POINT CLOUD INPAINTING ON GRAPHS FROM NON-LOCAL SELF-SIMILARITY

*Zeqing Fu, Wei Hu *, Zongming Guo*

Institute of Computer Science & Technology, Peking University

ABSTRACT

As 3D scanning devices and depth sensors advance, point clouds have attracted increasing attention as a format for 3D object representation, with applications in various fields such as tele-presence, navigation 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 point cloud inpainting method, leveraging on graph signal processing and based on the observation of non-local self-similarity in point clouds. Specifically, we split a point cloud into fixed-size cubes as the processing unit, and globally search for the most similar cube to the target cube with holes inside. The similarity metric between two cubes is defined based on the direct component and the proposed anisotropic graph total variation of normals in each cube. We then formulate the hole-filling step as an optimization problem, based on the selected most similar cube and regularized by a graph-signal smoothness prior. Experimental results show that the proposed approach outperforms three competing methods significantly, both in objective and subjective quality.

***Index Terms*—** Graph signal processing, point cloud inpainting, non-local self-similarity, anisotropic graph total variation

1. INTRODUCTION

Point clouds have received increasing attention as a basic form of 3D formats. It consists of a set of points, each of which corresponds to a measurement point and contains most original information of the point, including the 3D coordinates representing the geometric information and possibly attribute information such as color and normal. Hence, point cloud is a natural representation of arbitrarily-shaped objects. With the development of depth sensing and 3D laser scanning techniques, we can acquire point clouds conveniently, thus catalyzing its applications in various fields, such as 3D immersive tele-presence, navigation, and heritage reconstruction [1].

However, point clouds often exhibit several holes of missing data inevitably, as shown in Fig. 1. This is mainly due to incomplete scanning views and inherent limitations of the equipments. Besides, the data may lack some regions in itself (*e.g.*, heritage). Therefore, it is necessary to inpaint the incomplete point clouds prior to subsequent applications.

Inpainting has been studied extensively in the past decades, from 2D images to 3D formats. Regarding 2D images, two major families of methods have been proposed based on solving Poisson equations [2–4] and exemplar-based texture synthesis [5–7] respectively. As to depth maps, which serve as 2.5D reprepresentation, there are two classes of typical methods: patch-based [8, 9] and edge-based [10, 11]. When it comes to 3D formats, various approaches have been proposed to fill holes in 3D meshes, such as interpolating to create implicit surfaces which is used in RameshCleaner [12] and [13–15], and speculating boundary vertices iteratively to shrink holes such as Meshfix [16], Meshlab [17] and [18–20] .

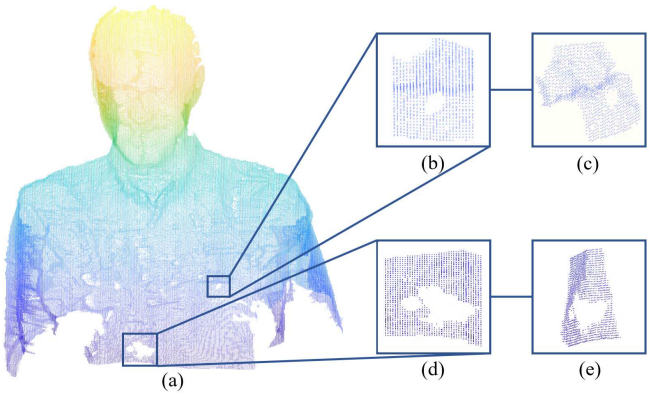


Fig. 1. Point cloud *Phili* (a), with two examples of holes generated due to the inherent limitations of the acquisition equipments. (b) (c) are different views for one hole and (d) (e) are different views for the other hole.

However, few inpainting methods [21–27] have been proposed for point clouds so far, and most of them address missing areas from the object itself such as heritage and sculptures [21–23]. The main hole-filling data source of these kinds of methods is online databases, due to the large size of the holes. Sahay et al. [22] firstly tries to fill big damage regions using neighbourhood surface geometry and geometric prior derived from registered similar examples in a library. Then they propose another method [23], which projects the point cloud to a depth map, searches a similar one from an online depth database using dictionary learning, and then adaptively propagates local 3D surface smoothness. However, the projection process inevitably introduces geometry loss. Besides, some works for point clouds deal with particular point cloud data such as flattened bar-shaped holes in the human body data in [24], and geometrically regular point clouds of buildings in [25].

