Point Cloud Inpainting with Normal-based Feature Matching

Yu Shi, Chuanchuan Yang, *Senior Member, IEEE*

This work was funded by the National Key R&D Program of China under Grant No. 2019YFB1802904.

Y. Shi is with Department of Electronics, Peking University, Beijing, 100871, China (e-mail: shiyu@pku.edu.cn) and C. Yang is with the State Key Laboratory of Advanced Optical Communication Systems and Networks, Peking University, Beijing 100871, China (e-mail: yangchuanchuan@pku.edu.cn).

Corresponding author: Chuanchuan Yang.

***Abstract*—**With the development of LiDAR technology, point cloud as a data format for representing 3D objects has become more and more widely used. However, negative factors like occlusion or the unfavorable properties of the material surface will lead to the presence of geometric deficiencies, which usually exhibit as holes in point clouds. To solve this kind of problem, point cloud inpainting is proposed. In this paper, we propose an improved point cloud inpainting method which searches for the proper context for the detected hole and adopts a more efficient feature matching strategy to refine data source in hole-filling step. Experimental results with comparisons demonstrate its competitive effectiveness with an average 14% gain in GPSNR.

***Index Terms*—**Point cloud, inpainting, normal-based feature matching, signal processing.

I. Introduction

With the introduction of advanced 3D scanning techniques, 3D point cloud acquisition has gained increasing conveniency. Point clouds, as a basic form of 3D formats, consist of a set of points, each of which corresponds to an actual measuring point on the surface and contains original geometric information of the measured object. As a natural representation of arbitrarily-shaped objects, point clouds have been applied widely in various fields, such as cultural heritage restoration, navigation in self-driving, 3D model completion, etc.

However, currently the most advanced scanning techniques still can't reach enough high-fidelity to acquire complete data without deficiency, let alone the existence of negative factors such as occlusion, low reflectance coefficients of the surface, high grazing angles, etc [1]. Therefore, it inevitably leads to partial loss of collected data. Hence, point cloud inpainting is the key to restore missing data in order to represent 3D object more truthfully. Point cloud inpainting aims to infer or search for plausible voxel data from a partial point clouds to achieve the automatic inpainting of missing regions and make the inpainted object present a complete visual effect [1-2].

Existing methods for point cloud inpainting can be classified into two categories according to the data format used in the inpainting process [3-10]: 1) Mesh-based inpainting methods. 2) Point cloud-based inpainting methods. Generally, main approaches proposed focus on hole-filling of surfaces represented by meshes and the related problem of hole-filling for 3D point clouds has received less attention comparatively. The mesh-based method usually detects the hole boundary based on the topological structure of the triangular mesh model, and uses the geometric structure together with topological characteristics of the neighborhood to interpolate the hole area [3-7]. It is convenient and straightforward to locate hole region and extract the geometry as well as topology information based on the mesh model. On the other hand, the efficiency of these methods depends on the quality of the mesh, which means the time-consuming spent on constructing triangular mesh model will be greatly increased and the efficiency will be reduced as data size increases and the shape of surface complicates.

From this perspective, the point cloud-based method has a natural advantage as it directly deals with the scattered point cloud data, thus saves the cost for transformation between point cloud and mesh [8-10]. These works use boundary detection algorithms in unorganized point cloud to extract hole boundary and then inpaint the hole region. Ref.[11] samples auxiliary points in the neighborhood of holes and fills holes with algebraic surfaces like planes, cylinders, and spheres by approximating the geodesic offsets of the boundary. However, this method can only handle simple scenes where the object can be explained by a compact set of simple geometric shapes. [2] is a typical work which inpaints depth gradients to fix large holes of point cloud in the form of LiDAR data. Despite its good achievement in inpainting, it completes the inpainting in depth image and then transforms it back to 3D point cloud, which not only increases the cost but also inevitably introduces geometric loss during the switching. [12] presents a method to fill holes based on boundary extension and boundary convergence, which is conducted by moving boundary points along the extension direction into the hole region. Due to the reference information limited to the local neighborhood only, the result suffers from excess flatness compared to actual ground-truth. [13] proposes an inpainting method which retrieves the most similar region for the occlusion by exploiting the non-local self-similarity in the geometry of point clouds. Though this method has a low

