary representation. In (10), α and β adjust the GFT representation of $\mathbf{c}_r$.

We first initialize $\mathbf{c}_r$ as $\mathbf{c}_t$ and compute $\boldsymbol{\eta}$ according to (10). Then we alternately fix one variable and update the other until the difference of the objective between iterations is small enough. Finally, we replace the target cube with the resulting cube, which serves as the output.

4. EXPERIMENTAL RESULTS

4.1. Experimental Setup

We evaluate the proposed method by testing on several point cloud datasets from Microsoft, including *Andrew*, *David*, *Phili*, *Ricardo*, *Sarah* and *Soldier* [17]. We test on two types of holes: 1) real holes generated during the capturing process, which have no ground truth; 2) synthetic holes on point clouds so as to compare with the ground truth.

Further, we compare our method with the PDE method [10] and Meshlab [18]. Note that Meshlab performs inpainting based on meshes. So we convert point clouds to meshes first, perform the algorithm, and then convert the inpainted meshes back to point clouds as the final output.

4.2. Experimental Results

It is nontrivial to measure the color difference of point clouds objectively due to the irregularity. We apply the color distortion metric MSE as the metric for evaluation. Note that traditional MSE [19] computes the difference of two data with one-to-one correspondence between each other, while the correspondence is nontrivial in point clouds. We propose to find the nearest point in inpainting results for each point in the ground truth to establish the correspondence. Then the MSE values of R, G, B channels are averaged to acquire the final MSE.

Table 1 shows the objective results for synthetic holes. We see that our scheme outperforms all the competing methods in MSE significantly. Specifically, we reduce 82.71% in

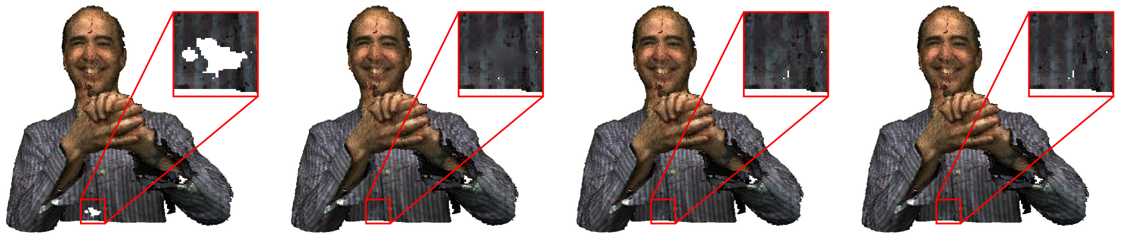
(a) (b) (c) (d)

Fig. 2. Inpainting for *Phili* with a real hole marked in red and magnified. (a) Original point cloud with holes. (b) Results obtained by Meshlab. (c) Results obtained by PDE. (d) Results obtained by the proposed method.

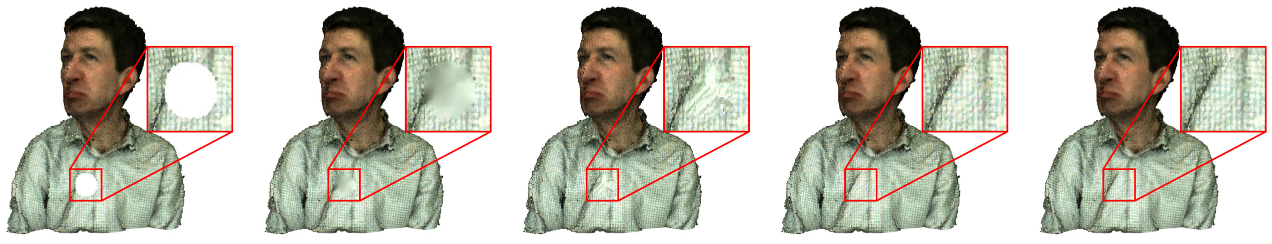
(a) (b) (c) (d) (e)

Fig. 3. Inpainting for *Andrew* with a synthetic hole marked in red and magnified. (a) Point cloud with holes. (b) Results obtained by Meshlab. (c) Results obtained by PDE. (d) Results obtained by the proposed method. (e) The ground truth.

MSE on average compared with Meshlab and reduce 67.97% in MSE compared with the PDE method.

Table 1. Performance Comparison in MSE

	Meshlab	PDE [10]	Proposed
Andrew	1045.08	751.15	**205.98**
David	254.51	31.59	**8.18**
Ricardo	44.15	8.01	**2.33**
Sarah	1544.93	1018.28	**283.64**
Soldier	1026.07	303.83	**176.62**

Further, Fig. 2 and Fig. 3 demonstrate subjective inpainting results for real holes and synthetic holes respectively. The results of Meshlab tend to be planar and blurry, because Meshlab attempts to fill the hole by connecting the boundary of the hole region via a simple plane. The results of the PDE method exhibit artificial inward and gradual change in color. This is because this method leverages the local neighborhood information to inpaint the missing areas from outside towards inside, which accumulates the inpainting error and leads to distortion in the results. Our results shown in Fig. 2 (d) and Fig. 3 (d) demonstrate that our method is able to inpaint holes with appropriate color information, even for holes with complicated color structure such as streaks and checker.