In this paper, we aim to fill holes particularly generated in the process of acquiring point clouds, whereas we leave inpainting missing areas in the object itself as the future work. To avoid planar hole-filling content, our key idea is to take advantage of the *non-local self-similarity* in the geometry of point clouds, i.e., finding a source region which is the most similar to the target region(*i.e.*, the region containing the hole), and then filling the hole according to the source region. Due to the irregularity of point clouds, it is difficult to find similar regions, thus we resort to graph signal processing [28], which represents point clouds on graphs naturally. Specifically, we first segment the input point cloud into cubes with the same size. Then we define the similarity metric between two cubes based on 1) the *direct component* (DC) of the normals of points in the cube and

This work was supported by Alibaba Innovative Research under contract No. XTG201800108 and MSRA Collaborative Research under project ID FY18-Research-Sponsorship-029.

2) the proposed *anistropic graph total variation* (AGTV) of normals, which is an extension of graph total variation [29–31]. We then obtain the most similar cube to the target as the final source cube via this metric. Next, we formulate the hole-filling step as an optimization problem, with a graph-signal smoothness prior [32] for the target cube. Finally, we acquire the closed-form solution of the optimization problem, leading to the inpainting result of the hole.

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

2. SPECTRAL GRAPH THEORY

We first provide a review on basic concepts in spectral graph theory [33], including graph, graph Laplacian and graph-signal smoothness prior, which will be utilized in the proposed point cloud inpainting.

2.1. Graph and Graph Laplacian

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, can be used to uncover many useful properties of a graph. Among different variants of Laplacian matrices, the *combinatorial graph Laplacian* used in [34–36] is defined as $\mathcal{L} := \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is the *degree matrix*—a diagonal matrix where $D_{i,i} = \sum_{j=1}^{N} w_{i,j}$.

2.2. Graph-Signal Smoothness Prior

Graph signal refers to data residing on the vertices of a graph, such as social, transportation, sensor, and neuronal networks. For example, if we construct a K-NN graph on the point cloud, then the normal of each point can be treated as graph signal defined on the K-NN graph, as shown in Fig. 2. This will be discussed further in the proposed cubic matching approach.

A graph signal $\mathbf{x}$ defined on a graph $\mathcal{G}$ is smooth with respect to the topology of $\mathcal{G}$ if

$$\sum_{i \sim j} w_{i,j}(x_i - x_j)^2 < \epsilon, \ \ \forall i, j, \tag{1}$$

where ϵ is a small positive scalar, and $i \sim j$ denotes two vertices i and j are one-hop neighbors in the graph. In order to satisfy (1), x_i and x_j have to be similar for a large edge weight $w_{i,j}$, and could be quite different for a small $w_{i,j}$. Hence, (1) enforces $\mathbf{x}$ to adapt to the topology of $\mathcal{G}$, which is thus coined *graph-signal smoothness prior*.

As $\mathbf{x}^T \mathcal{L} \mathbf{x} = \sum_{i \sim j} w_{i,j}(x_i - x_j)^2$ [37], (1) is concisely written as $\mathbf{x}^T \mathcal{L} \mathbf{x} < \epsilon$ in the sequel. This prior will be deployed in our problem formulation of point cloud inpainting as a regularization term, as discussed in Section 3.

3. PROPOSED METHOD

We now introduce the proposed point cloud inpainting method based on the spectral graph theory in Section 2. As shown in Fig. 3, the input data is a point cloud denoted by $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\}$ with $\mathbf{p}_i \in R^3$ meaning the coordinates of the i-th point. Firstly, we split it into fixed-size cubes as units to be processed in the subsequent steps. Secondly, we choose the target cube artificially. Thirdly, we search for the most similar cube to the target cube in $\mathbf{P}$, which is referred to as the source cube, based on the DC and AGTV in the normals of points. Fourthly, the inpainting problem is formulated into an optimization problem, which leverages the source cube with a graph-signal smoothness prior. The closed-form solution of the optimization problem gives the resulting cube. Finally, we replace the target cube with the resulting cube as the output.