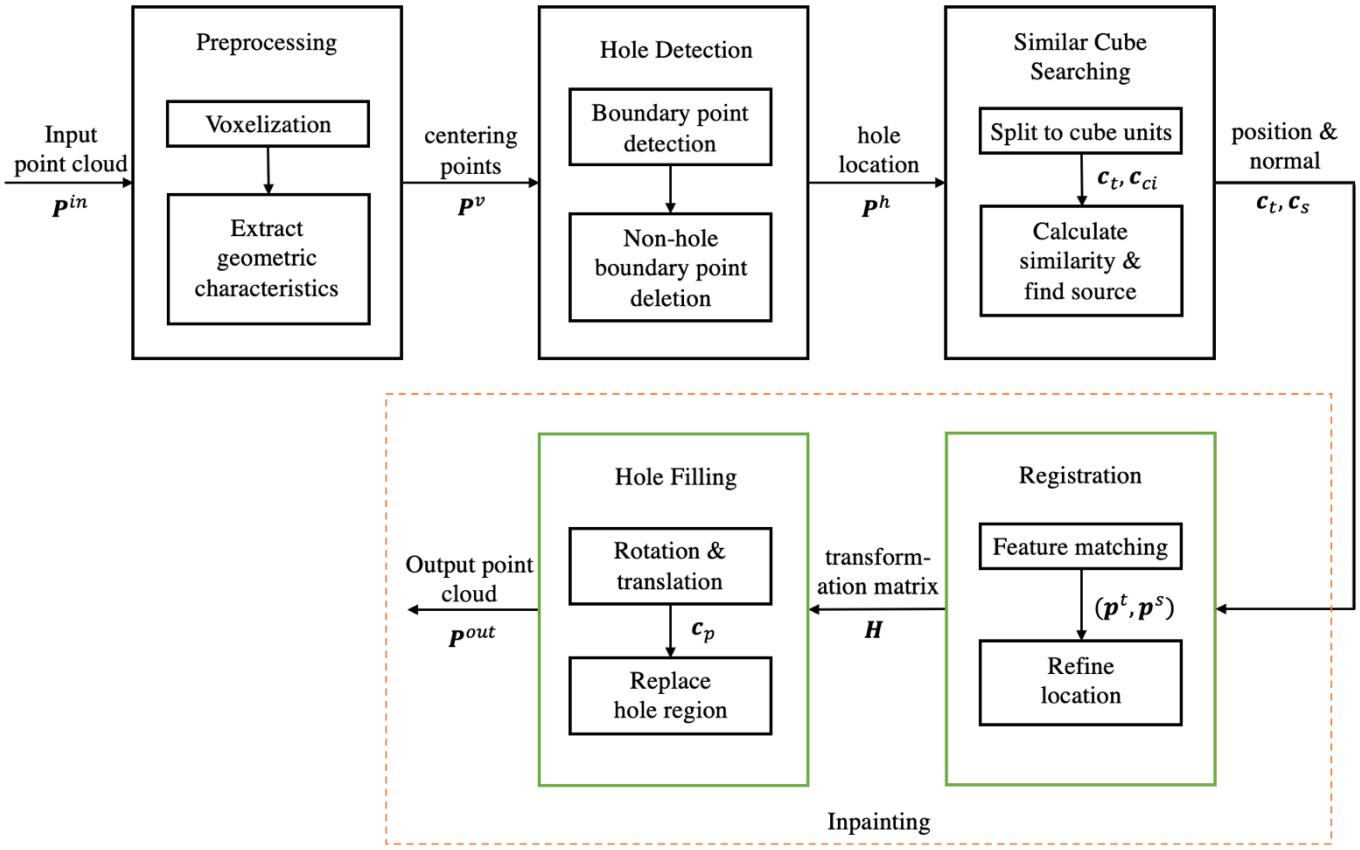


Fig. 1. Framework of proposed inpainting method, illustrating the complete process from input to output.

complexity by using local data source, the similarity is defined in the geometric structure, which leads to inaccuracy in the hole-filling step.