Besides, we validate the sparsity of the acquired weight vector in our formulation. Taking the green channel as an example, Fig. 4 presents how the weight vector $\boldsymbol{\eta}$ (essentially GFT coefficients in our formulation) varies with respect to eigenvalues. We see that $\boldsymbol{\eta}$ distributes almost evenly over all the frequencies prior to optimization. It then becomes quite *compact* in the low frequencies after optimization, which gives credits to the relaxation of sparse coding.

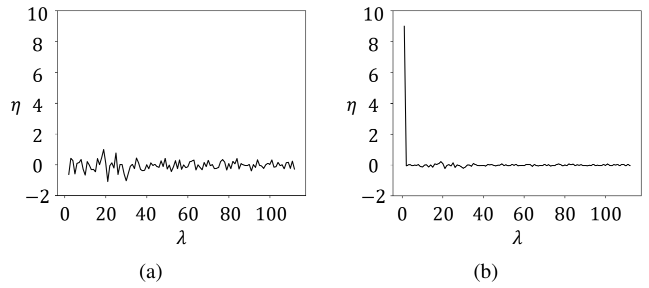

Fig. 4. Demonstration of the sparsity of the weight vector $\boldsymbol{\eta}$ with respect to eigenvalues for the green channel. (a) prior to our optimization; (b) after our optimization.

5. CONCLUSION

We propose an efficient 3D point cloud attribute inpainting approach in graph spectral domain. The key observation is that point clouds exhibit non-local self-similarity in the attributes and exhibit sparsity in the GFT domain. We thus propose to fill each hole in a point cloud from a cube with similar attribute information. We then cast point cloud attribute inpainting as a sparse coding problem, based on the selected most similar cube and regularized by sparsity in the GFT domain. Experimental results show that our algorithm outperforms existing competing methods significantly.

6. REFERENCES

[1] C. Tulvan, R. Mekuria, and Z. Li, "Use cases for point cloud compression (pcc)," in *ISO/IEC JTC1/SC29/WG11 (MPEG) output document N16331*, June 2016.

[2] P. Sahay and A. N. Rajagopalan, "Harnessing self-similarity for reconstruction of large missing regions in 3D models," in *International Conference on Pattern Recognition*, 2012, pp. 101–104.

[3] P. Sahay and A. N. Rajagopalan, "Geometric inpainting of 3D structures," in *Computer Vision and Pattern Recognition Workshops*, 2015, pp. 1–7.

[4] S. Shankar, S. A. Ganihar, and U. Mudenagudi, "Framework for 3D object hole filling," in *Fifth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics*, 2015, pp. 1–4.

[5] C. Dinesh, I. V. Bajic, and G. Cheung, "Exemplar-based framework for 3D point cloud hole filling," in *IEEE International Conference on Visual Communications and Image Processing*, May 2017.

[6] J. Wang, M. M. Oliveira, M. Garr, and M. Levoy, "Filling holes on locally smooth surfaces reconstructed from point clouds," *Image & Vision Computing*, vol. 25, no. 1, pp. 103–113, 2007.

[7] H. Lin and W. Wang, "Feature preserving holes filling of scattered point cloud based on tensor voting," in *IEEE International Conference on Signal and Image Processing*, 2017, pp. 402–406.

[8] Z. Fu, W. Hu, and Z. Guo, "Point cloud inpainting on graphs from non-local self-similarity," in *IEEE International Conference on Image Processing*, Athens, Greece, 2018.

[9] W. Hu, Z. Fu, and Z. Guo, "Local frequency interpretation and non-local self-similarity on graph for point cloud inpainting," *accepted to IEEE Transactions on Image Processing*, 2019.

[10] F. Lozes, A. Elmoataz, and O. Lzoray, "Pde-based graph signal processing for 3-d color point clouds : Opportunities for cultural heritage," *IEEE Signal Processing Magazine*, vol. 32, no. 4, pp. 103–111, 2015.

[11] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied & Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2009.

[12] F. K. Chung, "Spectral graph theory," vol. 92, no. 6, pp. 212, 1996.

[13] G. Shen, W.-S. Kim, S. K. Narang, A. Ortega, J. Lee, and H. Wey, "Edge-adaptive transforms for efficient depth map coding," in *IEEE Picture Coding Symposium*, Nagoya, Japan, December 2010, pp. 566–569.

[14] W. Hu, G. Cheung, X. Li, and O. C. Au, "Depth map compression using multi-resolution graph-based transform for depth-image-based rendering," in *IEEE International Conference on Image Processing*, Orlando, FL, September 2012, pp. 1297–1300.

[15] W. Hu, G. Cheung, A. Ortega, and O. C. Au, "Multi-resolution graph fourier transform for compression of piecewise smooth images," in *IEEE Transactions on Image Processing*, January 2015, vol. 24, pp. 419–33.

[16] Z. Jiang, L. Zhe, and L. S. Davis, "Learning a discriminative dictionary for sparse coding via label consistent k-svd," in *IEEE Conference on Computer Vision & Pattern Recognition*, 2011.

[17] Q. Cai and P. A. Chou, "Microsoft voxelized upper bodies a voxelized point cloud dataset," in *ISO/IEC JTC1/SC29 WG11 ISO/IEC JTC1/SC29/WG1 input document m38673/M72012*, May 2016.

[18] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "Meshlab: an open-source mesh processing tool," in *Eurographics Italian Chapter Conference*, S. Vittorio, D. C. Rosario, and F. Ugo, Eds. 2008, The Eurographics Association.

[19] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? a new look at signal fidelity measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, 2008.