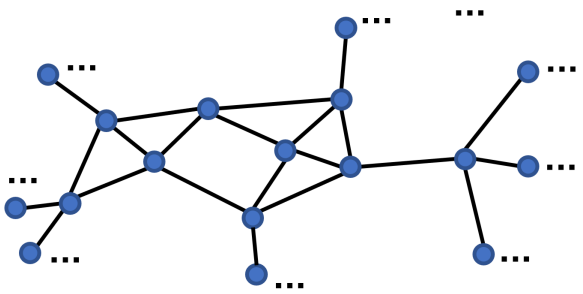


Fig. 2. A K-NN graph constructed when K= 4. The connections of boundary vertices are omitted.

3.1. Preprocessing

We first split the input point cloud into overlapping cubes $\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_z\}$ with $\mathbf{c}_i \in R^{M^3 \times 3}$ (M is the size of the cube), as the processing unit of the proposed inpainting algorithm. M is empirically set according to the coordinate range of $\mathbf{P}$ ($M = 20$ in our experiments). Then the overlapping step is empirically set as $\frac{M}{4}$. This is trade-off between computational complexity and ensuring enough geometry information available to search for the most similar cube.

Having obtained the cubes, we choose the cube with missing data as the target cube $\mathbf{c}_t$. Further, in order to save 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.

3.2. Cube Matching

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

$$\delta(\mathbf{c}_t, \mathbf{c}_c) = \exp\{-[\delta_D(\mathbf{c}_t, \mathbf{c}_c) + \delta_V(\mathbf{c}_t, \mathbf{c}_c)]\}, \tag{2}$$

where $\delta_D(\mathbf{c}_t, \mathbf{c}_c)$ and $\delta_V(\mathbf{c}_t, \mathbf{c}_c)$ are the difference in DC and AGTV between $\mathbf{c}_t$ and $\mathbf{c}_c$ respectively. Specifically, they are defined as

$$\delta_D(\mathbf{c}_t, \mathbf{c}_c) = \ | < \mathbf{d}(\mathbf{c}_t), \ \mathbf{d}(\mathbf{c}_c) > |, \tag{3}$$

$$\delta_V(\mathbf{c}_t, \mathbf{c}_c) = |v(\mathbf{c}_t) - v(\mathbf{c}_c)|, \tag{4}$$

where $\mathbf{d}(\mathbf{c}_t)$ and $\mathbf{d}(\mathbf{c}_c)$ are the DC of cube $\mathbf{c}_t$ and $\mathbf{c}_c$, while $v(\mathbf{c}_t)$ and $v(\mathbf{c}_c)$ are the AGTV of $\mathbf{c}_t$ and $\mathbf{c}_c$. We explain $\delta_D(\mathbf{c}_t, \mathbf{c}_c)$ and $\delta_V(\mathbf{c}_t, \mathbf{c}_c)$ in detail as follows.

Direct Component This is a function of the normals of points in the cube, which presents the *prominent geometry direction* of the

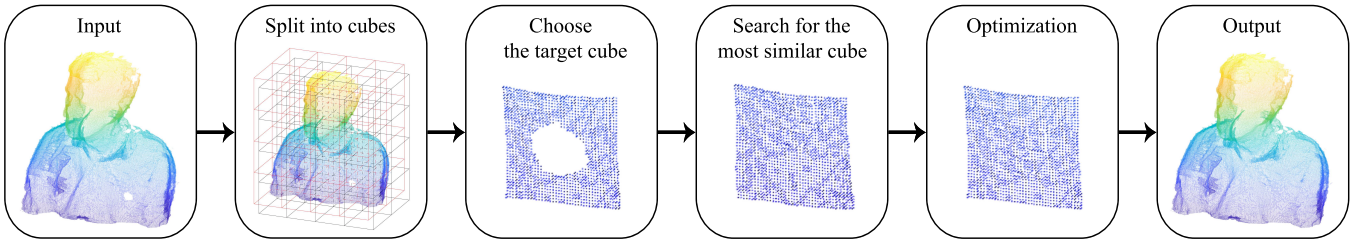


Fig. 3. The framework of the proposed point cloud inpainting method.