Considering the discussion above, when we need to inpaint holes caused by the limitation of scanning techniques but not the holes in the object itself, the optimal choice is to use local information around the hole as data source, scattered point cloud as data format to be processed. In order to reduce the occurrence of artifacts and improve the inpainted quality, in this paper we propose an improved point cloud inpainting method which aims to retrieve the most geometrically similar region for the occlusion by splitting point cloud into smaller processing unit, and construct the missed aera after refining the inpainting source which exploits position information of points and normal information of surface.

The proposed method provides efficient hole-inpainting with normal-based feature matching strategy, which allows better inpainting quality. Comparative experiments indicate its inpainted result has an average 14% gain in Geometric Peak Signal to Noise Ratio (GPSNR).

II. The Proposed Point Cloud Inpainting Method with Normal-Based Point Pair Matching Strategy

As shown in Fig 1, our proposed method consists of five following steps:

1) *Preprocessing:* Preprocess scattered input point cloud $\boldsymbol{P}^{in}$ with missing area, including voxelization and extraction of the geometric characteristics of point set using non-local graph. Then we obtain preprocessed point cloud $\boldsymbol{P}^{v}$, which is the input of hole detection.
2) *Hole Detection:* Based on the preprocessed point cloud $\boldsymbol{P}^{v}$, confirm location of holes with boundary detection algorithm. Having distinguished hole boundary point from other common points, point cloud $\boldsymbol{P}^{h}$ with hole location information is the input of subsequent similar cube searching.
3) *Similar Cube Searching:* Split point cloud $\boldsymbol{P}^{h}$ into smaller cubes, which will be treated as the basic unit to be processed in the subsequent steps. For each hole to be inpainted, mark the cube with hole as target cube $\boldsymbol{c}_t$ and search for the cube sharing the highest similarity with it among candidate cubes $\boldsymbol{c}_{ci}$, which is referred as the source cube $\boldsymbol{c}_s$.
4) *Registration:* Combine position and normal information to sift feature points in source cube $\boldsymbol{c}_s$ and target cube $\boldsymbol{c}_t$ as control points for registration, then compute transformation matrix $\boldsymbol{H}$ to optimize the error function in reference to control points.
5) *Hole Filling:* Transform source cube $\boldsymbol{c}_s$ to result cube $\boldsymbol{c}_p$ with transformation matrix $\boldsymbol{H}$, and replace target cube $\boldsymbol{c}_t$ with $\boldsymbol{c}_p$ as the output.

The novelties of this work are listed as follows:

1) In order to fully exploit the geometric and location information of known point cloud, after finding appropriate data source to inpaint, this paper makes improvement on the choice of feature point based on traditional ICP algorithm to refine the inpainting source before filling the hole. We not only consider the closest Euclidean distance between points, but also introduce point cloud shape

feature information based on normal directions at the data point, in which way to adapt better with the situation where the curvature of object's surface varies drastically.

2) Based on the feature points retrieved, we establish the transformation relationship between refined inpainting source and the replaced hole region to match the feature points by optimizing their geometric structure difference in 3D space, which can reduce the occurrence of artifacts and improve the fidelity.

The specific procedure will be elaborated as follows.

A. Preprocessing

The raw data required from scanning devices is unorganized, scattered point cloud P^{in}, whose irregularity will raise the computational complexity if we directly use it as the foundation of following inpainting. Out of this consideration, the preprocessing is the first step to conduct.

First of all, we voxelize the point cloud. KNN-search is applied to adjust the distance between each two points in neighborhood close to 1. Then, we perturb points in the original global coordinates to regular grids in the voxelized coordinates by substituting the set of points in a voxel with the centering point as long as there are points within the voxel. For the point set in each voxel, we build a non-local graph [14] based on the point in the geometric center and other original points to determine the property of centering point as voxelized result, as shown in Fig 2.

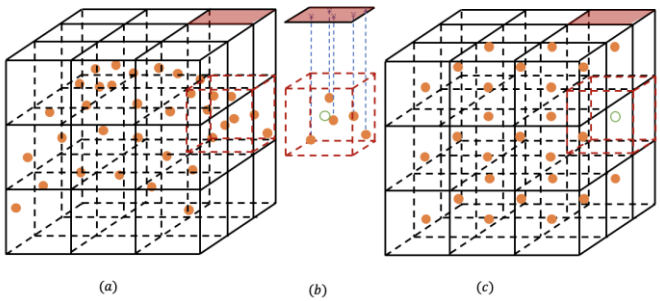


Fig. 2. (a) Original point cloud scattered in voxels. (b) Build a non-local graph based on the point in the geometric center and other original points. (c) Voxelized point cloud.

We first split original point cloud P^{in} into voxels with equal side length, and associate a vertex of a graph to each data point in the voxel which contains a set of data points $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \dots, \boldsymbol{p}_n\} \in \mathbb{R}^3$ to define a set of vertices $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$. Then we determine the neighbors of each vertex v_i according to its embedding data point $\boldsymbol{p}_i$ using the Euclidean distance $\mathcal{D}(v_i, v_j) = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2^2$. An undirected edge (v_i, v_j) is added between two vertices v_i and v_j if two corresponding data points satisfy particular geometric requirements. In our method, the requirement is that the distance between $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ is among the k smallest distances from either $\boldsymbol{p}_i$ or $\boldsymbol{p}_j$ to all the other data points which can be judged by the K-NN map established above. Given that the construction of such a graph will be computationally expensive for large point clouds, a Kd-tree is be used to speed up the K-NN search.

Once the graph is created, weights should be assigned to edges according to the similarities of vertices, which can be computed as:

$$weight(v_i, v) = \exp\left\{-\frac{\|\boldsymbol{p}_i - \boldsymbol{p}\|_2^2}{\sigma^2}\right\}, \tag{1}$$

where σ is a weighting parameter, the centering point is denoted as $\boldsymbol{p}$, its corresponding vertex as v, and the weight of edge (v_i, v) as $weight(v_i, v)$. The centering point $\boldsymbol{p}$ carries information which is calculated from geometric information of all points in the voxel, and in our case, it is normal at each point. Denote the geometric information of $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \dots, \boldsymbol{p}_n\}$ as $\{\boldsymbol{f}_1, \boldsymbol{f}_2, \dots, \boldsymbol{f}_n\}$, then we compute the geometric information $\boldsymbol{f}$ of $\boldsymbol{p}$ as:

$$\boldsymbol{f} = \frac{1}{\sum_i weight(v_i, v)} \sum_i weight(v_i, v)\, \boldsymbol{f}_i. \tag{2}$$

The preprocessed point cloud P^v with weighted centering point in each voxel is the input data for the subsequent procedure.

B. Hole Detection

Existing hole detection methods can fall into two categories, mesh-based and point-based. In our method, we use point-based method which detects all boundary points and filter out hole boundary points from them.

Firstly, we apply the BPD algorithm [15] to the point cloud P^v to find all the boundary points based on the fundamental idea that the point is a boundary point if it is not possible to find a path that surrounds it and passes through its k-nearest neighbor points, which is defined by K-NN search.

Furthermore, we apply Single-axis Searching method [16] to detect the points on the boundary of object's surface and remove them from the set of boundary points acquired before. The points left in the set are the boundary points of holes to be inpainted. Having distinguished hole boundary point from other common points, point cloud P^h with hole location information is to be the input of subsequent similar cube searching.

C. Source Cube Searching

Space division of point cloud data is an important step in the reconstruction of the surface. Hence, we split the point cloud $\boldsymbol{P}^h$ into cubes according to coordinates as minimum processing unit of the subsequent inpainting algorithm.