cube. A cube $\mathbf{c}_i$ consists of a set of points $\{c_{i,1}, c_{i,2}, ..., c_{i,m}\}$ (m is the number of the points), each of which corresponds to a normal computed via Meshlab [17], denoted as $\{\mathbf{n}_{i,1}, \mathbf{n}_{i,2}, ..., \mathbf{n}_{i,m}\}$. We compute DC of the cube $\mathbf{c}_i$ as

$$\mathbf{d}(\mathbf{c}_i) = \frac{\sum_{k=1}^{m} \mathbf{n}_{i,k}}{\|\sum_{k=1}^{m} \mathbf{n}_{i,k}\|_2^2}. \tag{5}$$

Anisotropic Graph Total Variation Unlike traditional total variation used in image denoising [38], graph total variation generalizes to describe the smoothness of a graph signal with respect to the graph structure. Further, in order to describe the geometry feature of point clouds more accurately, we propose AGTV based on the normals in a cube instead of the conventional graph total variation. The mathematical definition of the proposed AGTV for $\mathbf{c}_i$ is

$$v(\mathbf{c}_i) = \frac{\sum_k \sum_l | < \mathbf{n}_{i,k}, \mathbf{n}_{i,l} > | w_{k,l}}{K(K-1)}. \tag{6}$$

Here $w_{k,l}$ denotes the weight of the edge between k and l in a graph we construct over $\mathbf{c}_i$. Specifically, we choose to build a K-NN graph mentioned in Section 2.2, based on the affinity of geometric distance among points in $\mathbf{c}_i$. Also, we consider unweighted graphs for simplicity, which means $w_{k,l}$ is assigned as

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

Note that the proposed AGTV is the l_1 norm of the overall difference in normals in the graph. Hence, this favors the sparsity of the overall graph gradient in normals, which is able to describe abrupt signal changes efficiently.

Having computed the similarity metric in (2) between the target cube and candidate cubes, we choose the candidate cube with the largest similarity as the source cube $\mathbf{c}_s$. However, $\mathbf{c}_s$ cannot be directly adopted for inpainting, because it is just the most similar to $\mathbf{c}_t$ in geometry structure, but not in the relative location in the cube. Hence, we further translate $\mathbf{c}_s$ by the difference in location between $\mathbf{c}_s$ and $\mathbf{c}_t$. The difference in each dimension, denoted as $\mathbf{h} = (h_x, h_y, h_z)$, is computed as

$$h_x = \sum (\partial\Omega \mathbf{c}_t - \partial\Omega \mathbf{c}_s)_x \tag{8}$$

where $\partial\Omega$ is a $M^3 \times M^3$ diagonal matrix, extracting the boundary within one hop of the missing region in the cube. h_y and h_y are defined in the same way as (8).

The $\mathbf{h}$-translated cube, denoted as $\mathbf{c}'_s$, will be adopted in the hole-filling step as discussed next.

3.3. Problem Formulation

Next, we cast the inpainting problem as an optimization problem, which is regularized with a graph-signal smoothness prior as mentioned in Section 2.2. This problem is formulated as

$$\min_{\mathbf{c}_r} \ \|\overline{\Omega}\mathbf{c}_r - \overline{\Omega}\mathbf{c}_t\|_2^2 + \alpha\|\Omega\mathbf{c}_r - \Omega\mathbf{c}'_s\|_2^2 + \beta\mathbf{c}_r^T\mathcal{L}\mathbf{c}_r, \tag{9}$$

where $\mathbf{c}_r \in R^{M^3 \times 3}$ is the desired cube. Ω is a $M^3 \times M^3$ diagonal matrix, extracting the missing region in $\mathbf{c}_t$ and $\mathbf{c}'_s$, while $\overline{\Omega}$ is a $M^3 \times M^3$ diagonal matrix, extracting the known region. α and β are two weighting parameters (empirically $\alpha = 0.1$ and $\beta = 10$ in our experiments). Besides, $\mathcal{L}$ is the graph Laplacian matrix of $\mathbf{c}'_s$, which is computed from a K-NN graph we construct in $\mathbf{c}'_s$ in the same way as in (7).