Now we have point cloud data as cubes, then we choose the cube with hole as target cube c_{tj} and cubes with enough points as candidate cube c_{ci}, where j denotes the j-th target cube and i denotes the i-th candidate cube of c_{tj}. Then the destination of this section turns out to find the most similar cube to c_{tj} among candidate cubes, which is referred to as the source cube c_{sj}. We use the definition of the similarity between two cubes based on 1) the direct component (DC) of the normal of points and 2) the anisotropic graph total variation (AGTV) of normal [13].

After computing the similarity between the $\boldsymbol{c}_{tj}$ and $\boldsymbol{c}_{ci}$, we get $\boldsymbol{c}_{sj}$ with the largest similarity. However, it can't be directly adopted for inpainting, given that the similarity defined above only considers the geometric structure and does not take the relative locations of them into account. Hence, we need to

perform further structure matching as point cloud registration, as discussed in the following.

D. Registration

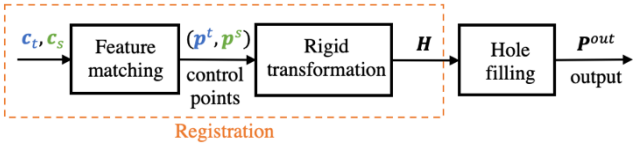


Fig. 3. Signal diagram in registration and hole filling section.

We consider matching the target cube and the source cube as a problem of point cloud registration, then it can be described as follows. A rigid body transformation is calculated to minimize the distance error between the source point cloud consisting of points in the source cube and the target point cloud consisting of points in the target cube.

At present, the most commonly used algorithms include the ICP [17] and its improved algorithms, which mainly contain two steps: the point pair calculation and optimal rigid body transformation including translation and rotation. These two steps execute alternately until the distance error between the source point cloud and the target point cloud drops below a certain threshold.

In the traditional ICP algorithm, the distance error is defined as a point-point function which needs to make point pair matching first. This leads to some problems if we use traditional ICP algorithm for registration directly in our inpainting method:

1) The numbers of points in target cube and source cube are different in most situations, so the one-to-one point pair matching inevitably cause one point in the target point cloud to be paired with multiple points in the source point cloud. Given that the algorithm will try to transform multiple points to the same point, which violates the requirements of rigid body transformation, it will bring large error even non-convergence in registration.
2) Considering the closest distance as the only criterion to do the point pair matching is defective and wasteful. Actually, point pairing using the closest point only considers the position of data points, and more geometric information is contained in other geometric features like normal directions.

Based on the discussion above, we propose to combine the spatial position information and normal information of points as general criterion when matching feature point. Thus, the position and the first-order shape feature are both used. Although higher-order information such as curvature can be utilized, it is often not accurate due to the noise and the complexity of calculation will raise.

The pseudocode of the feature matching algorithm is shown as follows:

The target cube is denoted by $\boldsymbol{c}_{tj}$=$\{\boldsymbol{p}_{i'}\}, 0 \le i' \le N$, with $\boldsymbol{p}_{i'} \in \mathbb{R}^3$ meaning the i'th point in the target cube, and the source target cube is denoted by $\boldsymbol{c}_{sj}$=$\{\boldsymbol{q}_{j'}\}, 0 \le j' \le M$, with $\boldsymbol{q}_{j'} \in \mathbb{R}^3$ meaning the j'-th point in the source cube.

Firstly, for each point $\boldsymbol{p}_{i'}$ in c_{tj}, traverse $\boldsymbol{q}_{j'}$ in $\boldsymbol{c}_{sj}$, and calculate the distance $d_{i',j'}$ between $\boldsymbol{p}_{i'}$ and $\boldsymbol{q}_{j'}$, the angle $\theta_{i',j'}$ between normal $\boldsymbol{n}_{i'}$ at $\boldsymbol{p}_{i'}$ and normal $\boldsymbol{n}_{j'}$ at $\boldsymbol{q}_{j'}$,