The first term in (9) is a fidelity term, which ensures the desired cube to be close to $\mathbf{c}_t$ in the known region. The second term constraints the unknown region of $\mathbf{c}_r$ to be similar to that of $\mathbf{c}'_s$. Further, the third term is the graph-signal smoothness prior, as introduced in Section 2, which enforces the structure of $\mathbf{c}_r$ to be smooth with respect to the graph constructed over $\mathbf{c}'_s$. In other words, the third term aims to make the structure of the missing area in $\mathbf{c}_r$ mimic that in $\mathbf{c}'_s$.

(9) is a quadratic programming problem. Taking derivative of (9) with respect to $\mathbf{c}_r$, we have the closed-form solution:

$$\hat{\mathbf{c}_r} = (\overline{\Omega}^T\overline{\Omega} + \alpha\Omega^T\Omega + \beta\mathcal{L})^{-1}(\overline{\Omega}^T\overline{\Omega}\mathbf{c}_t + \alpha\Omega^T\Omega\mathbf{c}'_s). \tag{10}$$

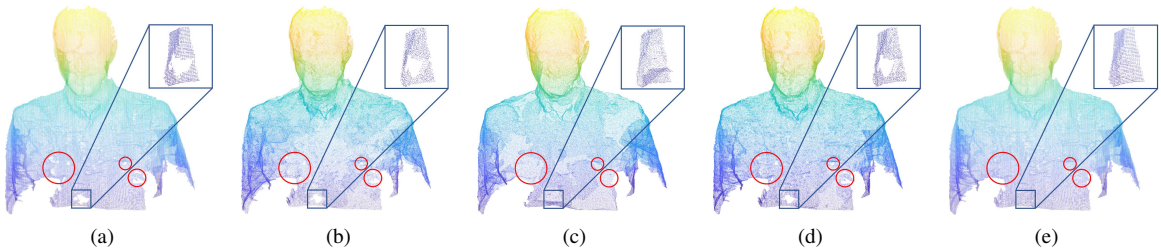
(9) is thus solved optimally and efficiently. Finally, we replace the target cube with the resulting cube, which serves as the output.

4. EXPERIMENTAL RESULTS

4.1. Experimental Setup

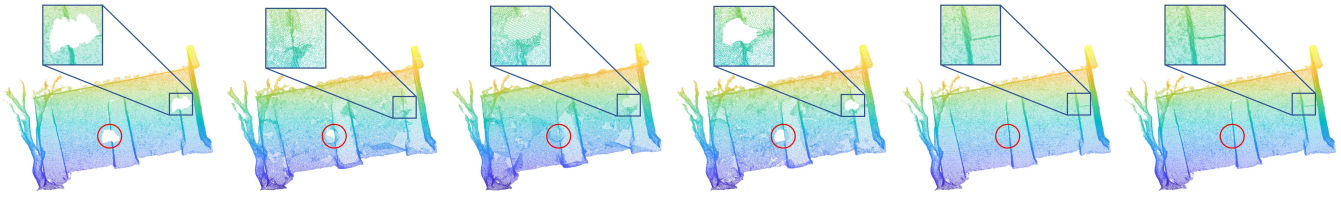
We evaluate the proposed method by testing on several point cloud datasets from Microsoft, including *Andrew*, *David*, *Phili*, *Ricardo*, *Sarah*, and *House* [39]. We test on two types of holes: 1) real holes generated during the capturing process, which have no ground truth; 2) synthetic holes on point clouds so as to compare with the ground truth. In particular, the number of nearest neighbors K is considered to be related to m, the number of existing points in the cube. Empirically, $K = \sqrt{m}$ in our experiments.

Further, we compare our method with three competing algorithms for 3D geometry inpainting, including RameshCleaner, Meshfix and Meshlab, which are based on meshes. When testing these algorithms, we convert the point clouds to meshes first, perform the algorithms, and then convert the inpainted meshes back to point clouds as the final output.



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

Fig. 4. Inpainting for *Phili* with real holes marked in red circles, with one representative cube magnified. (a) Original point cloud with holes. (b) Results obtained using Meshfix. (c) Results obtained using RameshCleaner. (d) Results obtained using Meshlab. (e) Results obtained using the proposed method.



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

Fig. 5. Inpainting for *House* with synthetic holes marked in red circles, with one representative cube magnified. (a) Point cloud with holes. (b) Results obtained using Meshfix. (c) Results obtained using RameshCleaner. (d) Results obtained using Meshlab. (e) Results obtained using the proposed method. (f) The ground truth.

4.2. Experimental Results

It is nontrivial to measure the geometry difference of point clouds objectively. We apply the geometric distortion metrics in [40], referred to as GPSNR, as the metric for evaluation. Note that GPSNR could be negative, depending on the assignment of the peak value [40].

Table 1 shows the objective results for synthetic holes. We see that our scheme outperforms all the competing methods in GPSNR significantly. Specifically, we achieve 50.46 dB gain in GPSNR on average over Meshfix, 35.98 dB over RameshCleaner, and 27.71 dB over Meshlab. Note that, holes are synthesized so that we have the ground truth for the objective comparison.

Table 1. Performance Comparison in GPSNR

	Meshfix	RameshCleaner	Meshlab	Proposed
Andrew	3.5201	20.0588	25.5331	**53.7969**
David	-10.7156	10.3991	13.7398	**39.5313**
Phili	0.9130	15.4113	25.3133	**55.7231**
Ricardo	-1.4875	7.7447	15.1558	**41.8501**
Sarah	-3.1453	6.0946	21.3919	**43.1411**
House	-5.2012	11.0806	19.2414	**52.6091**

Further, Fig. 4 and Fig. 5 demonstrate subjective inpainting results for real holes and synthetic holes respectively. The results of Meshfix and RameshCleaner exhibit non-uniform density, because they generate redundant mesh faces to fill wrong holes selected by their automatic hole detection algorithms. For the real holes in Fig. 4(a), which are fragmentary, Meshfix and Meshlab can fill small holes with planar geometry structure, but it cannot fully inpaint big missing regions, while RameshCleaner fills most of the holes by generating more redundant mesh faces. Our result shown in Fig. 4(e) demonstrates that our method is able to inpaint both small holes and big holes with appropriate geometry structure, even for holes with complicated geometry. In Fig. 5, we synthesize two holes in *House*, with simple and complex geometry structure respectively. We observe that Meshlab cannot fill the holes completely, while Meshfix and RameshCleaner introduce wrong geometry around the holes since they try to connect the boundary of the hole region using simple planes. In comparison, our result shown in Fig. 5(e) fills the hole from a nonlocal cube with the most similar structure, which is almost the same as the ground truth in Fig. 5(f).

5. CONCLUSION

Leveraging on graph signal processing, we propose an efficient 3D point cloud inpainting approach. The key observation is that point clouds exhibit non-local self-similarity in geometry. We thus propose to fill the holes in a point cloud from similar geometry. Specifically, we adopt fixed-sized cubes as the processing unit, and search for the most similar cube to the target cube which contains holes. The similarity metric is based on the direct component and the proposed anisotropic graph total variation of normals in the cubes. We then cast the hole-filling problem as a quadratic programming problem, based on the selected most similar cube and regularized by a graph-signal smoothness prior. Experimental results show that our algorithm outperforms three competing methods significantly. Future works include automatic hole detection and extension to inpainting the color attribute of point clouds.

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. Prez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 313–318, 2003.

[3] X. Shao, Z. Liu, and L. Hou-Qiang, "An isolated image inpainting approach based on the poisson equation," *Journal of Circuits & Systems*, vol. 13, no. 6, pp. 1–6, 2008.

[4] S. Muthukumar, N. D. Krishnan, P. Pasupathi, and S. Deepa, "Analysis of image inpainting techniques with exemplar, poisson, successive elimination and 8 pixel neighborhood methods," *International Journal of Computer Applications*, vol. 9, no. 11, pp. 24–28, 2011.

[5] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, "Simultaneous structure and texture image inpainting," *IEEE Transactions on Image Processing*, vol. 12, no. 8, pp. 882–889, 2003.

[6] A. Criminisi, P. Rez, and K. Toyama, *Region filling and object removal by exemplar-based image inpainting*, IEEE Press, 2004.