Algorithm 1 Feature matching Algorithm

Input: Target cube $\mathbf{c}_{tj} = \{\mathbf{p}_{i'}\}, 0 \le i' \le N$;
Source cube $\mathbf{c}_{sj} = \{\mathbf{q}_{j'}\}, 0 \le j' \le M$;
Normal angle threshold ϵ between two points.
Output: Target point cloud $\mathbf{P}_j{}^t = \{\mathbf{p}_j{}^t{}_{i''}\}, 0 \le i'' \le |\mathbf{P}_j{}^t|$;
Source point cloud $\mathbf{P}_j{}^s = \{\mathbf{p}_j{}^s{}_{i''}\}, 0 \le i'' \le |\mathbf{P}_j{}^s|$;

```
for i' = 0 to N do
  CurrentMinDistance = 0;
  Index = 0;
  for j' = 0 to M do
    if (NormalAngle(p_i', q_j') < ε) then
      if (Distance(p_i', q_j') < CurrentMinDistance) then
        CurrentMinDistance = Distance(p_i', q_j');
        Index = i'
      end if
    end if
  end for
  AddToNewPointCloud(q_Index, P_j^s);
  AddToNewPointCloud(p_i', P_j^t);
end for
```

which are defined as:

$$d_{i',j'} = \left\|\boldsymbol{p}_{i'} - \boldsymbol{q}_{j'}\right\|_2^2 \tag{3}$$

$$\theta_{i',j'} = \langle \boldsymbol{n}_{i'}, \boldsymbol{n}_{j'} \rangle. \tag{4}$$

Among the point pairs whose angle is below the threshold ε, choose the pair which has smallest distance and mark the smallest distance as $d_{i'}$, which is defined as:

$$d_{i'} = \min\{d_{i',j'}\}, 0 \le j' \le M, \theta_{i',j'} < \varepsilon. \tag{5}$$

If there exists eligible $d_{i'}$, we consider matching successful. Add feature point-pair $(\boldsymbol{p}_{i'}, \boldsymbol{q}_{j'})$ to the set of best pair $S = \{(\boldsymbol{p}_1, \boldsymbol{q}_1), (\boldsymbol{p}_2, \boldsymbol{q}_2), \dots, (\boldsymbol{p}_k, \boldsymbol{q}_k)\}, \mathrm{k} \le \min(\mathrm{N}, \mathrm{M})$ and remove the already matched point $\boldsymbol{q}_{j'}$ from source cube. Repeat the steps above to search matching points in source cube for all the points in target cube, until the number of points in the source cube is reduced to zero or the target cube is traversed thoroughly.

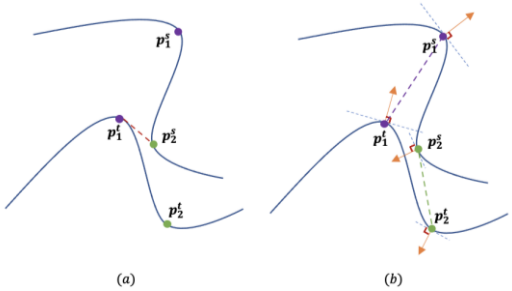
Secondly, we construct target point cloud $\boldsymbol{P}_j^t = \left\{\boldsymbol{p}_{j_{i''}}^t\right\} (0 \le i'' \le |\boldsymbol{P}_j^t|)$ and source point cloud $\boldsymbol{P}_j^s = \left\{\boldsymbol{p}_{j_{i''}}^s\right\} (0 \le i'' \le |\boldsymbol{P}_j^s|)$ separately with feature point-pairs in the set S. Based on the feature point retrieved, we can establish the transformation relationship between raw and refined inpainting source by computing the rigid body transformation matrix H_j which minimizes the distance error between $\boldsymbol{P}_j^s$ and $\boldsymbol{P}_j^t$. Rigid body transformation contains two steps including rotation and translation, denoted by rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$ independently. The resolution problem of $\boldsymbol{H}_j$ can be formulated as:

$$\boldsymbol{R}^*, \boldsymbol{t}^* = \underset{R,t}{\operatorname{argmin}} \frac{1}{|\boldsymbol{P}_j^s|} \sum_{i''=1}^{|\boldsymbol{P}_j^s|} \left\|\boldsymbol{p}_{j_{i''}}^t - \left(\boldsymbol{R} \cdot \boldsymbol{p}_{j_{i''}}^s + \boldsymbol{t}\right)\right\|^2. \tag{6}$$