[7] Z. Xu and J. Sun, "Image inpainting by patch propagation using patch sparsity," *IEEE Transactions on Image Processing*, vol. 19, no. 5, pp. 1153–1165, 2010.

[8] D. Doria and R. J. Radke, "Filling large holes in lidar data by inpainting depth gradients," in *Computer Vision and Pattern Recognition Workshops*, 2012, pp. 65–72.

[9] D. Miao, J. Fu, Y. Lu, and S. Li, "Texture-assisted kinect depth inpainting," in *IEEE International Symposium on Circuits and Systems*, 2012, pp. 604–607.

[10] M. Kulkarni and A. N. Rajagopalan, "Depth inpainting by tensor voting," *Journal of the Optical Society of America A Optics Image Science & Vision*, vol. 30, no. 6, pp. 1155–65, 2013.

[11] W. Chen, H. Yue, J. Wang, and X. Wu, "An improved edge detection algorithm for depth map inpainting," *Optics & Lasers in Engineering*, vol. 55, no. 7, pp. 69–77, 2014.

[12] M. Centin and A. Signoroni, "Rameshcleaner: conservative fixing of triangular meshes," in *Stag: Smart TOOLS & Apps for Graphics*, 2015.

[13] L. S. Tekumalla and E. Cohen, "A hole-filling algorithm for triangular meshes," *School of Computing University of Utah*, 2014.

[14] C. Y. Chen and K. sY. Cheng, "A sharpness-dependent filter for recovering sharp features in repaired 3d mesh models," *IEEE Transactions on Visualization & Computer Graphics*, vol. 14, no. 1, pp. 200, 2008.

[15] R. Sagawa and K. Ikeuchi, "Hole filling of 3d model by flipping signs of signed distance field in adaptive resolution," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 30, no. 4, pp. 686–699, 2008.

[16] M. Attene, "A lightweight approach to repairing digitized polygon meshes," *The Visual Computer*, vol. 26, no. 11, pp. 1393–1406, 2010.

[17] 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.

[18] El Din Elshishiny, Fausto Bernardini, and Holly Edith Rushmeier, "System and method for hole filling in 3d models," 2007.

[19] W. Zhao, S. Gao, and H. Lin, "A robust hole-filling algorithm for triangular mesh," *Visual Computer*, vol. 23, no. 12, pp. 987–997, 2007.

[20] X. Li, L. Kang, and X. Hu, "Hole filling method of triangle mesh," in *International Conference on Multimedia Technology*, 2011, pp. 3085–3089.

[21] 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.

[22] 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.

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

[24] X. Wu and W. Chen, "A scattered point set hole-filling method based on boundary extension and convergence," in *Intelligent Control and Automation*, 2015, pp. 5329–5334.

[25] S. Friedman and I. Stamos, "Online facade reconstruction from dominant frequencies in structured point clouds," in *Computer Vision and Pattern Recognition Workshops*, 2012, pp. 1–8.

[26] F. Lozes, A. Elmoataz, and O. Lzoray, "Partial difference operators on weighted graphs for image processing on surfaces and point clouds," *IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society*, vol. 23, no. 9, pp. 3896–909, 2014.

[27] 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.

[28] 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.

[29] 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 Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[30] N. Shahid, N. Perraudin, V. Kalofolias, B. Ricaud, and P. Vandergheynst, "Pca using graph total variation," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2016.

[31] P. Berger, G. Hannak, and G. Matz, "Graph signal recovery via primal-dual algorithms for total variation minimization," *IEEE Journal of Selected Topics in Signal Processing*, vol. PP, no. 99, pp. 1–1, 2017.

[32] X. Liu, G. Cheung, X. Wu, and D. Zhao, "Inter-block consistent soft decoding of jpeg images with sparsity and graph-signal smoothness priors," in *IEEE International Conference on Image Processing*, 2015, pp. 1628–1632.

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

[34] 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.

[35] 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.

[36] 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.

[37] D. A. Spielman, "Lecture 2 of spectral graph theory and its applications," September 2004.

[38] A. Chambolle, *An Algorithm for Total Variation Minimization and Applications*, Kluwer Academic Publishers, 2004.

[39] 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.

[40] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *IEEE International Conference on Image Processing*, Beijing, China, September 2017.