Only if we have $\boldsymbol{R}^*, \boldsymbol{t}^*$, which is the optimum resolution among all $\boldsymbol{R}$ and $\boldsymbol{t}$, we can compute H_j which can be denoted as:

$$\boldsymbol{H}_j = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix}. \tag{7}$$

As shown in Fig 4, traditional method in (a) leads to inappropriate matching result $(\boldsymbol{p}_1^t, \boldsymbol{p}_2^s)$ while our method in (b) produces more reasonable result of treating $(\boldsymbol{p}_1^t, \boldsymbol{p}_1^s), (\boldsymbol{p}_2^t, \boldsymbol{p}_2^s)$ as two pairs.



(a) (b)

Fig. 4. (a) Only considering the nearest distance leads to inappropriate matching result $(\boldsymbol{p}_1^t, \boldsymbol{p}_2^s)$. (b) Combining normal direction and Euclidean distance leads to more reasonable result $(\boldsymbol{p}_1^t, \boldsymbol{p}_1^s), (\boldsymbol{p}_2^t, \boldsymbol{p}_2^s)$.

E. Hole Filling

We apply $\boldsymbol{H_j}$ to the source cube $\boldsymbol{c}_{sj}$ to get $\boldsymbol{c}_{pj}$ by equation

$$\boldsymbol{c}_{pj} = \boldsymbol{R} \cdot \boldsymbol{c}_{sj} + \boldsymbol{t}. \quad (8)$$

After the transformation, we remove the points outside the cube, and in order to maintain points on regular grids, further voxelize the new points. We replace the target cube $\boldsymbol{c}_{tj}$ with $\boldsymbol{c}_{pj}$ to finish this hole region's inpainting. Then we repeat steps C-D until all the holes are inpainted, and output the new point cloud as $\boldsymbol{P}^{out}$.

III. Experiments

A. Implementation

The proposed point cloud inpainting method is implemented with C++ on a computer with Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 8GB RAM.

B. Experimental Setup

The *proposed method is evaluated on several point cloud* datasets coming from JPEG Pleno [18]. We test on synthetic holes on the point cloud so as to compare with the ground truth. Geometric Peak Signal to Noise Ratio (GPSNR) is used to quantify the inpainting performance of the proposed method, which measures the errors between different point clouds based on PSNR [13]. A higher GPSNR means a lower gap between two point clouds.

C. Experimental Results

Fig 5 shows the subjective inpainting results for synthetic holes of *Sarah*, *David* and *Ricardo*. The inpainted results of Hu *et al* [13] exhibit artifacts and non-uniform density. In the case of *Ricardo*, Hu *et al* [13] covers the entire aera around the

(a)*Hole* (b)*Hu et al.* [16] (c)*Proposed* (d)*Ground Truth*

Fig. 5. Inpainting results by proposed method compared with those by [13]. Original data and ground truth are also presented.

hole but generates redundant points. In the case of *David*, Hu *et al* [13] doesn't match the texture well and the result presents lower consistency with the ground truth. Compared with Hu *et al* [13], our proposed method manages to inpaint holes with appropriate geometric structure consistent with existing points and displays higher smoothness over the missing region.

Table I demonstrates the objective results measured by GPSNR in datasets coming from JPEG Pleno [18]. We find that our proposed method achieves obviously better performance and shows competitive potential significantly on the indicators of GPSNR by an average of 14% over Hu *et al* [13].

TABLE I
PERFORMANCE COMPARISON IN GPSNR(DB)

Dataset	Hu et al	Proposed
Sarah	44.3576	53.4859
David	45.2667	50.1346
Ricardo	41.4784	46.7880

IV. CONCLUSION

In this paper, we propose an improved point cloud inpainting method with normal-based feature point matching strategy. Firstly, it makes refinement on the inpainting source by retrieving feature points, which combine the location information and 3D geometric feature information of point clouds in order to adapt better to the situation where the inpainting job involves objects whose surface curvatures vary drastically. Secondly, based on the feature points retrieved, we establish the transformation relationship between refined inpainting source and the replaced hole region to match the feature points by optimizing their geometric structure difference in 3D space.

Experiments prove that in comparison to existed point cloud inpainting methods, the proposed method reduces the occurrence of artifacts and improves the fidelity, which means better inpainting quality.

REFERENCES

[1] Dinesh, C., Bajic, I. V., Cheung, G. (2018). Adaptive non-rigid inpainting of 3d point cloud geometry. *IEEE Signal Processing Letters, 25*(6), 878-882.

[2] Doria, D., Radke, R. J. (2012). Filling large holes in LiDAR data by inpainting depth gradients. *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE. https://doi.org/10.1109/CVPRW.2012.6238916

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

[4] Sahay, P., Rajagopalan, A. N. (2015). Geometric inpainting of 3D structures. *Computer Vision & Pattern Recognition Workshops*(pp.1-7). IEEE.

[5] Lozes, F., Elmoataz, A., Lezoray, O. (2015). Pde-based graph signal processing for 3-d color point clouds: opportunities for cultural heritage. *Signal Processing Magazine IEEE*. https://doi.org/10.1109/MSP.2015.2408631

[6] Dinesh, C., Bajic, I. V., Cheung, G. (2017). Exemplar-based Framework for 3D Point Cloud Hole Filling. *IEEE VCIP 2017*. IEEE. https://doi.org/1 0.1109/VCIP.2017.8305070

[7] A, J. W., B, M. M. O. (2007). Filling holes on locally smooth surfaces reconstructed from point clouds - sciencedirect. *Image and Vision Computing, 25*(1), 103-113. https://doi.org/10.1016/j.imavis.2005.12.006

[8] Sagawa, Ryusuke, Katsushi Ikeuchi. (2008). Hole filling of a 3D model by flipping signs of a signed distance field in adaptive resolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 686-699. https://doi.org/10.1109/TAMI.2007.70726

[9] Hongbin, L., Wei, W. (2017). Feature preserving holes filling of scattered point cloud based on tensor voting. *IEEE International Conference on Signal & Image Processing*. IEEE. https://doi.org/10.1109/SIPROCESS.2016.7888293

[10] Setty, S., Ganihar, S. A., Mudenagudi, U. (2015). Framework for 3D object hole filling. *Fifth National Conference on Computer Vision*. IEEE. https://doi.org/10.1109/NCVPRIPG.2015.7490062

[11] Chalmoviansky, P., Jttler, B. (2003) Filling holes in point clouds. *Mathematics of Surfaces. Springer Berlin Heidelberg*, (pp. 196-212).

[12] Wu, X., Chen, W. (2014). A scattered point set hole-filling method based on boundary extension and convergence. *Intelligent Control and Automation (WCICA), 2014 11th World Congress*, (pp. 5329-5334). IEEE. https://doi.org/10.1109/WCICA.2014.7053624

[13] Wei, Zeqing, Zongming, Guo. (2019). Local frequency interpretation and non-local self-similarity on graph for point cloud inpainting. *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society*. https://doi.org/10.1109/TIP.2019.2906554

[14] Lozes F, Elmoataz, A., Lezoray, O. (2014). 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,23*(9),3896-909. https://doi.org/10.1109/TIP.2014.2336548

[15] Carmelo, M., Gareth, P. S., Rahul, S. (2018). Novel algorithms for 3d surface point cloud boundary detection and edge reconstruction. *Journal of Computational Design & Engineering* (1), 1. https://doi.org/10.1016/j.jcde.2018.02.001

[16] Zhu, RF., Fang, Y. (2012). Boundary Extraction and Simplification of Scattered Point Cloud Based on Improved Single-axis Searching Method. *Science Technology and Engineering*.

[17] Arun, K. S., Huang, T. S., Blostein, S. D. (1987). Least-squares fitting of 2 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-9*(5), 698-700. https://doi.org/10.1109/TPAMI.1987.4767965

[18] http://plenodb.jpeg.org